COMBINATORIAL DISAMBIGUATION

P. S. Newman
IBM Los Angeles Scientific Center
11601 Wilshire Boulevard
Los Angeles, CA 90025-1738

Abstract

The disambiguation of sentences is a combinatorial problem. This paper describes a method for treating it as such, directly, by adapting standard combinatorial search optimizations. Traditional disambiguation heuristics are applied but, instead of being embedded in individual decision procedures for specific types of ambiguities, they contribute to numerical weights that are considered by a single global optimizer. The result is increased power and simpler code. The method is being implemented for a machine translation project, but could be adapted to any natural language system.

1. Introduction

The disambiguation of sentences is a combinatorial problem. Identification of one word sense interacts with the identification of other word senses,

(1) He addressed the chair

and with constituent attachment,

(2) He shot some bucks with a rifle

Moreover, the attachment of one constituent interacts with the attachment of other constituents:

(3) She put the vase on the table in the living room

This paper describes a method of addressing the problem directly, by adapting standard search optimization techniques. In the first section we describe the core of the method, which applies a version of *best-first* search to a uniform representation of the set of possibilities. In the second section we relate the work to other approaches to preference-based disambiguation. The final sections describe how the representation may be obtained from a lexicon.

2. The Search Method

In the machine translation project for which this technique is being developed, disambiguation begins after a parser, specifically the PLNLP English Grammar by Jensen (1986), has identified one or more parses based primarily on syntactic considerations (including subcategorization). Words are disambiguated up to part-of-speech, but word senses are not identified. Individual parses may indicate alternative attachments of many kinds of constituents including prepositional phrases and relative clauses.

Beginning disambiguation only after a general parse has the advantage of making clear what all the possibilities are, thus allowing their investigation in an efficient order. Performing a full parse before disambiguation need not consume an inordinate amount of space or time; techniques such as those used by Jensen (default rightmost prepositional phrase attachment), and Tomita (1985) (*parse forests*) adequately control resource requirements.

The parser output is first transformed so that it is represented as a set of of *semantic choice points*. Each choice point generally represents a constituent. It contains a group of weighted *semantic alternatives* that represent the different ways in which word-senses of the constituent head can be associated semantically with word-senses of a higher level head. This allows word-sense and attachment alternatives to be treated uniformly.

Combinatorial disambiguation then selects the consistent combination of alternatives, one from each choice point, that yields the highest total weight. To illustrate the method, we use the extension of the classic example mentioned above:

(2) He shot some bucks with a rifle

A decomposition of the sentence into choice points c1, c2, and c3 is shown in Figure 1. (The illustration

assumes that "shot" and "bucks" have two meanings each, and ignores determiners.) Choice point "c1" gives the alternative syntactic and semantic possibilities for the attachment of the constituent "he" to its only possible head "shot". Alternative c11 is that "he" is the subject of "shoot1", with the semantic function "agent", and is given (for reasons which will be discussed later) the weight "3". Alternative c12 is similar, but the meaning used for "shoot" is "shoot2". Similar alternatives are used for the attachment (c2) of the object "some bucks" to its only possible head, "shoot". Alternative c23 represents the unlikely combinations.

Choice point c3 reflects the different possible attachments of "with a rifle"; the highest weight (3) is given to its attachment as an instrumental modifier of "shoot1". The other possibilities range from barely plausible to implausible and are weighted accordingly.

Having obtained this repesentation (whose construction is described in later sections), the next step is to establish the single-alternative choice points as given and to propagate any associated word-sense constraints to narrow or eliminate other alternatives. (This does not occur in our example.)

Then combinations of the remaining choices are searched using a variation of the A* best-first search method. See Nilsson (1980) for a thorough description of A*. Briefly, A* views combinatorial search as a problem of finding an optimal path from the root to a leaf of a tree whose nodes are the weighted alternatives to be combined. At any point in the process, the next node n to be added is one for which the potential

$$F(n) = G(n) + H(n)$$

is maximal over all possible additions. G(n) represents the value of the path up to and including n. H(n) represents an estimated upper bound for the value of the path below n (i.e., for the additional value which can be obtained)[1]. When a complete path is found by this method, it must be optimal. The efficiency of the method, i.e., the speed of finding an optimal path, depends on the accuracy of the estimates H(n).

To apply the method in our context, the search tree is defined in terms of levels, with each level corresponding to a choice point. Choice points are assigned to levels so that those which would probably be responsible for the greatest difference between estimated and actual H(n) in an arbitrary assignment are examined first. Looked at in another way, the assignment of choice points to levels is made so that those which will best differentiate among potential path scores are examined first.

This is done by (partially) ordering the choice points in descending order of their difference potential Dc, the difference in weight between their highest weighted alternative and the next alternative. If the highest weight is represented by two different alternatives, Dc = 0. Within this ordering the choice points are further ordered by weight differences between the 2nd and 3rd highest weighted alternatives, etc. For our example this results in choice point c3 ("with a rifle") being assigned to the highest level in the tree, followed by choice points c2 and then c1.

```
    c1.He
----------------------
c11  agt (he shoot1) 3                    (i.e., fired-at)
c12  agt (he shoot2) 3                    (i.e., wasted)

    c2.some bucks
----------------------
c21  goal (buck1 shoot1) 3                (i.e., male deer)
c22  goal (buck2 shoot2) 3                (i.e., dollar)
c23  goal (buck1 shoot2),(buck2 shoot1)  0

     c3.with a rifle
     ---------------
c31 inst (rifle shoot1) 3
c32 inst (rifle shoot2) 2
c33 togw (rifle (buck1,buck2))   0        (i.e., together-with)
c34 accm (rifle (shoot1,shoot2)) 0        (i.e., accompanied-by)
```

Figure 1: Choice points and alternatives

We also associate with each level = choice point the value Hc, which is the sum of the maximum weights that can be added by contributions below that choice point. This is needed by the algorithm to estimate potential path scores below a given node. For this example, the Hc values are:

```
H0: top=9   H3: rifle=6
H2: buck=3  H1: he=0
```

Then the best-first search is carried out. At each point in the search, the next node to be added is that which (a) is consistent in word-sense selection with previous choices, and (b) has the highest potential. The potential is calculated as the accumulated weight down to (and including) that node plus Hc for that level. A diagram of the procedure, as applied to the example, is shown in Figure 2. The first node to be added is "with rifle shoot!", which has the highest potential. At that point, the highest weighted consistent alternative is c21, etc.

While the set of choice points implies that there are (4 x 3 x 2) = 24 paths to be searched, only one is pursued to any distance. Thus while the approach takes a combinatorial view of the problem, it does so without loss of efficiency.

When a full path is found, it is examined for semantic consistency (beyond word-sense consistency). The checks made include: (a) ensuring that the interpretation includes all required semantic functions for a word-sense (specified in the lexicon), and (b) ensuring that non-repeatable functions (e.g., the goal of an action) are not duplicated.

Even if the full path is found to be consistent, the search does not terminate immediately, but continues until it is certain that no other path can have an equal score. This will be true whenever the maximum potential for open nodes is less than the score for an already-found path. A more precise description of the algorithm is given in the Appendix.

When more than one path is found with the same high score, additional tests are applied. These tests include comparisons of surface proximity and, as this work is situated within a multi-target translation system, user queries in the source language, as outlined by Tomita (1985).

An extended version of the method is used in comparing alternate parses which differ in constituent composition, and thus are more easily analyzed as different parse trees, each with its own set of choice points. An example is the classic:

(4) Time flies like an arrow

(where the main verb can be any one of the first three words). In such cases, one set of choice points is constructed per parse tree. In general, the search alternates among trees, with the next node to be added being that with the greatest potential across trees. If such trees always had the same number of choice points, this would be the only revision needed. However, the number of choice points may differ, for one thing because the parser may have detected and condensed non-compositional compounds (e.g., "all the same") in one parse but not in another. For this reason the algorithm changes to evaluate paths not by total scores, but by average scores (i.e., the total scores divided by the number of choice points in the particular parse).

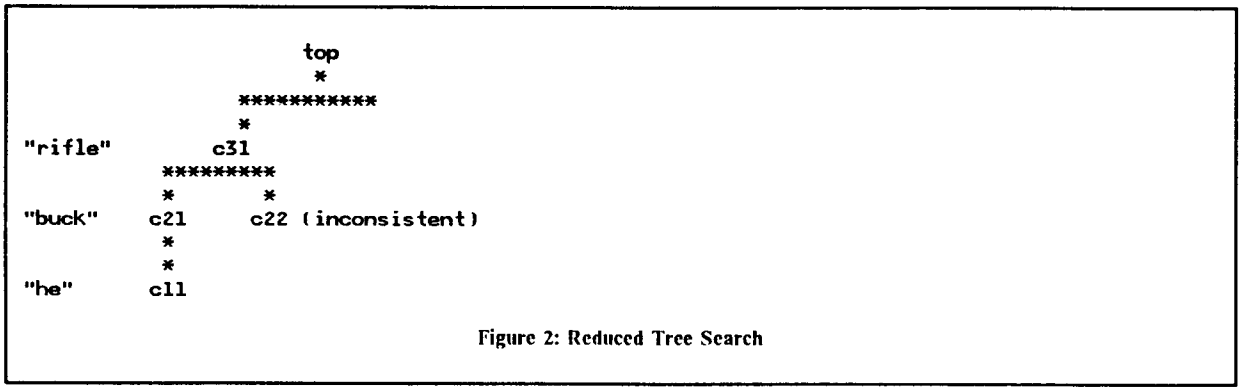

Figure 2: Reduced Tree Search

1 The basic A* algorithm is usually described as "expanding" (i.e., adding all immediate successors of) the most promising node. The variant described here, which is more appropriate to our situation (and also mentioned by Nilsson), adds a single node at each step.

3. Related Work

There seems to be little work which directly addresses the combinatorial problem. First, there is considerable work in preference-related disambiguation that assumes, at least for purposes of discussion, that individual disambiguation problems can be addressed in isolation. For example, treatments of prepositional phrase attachment by by Dahlgren and McDowell (1986) and Wilks et. al. (1985) propose methods of finding the "best" resolution of a single attachment problem by finding the first preference which is satisfied in some recommended order of rule application. Other types of ambiguity, and other instances of the same type, are assumed to have been resolved. This type of work contributes interesting insights, but cannot be used directly.

One type of more realistic treatment, which might be called the *deferred decision* approach, is exemplified by Hirst (1983). When, in the course of a parse, an immediate decision about a word sense or attachment cannot be made, a set of alternative possibilities is developed. The possibility sets are gradually narrowed by propagating the implications of both decisions and narrowings of other possibility sets.

This approach has a number of problems. First, it is susceptible to error in semantic "garden path" situations, as early decisions may not be justifiable in the context. For example, in processing

(5) He shot a hundred bucks with one rifle.

a particular expert might decide on "dollars" for bucks, because of the modifier "hundred", before the prepositional phrase is processed. Also, it is difficult to see how versions of this method could be extended to deal with comparing alternate parses where the alternatives are not just ones of attaching constituents, but of deciding what the constituents are in the first place.

A full-scale deferred-decision approach also has the potential of significant design complexity (the Hirst version is explicitly limited), as each type of decision procedure (for words and for different kinds of attachments) must be able to understand and process the implications of the results of other kinds of decision procedures.

Underlying these problems is the lack of quantification of alternatives, which allows for comparison of combinations.

There are, however, early and more recent approaches which do apply numeric weights to sentence analysis. Early approaches using weights applied them primarily to judge alternative parses. Syntactically-oriented approaches in this vein attached weights to phrase structure grammar rules (Robinson 1975, 1980) or ATN arcs (Bates 1976). Some approaches of this period focussing on semantic expectations were those of Wilks (1975) and Maxwell and Tuggle (1975), which employed scores expressing the number of expected dependents present in an interpretation. An ambitious approach combining different kinds of weights and cumulative scores, described by Walker and Paxton et. al. (1977), included heuristics to select the most promising subtrees for earlier development, to avoid running out of space before a "best" tree can be found. However, except for this type of provision, none of the early approaches using weights seem to address the combinatorial problem.[2]

A contemporary approach for thorough syntactic and semantic disambiguation using weights is described by L. Schubert (1986). During a left-to-right parse, individual attachments are weighted based on a list of considerations including preference, relative sentential position, frames/scripts, and salience in the current context. The multiple evolving parse trees are rated by summing their contained weights, and the combinatorial problem is controlled by retaining only the two highest scoring parses of any complete phrases.

This approach is interesting, although some details are vague[3]. However, the post-parse application of A* described in this paper obtains the benefits of such a within-parse approach without its deficiences in that: (a) combinatorial computations of weights and word-sense consistencies are avoided except when warranted by total sentence information, and (b) there is no possibility of early erroneous discarding of alternatives.

---

2 Heidorn (1982) provides a good summary of early work in weight-based analysis, as well as a weight-oriented approach to attachment decisions based on syntactic considerations only.

3 No examples are given, so it is unclear whether a parse for a phrase or part thereof represents only one interpretation, or all interpretations having the same structure, scored by the most likely interpretation. The former is obviously inadequate (e.g., for highly ambiguous subject NPs like "The stands"), while the latter seems to require either the calculation of all alternative cumulative scores, or recalculation of scores if an interpretation fails.

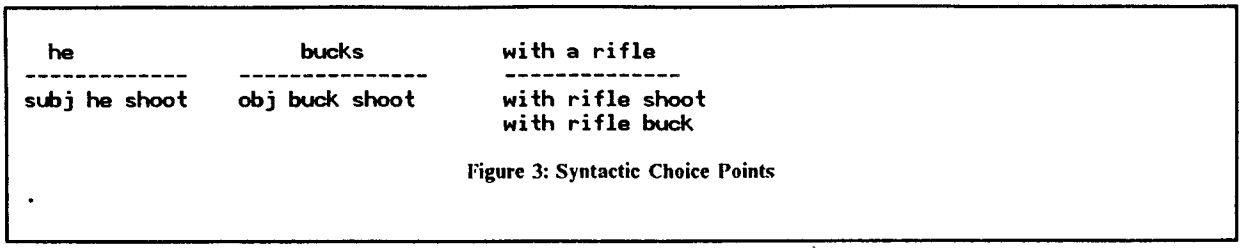

Figure 3: Syntactic Choice Points

One other parser-based work should be noted, that of Wittenburg (1986), as it is explicitly based on A*. The intent and content of the method is quite different from that described here. It is situated within a chart-parser for a categorial grammar, and the intent is to limit parsing expense by selecting that rule for next application which has the least likelihood of leading to an incomplete tree. While selectional preference is mentioned in passing as a possible heuristic, the heuristics discussed in any depth are grammar oriented, and operate on the immediate facts of the situation, rather than on informed estimates of total parse scores.

It should be also be mentioned that the representation of alternatives in schemes which combine syntactic and semantic disambiguation is rarely discussed, although maintaining a consistent representation of the relationships among word-sense and attachment alternatives is fundamental to a systematic treatment of the problem. An exception is the discussion by K. Schubert (1986), who describes a representation for alternatives with some affinities to that described here[4].

The information limitations of disambiguation during parsing are not found in *spreading-activation* approaches, exemplified by Charniak (1986), Cottrell and Small (1984), and Waltz and Pollack (1985). These approaches are still in the experimental stage, and are primarily intended for parallel hardware, while the A* algorithm used in this paper is designed for conventional serial hardware. But, in a sense, these approaches reinforce the main point of this paper: they argue for a single global technique for optimized combinatorial disambiguation based on all available information.

4. Preparing Semantic Choices

Having described how the choice points are used, we address their development. Two steps are involved: (1) the development of syntactic choice points, and (2) the development of semantic choice points. The first step transforms the parse-level syntactic functions into a form appropriate to the second step, which is the application of the lexicon to those functions to obtain the semantic alternatives.

In our example, the first step is a simple one. Syntactic relationships among constituents are transformed into syntactic relationships among head words, and the syntactic relationships are refined, so that "ppmod" is replaced by the actual prepositions used. The result of this step is shown in Figure 3. The development of syntactic choice points for some other types of constituents is more complex. Before discussing these situations, we discuss step 2: application of the lexicon to the syntactic choice points to obtain the semantic choice points, i.e., those shown in Figure 1.

4.1 The Lexicon

The lexicon contains entries for word stems (distinguished by part-of-speech), linked to word-sense entries, which are the lowest level "concepts". Concept entries are linked to superordinate concept entries, forming a lattice. Concept entries include a set of concept features (e.g. a list of superordinate concepts), and one or more rules for each syntactic function associated with the concept. The more relevant parts of the lexicon entries for the concepts used in the ongoing example are shown in Figure 4. The "classes" are lists of superordinate concepts. The syntactic function rules have the form:

```
synfun dependent head semfun weight
```

Thus the first rule under "shoot1" indicates that word-senses falling into the class "animate" are its preferred objects, with the weight 3, and the combination is given the semantic function "goal". The concept entry

4 However, the weighting scheme is different, and rather interesting. The reference does not discuss the selection of a particular combination of alternatives in any detail, but it appears to be based on the presence in a combination of one (or more?) highly weighted alternative (or alternatives?).

```
shoot1 classes (humanact, transv)
  (object animate shoot1 goal 3)
  (with   firearm shoot1 inst 3)

humanact classes (act...)
   (subj human humanact agt  3)
   (with human humanact accm 3)

buck1 classes (animate...)
buck2 classes (money..)
```

```
shoot2 classes (humanact, transv)
  (object money shoot2 goal 3)

transv
  (object thing transv goal 0)
act
  (with   tool  act inst 2)

rifle classes (firearm...)
firearm classes (tool...)
```

Figure 4: Lexicon Entries

"humanact" contains other rules applicable to shoot1 and other verb-senses taking human actors.

Weights are actually assigned symbolically, to allow experimentation. Current settings are as follows:

- Rules for idioms (e.g., kick the bucket), 4.
- Rules for more general selectional preferences, 3.
- Rules for acceptable but not preferred alternatives (e.g., locative prepositional phrases attached to arbitrary actions), 2.
- Very general attachments (e.g., "nounmod noun1 noun2), 0. These allow for uninterpreted metaphoric usage.[5]

One major objective in assigning weights to ensure that combinations of very high (idiom) weights together with very low weights do not outscore more balanced combinations. Thus, for:

(6) He kicked the ugly bucket

weights such as:

```
subj he      kicked1   3
obj  bucket1 kicked1   4
adjm ugly    bucket1   0

subj he      kicked2   3
obj  bucket2 kicked2   3
adjm ugly    bucket2   2
```

provide the necessary balance. (Here kicked1 is the idiomatic interpretation, and bucket1 is a word-sense of bucket used only in that interpretation.)

By convention, rules for syntactic functions are assigned, by class, to entries for specific kinds of concepts. Thus rules for verb-attached arguments or prepositional phrases are stored with verbs or verb classes. Adjective-noun rules are generally associated with adjectives, and noun-noun rules with the right-hand nouns (the next section discusses the treatment of noun-phrase choice points in somewhat greater detail).

Lexicon entries also contain additional information. First, a list of required syntactic functions is generally associated with word-senses. Also, syntactic function rules may contain additional conditions limiting their applicability. For example, a combination "noun1 IN noun2" might be limited to apply only if the second concept denotes an object larger than the first.

4.2 Lexicon Application

Given these lexicon entries, the set of semantic alternatives corresponding to each syntactic alternative "synfun word1 word2" may be derived. The goal of the derivation process is to account for all possible combinations of word-senses for word1 and word2 related by the syntactic function "synfun". To do this, all concept entries containing potentially applicable rules are searched. For each satisfied rule found, an alternative is created of the form:

```
semantic-relation sensepairs weight
```

where "sensepairs" is a list of pairs. Each pair is in the form ((di,dj,...) (hm,hn,...)), where the di's are senses of the dependent participant of the function,

5 Obtaining the necessary lexicon information is of course a major problem. But there is significant contemporary work in the automatic or semi-automatic derivation of that information. For example, the aproached described by Jensen and Binot (1986) obtains attachment preference information by parsing dictionary definitions.

and the hi's are senses of the headword. The semantic relationship is stated to apply to all combinations of word-senses in the cross-products of those lists. For the example sentence, this process would obtain essentially the alternatives shown in Figure 1, except that alternative c23 would first be expressed as:

```
obj ((buck1,buck2)(shoot1,shoot2)) 0
```

The last step in the process reduces this result. If some of the word-sense combinations are also found in an alternative of higher weight, the "dominated" combinations are deleted. And if all word sense combinations are so dominated, the alternative is deleted. In this way alternative c23 is reduced to its final form.

After the semantic choice point list is completed, the search algorithm is applied as described above.

5. Preparing Syntactic Choices

In the example above, the preparation of syntactic choice points from parser output was very simple. Assuming an input choice point for a constituent to be a list of (one or more) parser-provided alternative relationships with an immediately containing constituent, the process consisted of obtaining the headwords of each constituent, and of substituting literal prepositions for the general syntactic function "ppmod".

However, in other cases this step is a more significant one. In the lexicon, selectional preferences are expressed in terms of the syntactic functions of some basic constituent types. For example, verb preferences are expressed in terms of the syntactic functions of active-voice, declarative main clauses, with dependents in unmarked positions. Adjective preferences are expressed in terms of classes of nouns occurring in the relationship "adjective noun". But there are many other syntactic relationships whose disambiguation depends on this information. The major function of the syntactic choice identification step is to re-express, or "normalize" input syntactic relationsips in terms of the relationships assumed by the lexicon. For example, passive constructions such as:

(7) The bucks were shot with a rifle

are normalized by replacing the choice "subj buck shoot" with "object buck shoot". (A lexicon condition barring or lowering preference for the "gambling" interpretation in the passive voice is also needed here.) In ditransitive cases both indirect and direct object functions are used as alternatives.

Thus the sequence of deriving semantic choice points consists of two significant steps, which may be depicted in terms of results as shown in Figure 5.

The transformation of input syntactic choice points to normalized syntactic choice points is governed by declarative specifications indicating, for each syntactic function, how its choice points are to be transformed. The changes are specified as replacements for one or more positions of the choice triple. For example, some of the "subj" rules are:

```
(subj
  (test (voicepassive)
     synfun 'obj))
  (test (not (voicepassive))
     synfun 'subj) ...
```

stating that for the input function "subj", if the specified test (voicepassive) succeeds, then "obj" is used for the synfun part of the normalized choice. The additional rule is used to ensure that the "subject" function is retained only for the active voice.

Additional applications of these transformations include those for modifiers of nominalized verbs, attributive clauses, and relative clauses.

Input Syn Chptl	Normalized Syn Chptl	Semantic Chptl
Choice C11	Choice C111	Choice C1111
		Choice C1112
	Choice C112	Choice C1121
		Choice C1122

Figure 5: Steps in Semantic Choice Point Derivation

Noun phrases whose heads are nominalized verbs are addressed by adding choice points corresponding to verb arguments. Thus for

(8) The bucks' shooting .....

the alternative "nounmod bucks shooting" is expanded to include the alternatives "subj bucks shooting" and "obj bucks shooting". Then, during lexical processing, rules for word-senses of the noun "shooting" having an associated verb are understood as expanded to include the expected verb arguments.

Attributive clauses such as:

(9) The bucks were handsome.

are transformed to allow the application of adjective information. Here "obj handsome were" is transformed to "adjmod handsome bucks".

For relative clauses, the core of the transformation expresses alternative attachments of the relative clause as alternative connections between the head of the relative clause, and the possible fillers of the gap position. (Relative clauses with multiple gaps are generally handled in separate parses.) Thus for:

(10) The rifle above the mantle that the bucks were shot with...

transformations produce the alternatives "with shoot rifle" and "with shoot mantle".

The handling of relative clauses, however, is more complicated than this, as it is desirable to also use information from the relative pronoun (if present) for the disambiguation. Two initial choice points are involved, one attaching the relative clause to its higher level head, and one attaching the gap to its head. The first is expanded to to obtain relationships "relp that rifle", "relp that mantle", and the other to obtain the "with ...." relationships. And an additional consistency check is made during the tree search, beyond word-sense consistency, to keep the choices consistent.

It should be noted that the transformation rules for syntactic choice points also include "fixup specifications" (not shown above) indicating how result semantic functions and attachments are to be modified if the transformed alternatives are used in the final interpretation. For example, to "fixup" the results of transforming attributive clauses, the noun-modifier semantic role is replaced with one suitable to a direct role in the clause.

6. Concluding Remarks

This paper has summarized a three-step method for optimized combinatorial preference based disambiguation:

1. obtaining syntactic choice points, with alternatives stated as syntactic functions relating words.
2. transformation into semantic choice points with alternatives stated as weighted semantic functions relating word-senses, via lexicon search.
3. application of A* to search alternative combinations.

This method, currently being implemented in the context of a multi-target machine translation system, is more powerful and systematic than approaches using isolated or interacting decision procedures, and thus is easier to modify and extend with new heuristics as desired.

The method is applicable to any post-parse disambiguation situation, and can be taken as an argument for that approach. To demonstrate this, we first note that aspects of the method are useful for within-parse disambiguation. In any realistic scheme, decisions must often be deferred, making two aspects of the method relevant: (a) the unified way of representing word sense and attachment alternatives and their interdependency, and (b) the explicit, additive weights. Explicit additive weights substitute for elaborate, case-specific rules, and also make possible a systematic treatment of alternative parses which differ in more than word-senses and attachments.

However, using weighted attachments for within-parse disambiguation requires calculating the summed weights of, and examining the consistency of, all combinations encountered whose elements cannot be discarded (assuming some good criteria for discarding can be found). Deferring disambiguation until after the parse allows for optimized searching of alternatives, as described above, to significantly limit the number of combinations examined.

Future work in this direction will include refining the weighting criteria, extending the method to deal with anaphoric references (using considerations developed by, for example, Jensen and Zadrozny (1987), and integrating a treatment of non-frozen metaphor.

7.Acknowledgements

I thank John Sowa, Maria Fuenmayor, and Shelley Smith for their careful reviews and many helpful suggestions. Also, I thank Peter Woon for his patient managerial support of this project.

8. References

1. Bates, Madeleine 1976. "Syntax in Automatic Speech Understanding", *Am. J. Comp. Ling.* Microfiche 45.
2. Charniak, Eugene 1986. "A Neat Theory of Marker Passing" *Proc. AAAI-86* 584-588
3. Cottrell, Garrison W. and Steven L. Small 1984. "Viewing Parsing as Word Sense Discrimination: A Connectionist Approach", in B.G. Bara and G. Guida (eds), *Computation Models of Natural Language Processing*, Elsevier Science Publishers B.V.
4. Dahlgren, Kathleen and Joyce McDowell 1986. "Using Commonsense Knowledge to Disambiguate Prepositional Phrase Modifiers", *Proc. AAAI-86*, 589-593
5. Heidorn, George 1982. "Experience with an Easily Computed Metric for Ranking Alternative Parses" *Proc. 20th Annual Meeting of the ACL*, June 1982
6. Hirst, Graeme 1983. "Semantic Interpretation Against Ambiguity", Technical Report CS-83-25, Brown University, December 1983
7. Jensen, Karen 1986. "Parsing Strategies in a Broad-coverage Grammar of English", IBM Research Report RC 12147, 1986
8. Jensen, Karen and Jean-Louis Binot 1987. "A Semantic Expert Using an Online Standard Dictionary", *Proc. IJCAI-87*, 709-714
9. Jensen, Karen and Wlodzimierz Zadrozny 1987. "The Semantics of Paragraphs", presented at *Logic and Linguistics*, Stanford, July 1987.
10. Maxwell, B.D and F. D. Tuggle 1975. "Toward a Natural Language Question Answering Facility", *Am. J. Comp. Ling.*, Microfiche 61.
11. Nilsson, Nils J. 1980. *Principles of Artificial Intelligence*, Tioga Publishing Co.
12. Robinson, Jane J. 1982. "DIAGRAM: A Grammar for Dialogues", *Comm. ACM* Vol 25 No 1, 27-47
13. Schubert, Klaus 1986. "Linguistic and Extra-Linguistic Knowledge" *Computers and Translation* Vol 1, No 3, July-September 1986
14. Schubert, Lenhart K. 1986. "Are There Preference Trade-offs in Attachment Decisions", *Proc. AAAI-86*, 601-605
15. Tomita, Masaru 1984. "Disambiguating Grammatically Ambiguous Sentences by Asking", *Proc. COLING 84*
16. Tomita, Masaru 1985. "An Efficient Context-Free Parsing Algorithm for Natural Languages", *Proc. IJCAI-85*,756-763
17. Walker, Donald E. and William H. Paxton with Gary G. Hendrix, Ann E. Robinson, Jane J. Robinson, Jonathan Slocum 1977. "Procedures for Integrating Knowledge in a Speech Understanding System", SRI Technical Note 143.
18. Waltz, David L. and J. B. Pollack 1985. "Massively Parallel Parsing: A Strongly Interactive Model of Natural Language Interpretation", *Cognitive Science* Vol 9,No 1, January-March 1985, 51-74
19. Wilks, Yorick, Xiuming Huang, and Dan Fass 1985. "Syntax, Preference and Right Attachment", *Proc. IJCAI-85* 779-784
20. Wilks, Yorick 1975. "An Intelligent Analyzer and Understander of English", *Comm. ACM*, Vol 18 No 5, 264-274
21. Wittenburg, Kent 1986. "A Parser for Portable NL Interfaces Using Graph-Unification-Based Grammars", *Proc. AAAI-86*, 1053-1058

9. Appendix: Search Algorithm

Figure 6 describes the step by step application of A* to searching semantic choices.

Assume an "open list" containing, for each node n with an unexamined child, the following information:

1. The list of choices made on the path up to and including n.
2. The set of constraints on word senses imposed by nodes on the path.
3. The index of the highest weighted unexamined child (choice at next level) of n, where choices within levels are sorted by descending weight.
4. The potential Fc = An + Wc + Hc for that child, where An is the accumulated weight on the path up to and including n, Wc is the weight of the child, and Hc is the upper bound on the cumulative potential for paths below the child.

Then the following algorithm is used to search the tree.

1. Put an entry for the dummy "top" node in the open list, with path=self, index of first child, and its potential Fc.
2. Find the node n in the "open list" which has the highest potential Fc for a child node.
3. If a full path has already been found, and Fc is lower than the total weight for that path, the search is over.
4. Otherwise check the consistency of the designated child in entry n. If it is consistent, add an open list entry for the child, including a new, more constrained consistency requirement.
5. Whether or not the designated child is consistent, determine if there are any unexamined children of node n. If so, modify entry n accordingly. Otherwise remove entry n from the open list.
6. If there is a new entry, and it represents a completed path, remove it from the open list and perform additional consistency checks. If the checks fail, ignore the new path. If they succeed, record the path and its score as a competing alternative.
7. Return to Step 2.

Figure 6: Search Algorithm