Towards Incremental Disambiguation with a Generalized Discrimination Network

Manabu Okumura and Hozumi Tanaka
Department of Computer Science,
Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo, 152, Japan
dora@cs.titech.ac.jp

Abstract

Semantic disambiguation is a difficult problem in natural language analysis. A better strategy for semantic disambiguation is to accumulate constraints obtained during the analytical process of a sentence, and disambiguate as early as possible the meaning incrementally using the constraints. We propose such a computational model of natural language analysis, and call it the 'incremental disambiguation model.' The semantic disambiguation process can be equated with the downward traversal of a discrimination network. However, the discrimination network has a problem in that it cannot be traversed unless constraints are entered in an a priori-fixed order. In general, the order in which constraints are obtained cannot be a priori fixed, so it is not always possible to traverse the network downward during the analytical process. In this paper, we propose a method which can traverse the discrimination network according to the order in which constraints are obtained incrementally during the analytical process. This order is independent of the a priori-fixed order of the network.

Introduction

Semantic disambiguation is a difficult problem in natural language analysis. The meaning of a word is ambiguous because it cannot be uniquely determined unless information about other words in the sentence is obtained.

A possible strategy for semantic disambiguation would be to determine the meaning of words at the end of the sentence. Yet this strategy might cause a combinatorial explosion of the number of total ambiguities if the sentence were long. We think it is impracticable for a natural language analysis system to determine the meaning of words from a number of candidates after it has finished reading the whole sentence. Another strategy, trying to force a decision with insufficient information, would be faced with serious uncertainties. A better strategy for semantic disambiguation would be to accumulate information(constraints) obtained during the analytical process of a sentence, and disambiguate as early as possible the meaning incrementally using the constraints. We propose such a computational model of natural language analysis, and call it the 'incremental disambiguation model[Mellish, 1985].'

The incremental disambiguation process is considered to be the refinement of ambiguous(undetermined) results of semantic processing by newly obtained constraints. Therefore, the incremental disambiguation approach allows us to deal with cases where it is impossible to disambiguate unless information about succeeding sentences is taken into account.

The semantic disambiguation process can be equated with the downward traversal of a discrimination network[Charniak *et al.*, 1980]. Using a discrimination network for semantic disambiguation has the following advantages:

- a discrimination network doesn't treat multiple word senses as being unrelated, and allows us to take into account interrelationships among multiple word senses;
- multiple word senses represented as elements of a list, the algorithm of selecting among them, that is, a linear search of the list, is very inefficient when candidate word senses are too numerous. On the other hand, the discrimination network's search algorithm is considered to be more efficient because the downward traversal from the root node to a leaf node which represents a word sense is guided by constraints which are labels of branches.

The partial traversal of a discrimination network is considered to be the representation of the ambiguous result of partial semantic processing which is easily represented by nodes in the network except leaf nodes. Making a new decision by additional constraints from subsequent inputs naturally corresponds to traversing from the current node further downward using newly obtained constraints. This method is therefore well suited for the incremental disambiguation approach.

However, the discrimination network has a problem in that it cannot be traversed unless constraints are

the root node to a leaf node which represents a word sense is guided by constraints which are labels of branches, so the search space can be gradually narrowed down[Lytinen, 1988]. This search takes time $O(l)$, where l is the height of the tree, and is independent of the number of word senses n[Aho *et al.*, 1983].

The ambiguity in a sentence is often not fully resolved unless information about subsequent inputs is taken into account. To cope with these cases, it must be possible to produce a partial semantic interpretation of a sentence and disambiguate it by additional constraints from subsequent inputs. This is the aim of the incremental disambiguation model.

The partial traversal of the discrimination network is considered to be the representation of the ambiguous result of partial semantic processing; the ambiguous result is easily represented by nodes in the network except leaf nodes. Making a new decision by additional constraints from subsequent inputs naturally corresponds to traversing from the current node further downward using newly obtained constraints. This method is therefore well suited for the incremental disambiguation approach[Moerdler and McKeown, 1988].

In the next section, we will describe some problems of past works which used the discrimination network.

Problems of using discrimination networks for semantic disambiguation

In the last section, we described the merits of using the discrimination network for semantic disambiguation, and asserted that the discrimination network is suited for the incremental disambiguation approach.

However, the discrimination network has a problem in that it cannot be traversed and the analysis may be suspended in some node unless constraints are entered in an a priori-fixed order. Because the network is traversed downward from the root node, constraints must be entered one by one from thoses which are labels of branches connected to the root node. This order depends on the original structure of the network. In the above example of the sentence 'John took a plane to London,' constraints must be entered in the order of S/John, O/plane, to/London. Unfortunately, surface variations such as passive forms make it difficult to assume that constraints are obtained in any fixed order during the analytical process. Therefore, it is not always possible to traverse the network downward during the analytical process. Adopting the semantic analysis method integrated with traditional bottom-up parsing[Matsumoto *et al.*, 1983], the natural order of the obtained constraints is O/plane, to/London, S/John for the above sentence. Traversal of Figure 1 is thus impossible. In addition, if the subject constraint is omitted, as in the case of passive sentences, traversal of the network will reach a deadlock.

To cope with this problem, a solution has been suggested where traversal of the network is performed after the total set of constraints is obtained. However, this strategy might cause combinatorial explosion of the number of total ambiguities, because semantic disambiguation is delayed after the analysis of the whole sentence is finished. This approach conflicts with the idea of the incremental disambiguation model.

WEP(Word Expert Parser)[Adriaens and Small, 1988] suggests another solution where special procedures such as demons[Charniak *et al.*, 1980] are provided for cases of irregular order of constraints, e.g. relative clause constructions. In general, the order of constraints obtained during the analytical process depends on the word order of the input sentence. So the degree of deviation from the a priori-fixed order is proportional to the degree of word-order freedom of the language to be analyzed. Therefore, this 'special procedure' approach seems to cope with languages having less word-order freedom such as English. However, in the case of languages having greater word-order freedom such as Japanese, the procedures for the cases of deviated order will be very large and their readability lost, if describable.

The taxonomic lattice[Woods, 1978, Bobrow and Webber, 1980] is a generalization of a discrimination tree which can be traversed independently of order. However, transforming a tree into a lattice makes internal representation very redundant and requires a lot of memory space in the computer system.

In the next section, we propose a method which can traverse the discrimination network according to the order in which constraints are obtained incrementally during the analytical process. We call our system a 'generalized discrimination network.' This approach is considered to be an implementation of the incremental disambiguation model with a discrimination network. It seems to have the same expressive power as the taxonomic lattice, but doesn't necessiate increased memory space because it uses the original 'tree' form as its representation.

Principles of generalized discrimination networks

Consider the discrimination network shown in Figure 2. Labels of branches stand for discrimination constraints. First, a numerical string is assigned to each node as a unique identifier. '1' is assigned to the root node. To each child node of the root node, an identifier of two digits $1i$(where i is an integer between 1 and n which represents the number of child nodes) is assigned. Similarly, to each child node of the node $1i_1i_2...i_m$, an identifier $1i_1i_2...i_mi$(where i is an integer between 1 and n which represents the number of child nodes) is assigned. To the nodes in Figure 2, identifiers are assigned as shown in Figure 3.

Second, to each node identifier a bit vector is attached which has the same length as the identifier and consists of 1's except for the leftmost and rightmost bits. To the identifiers in Figure 3, bit vectors are at-

entered in an a priori-fixed order. In general, the order in which constraints are obtained cannot be a priori fixed, so it is not always possible to traverse the network downward during the analytical process. In this paper, we propose a method which can traverse the discrimination network according to the order in which constraints are obtained incrementally during the analytical process. This order is independent of the a priori-fixed order of the network. This method is based on the notion of constraint logic programming and is implemented by extended unification. We call it a 'generalized discrimination network.'

In section two, the advantages and problems of using a discrimination network for semantic disambiguation are described and in section three, principles of generalized discrimination networks are presented. Finally, in section four, the merits of the generalized discrimination network are described.

Semantic disambiguation using discrimination networks

Figure 1 is a portion of the discrimination network which represents the word senses of the verb 'to take.' Each branch of the network has as its label a selectional restriction on surface cases such as subject(S), object(O), prepositions like 'with,' 'to,' and so on. Each leaf node of the network points to a unique word sense, which is represented by the underlined label[1]. Other nodes represent ambiguous word meanings which include all word senses corresponding to the leaf nodes below them, because from these, the further traversal along branches to multiple nodes are possible. The root node corresponds to the most ambiguous meaning: it is a representation which includes all leaves, namely all word senses.

The semantic disambiguation process using a discrimination network is a step by step downward traversal of the network from the root node to a leaf node guided by branches which satisfy the obtained constraints. In this process, semantically inappropriate alternatives are rejected and appropriate word senses are selected by virtue of information about other words in the sentence. The reaching of a leaf node means that the ambiguity has been fully resolved. We can say a verb is semantically ambiguous if a leaf node cannot be reached when the analysis of the whole sentence is finished. Reached nodes at that time are semantic representations of ambiguous verb meanings.

Consider the analytical process of the sentence 'John took a plane to London.' From the sentence, constraints such as [S/John, O/plane, to/London] are obtained. Traversal of Figure 1 guided by these constraints succeeds as follows: 'John' is human and satisfies a selectional restriction of subject, so node 1 is reached; it proceeds the same for 'plane' and 'London,'

[1]Some underlined labels are omitted in Figure 1 for clarity.

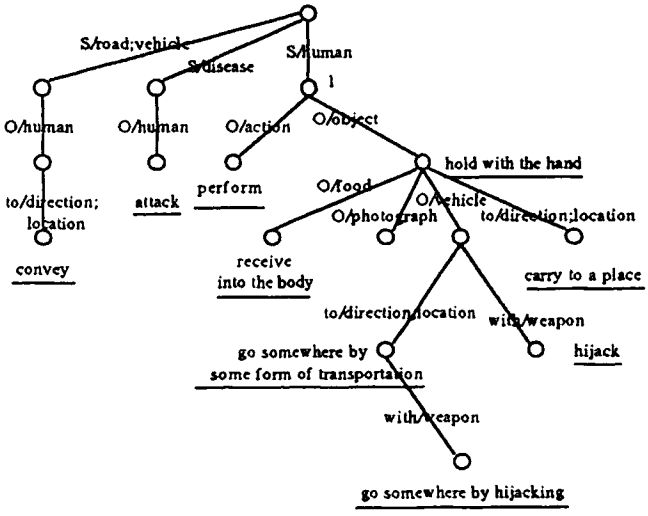

Figure 1: A portion of the discriminatrion network of the word senses of the verb 'to take'

and the node is reached representing the word sense 'going somewhere by some form of transportation.'

In the next two subsections, we describe the merits and problems of using a discrimination network for semantic disambiguation.

Merits of using discrimination networks for semantic disambiguation

Works such as [Jacobs, 1988, Moerdler and McKeown, 1988, Lytinen, 1988, Adriaens and Small, 1988] realize the semantic disambiguation process as a downward traversal of the discrimination network. In these works, the following merits of such a network are described:

- in traditional approaches to word sense ambiguities, such as Hirst's Polaroid Words[Hirst, 1987], multiple word senses tend to be treated as being unrelated to each other. The problem with this approach is that it fails to grasp common characteristics of multiple word senses. The discrimination network, on the other hand, can represent similar word senses as close nodes in the network and less similar word senses as farther nodes. Therefore, the downward traversal of the network corresponds to the continuous refinement of an ambiguous word meaning into a more specific one[Jacobs, 1988].
- a set of multiple word senses can be represented by a list, where each item in the list is a word sense. The algorithm for selecting among them – a linear search of the list – is very inefficient when candidate word senses are too numerous. It takes time $O(n)$, where n is the number of candidate word senses, namely the length of the list. On the other hand, the discrimination network's search algorithm is considered more efficient because the downward traversal from

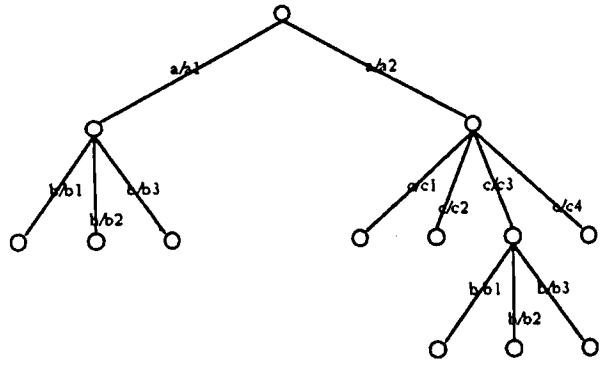

Figure 2: A sample discrimination network

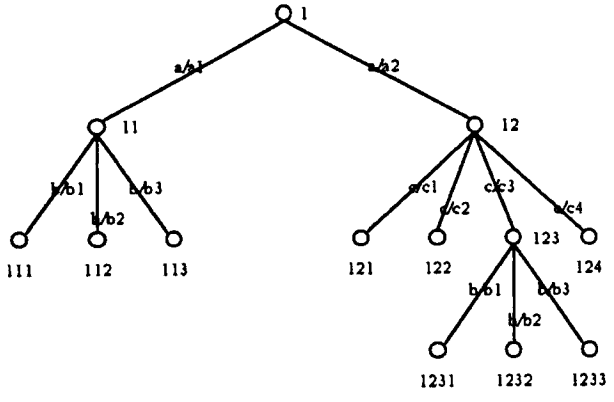

Figure 3: Discrimination network with identifier-assigned nodes

1	0
11,12	00
111,112,113,121,122,123,124	010
1231,1232,1233	0110

Table 1: Correspondence between node identifier and bit vector

a/a_1	11
a/a_2	12
b/b_1	$\{111, 1231\}$
b/b_2	$\{112, 1232\}$
b/b_3	$\{113, 1233\}$
c/c_1	121
c/c_2	122
c/c_3	123
c/c_4	124

Table 2: Correspondence between constraint and node identifier

tached as in Table 1. This bit vector represents the positions of the unsatisfied constraints in the network. For example, identifier 122 has the bit vector 010. Because the second bit from the left is 1, this vector indicates that constraint a/a_2, which corresponds to 12(the two digit identifier from the left of 122),[2] is unsatisfied. Similarly, in the case of identifier 1232, the attached bit vector 0110 signifies that constraints a/a_2 and c/c_3(which correspond to the two and three digit identifiers 12, 123 of 1232), are both unsatisfied because the second and third digits are 1.

Third, constraint-identifier pairs are extracted from the network in the following form: a branch and the subordinate node which is directly connected by that branch. From Figure 3, pairs in Table 2 are obtained. For the constraints of attribute name b, multiple pairs exist and so sets like $\{111, 1231\}$ correspond to them. This correspondence between constraint and identifier means that if a constraint in Table 2 is satisfied, the nodes of corresponding identifiers can be reached in the network. For example, if constraint c/c_2 is satisfied, the network can be traversed downward to the node of corresponding identifier 122.

Here, we must pay attention to the bit vector attached to the identifier. In the case of the above identifier 122, the corresponding bit vector 010 indicates that constraint a/a_2 is unsatisfied. Therefore, the reachability of node 122 is 'conditional' in that node 122 can be reached if constraint a/a_2 is satisfied. The existence of multiple pairs means that multiple corresponding nodes can be reached if a constraint is satisfied.

The regular order of constraints is a/a_2, c/c_3, b/b_1 for traversal of the network in Figure 3 downward to

[2]The correspondence between constraint and identifier is given in Table 2 and explained later.

node 1231. Here, in contrast, the case where constraints are obtained in the order of c/c_3, b/b_1, a/a_2 is considered. The discrimination process in our approach for that case is described below. We introduce a 'state' which represents a discrimination process. A state is expressed as a pair ⟨ a node identifier which can be reached, and a bit vector which represents the positions of unsatisfied constraints ⟩. The initial state(a state where no constraints are obtained) is ⟨ 1(the identifier of root node),0(the bit vector corresponding to identifier 1) ⟩. After constraint c/c_3 is obtained, the state is computed as follows, with the current state(the initial state), identifier 123 corresponding to the obtained constraint by Table 2, and bit vector 010 corresponding to the identifier 123 by Table 1:

operation between identifiers If one identifier includes the other as a prefix-numerical string, return the longer string;

operation between bit vectors After adjusting the length of bit vectors by attaching 1's to the end of the shorter vector, return the bit vector for which each bit is a conjunction of the bits of two vectors.

The operation between identifiers checks whether one node can be reached from the other in the network. As shown in Figure 3, identifiers of mutually reachable nodes in the network are in prefix-numerical string relation with each other. For example, from node 12, nodes 121, 1232 are reachable by satisfying some constraints, but it is impossible to reach node 112 from node 12 on any account. If one node is reachable from the other, the identifier of the subordinate one is returned. This operation corresponds to a downward traversal of the network by satisfying the obtained constraints.

The analysis will fail if one identifier is not a prefix of the other. For example, even if constraint c/c_2 is obtained when node 11 is reached, the network cannot be traversed any more because identifier 11 is not a prefix of identifier 122, which corresponds to constraint c/c_2.

The operation between bit vectors allows us to cope with the irregular order of the obtained constraints. The bit vector represents all the constraints that must be satisfied between the root node and the reached node. A bit of 1 means that the corresponding constraint is unsatisfied. Because no constraints are obtained, bits of the initial state bit vector are all 1 except the leftmost bit[3]. However, at the initial state, the reached node is unknown and the vector length is obscure. Therefore, the initial state bit vector is 0, and the vector length is adjusted by adding 1's to the end whenever a constraint is obtained.

Bit vectors in Table 1 have the same length as corresponding identifiers and their rightmost digit is 0. This means that the constraint which corresponds to that digit by combination of Tables 1 and 2 is satisfied. For example, identifiers 12(with bit vector 00) and 1232(with bit vector 0110) represent the satisfaction of constraints a/a_2 and b/b_2 respectively. By taking the conjunction of bits of these vectors, which represent the position of the satisfied constraint as 0, bits of the current state vector are incrementally changed to 0. If all vector bits are 0, it means that all constraints are satisfied and the network can be traversed to the reached node unconditionally. The bit conjunction operation which changes bits to 0 is executable in any order from any bit, so it is possible to cope with an arbitrary order of the obtained constraints[4].

The next state becomes ⟨ 123,010 ⟩ after constraint c/c_3 is obtained as the result of the above operations. The state bit vector shows that constraint a/a_2, which corresponds to identifier 12, is unsatisfied. Therefore, state ⟨ 123,010 ⟩ means 'the discrimination network can be traversed to node 123 if constraint a/a_2 is satisfied'.

Next, constraint b/b_1 has multiple corresponding identifier-bit vector pairs {⟨ 111,010 ⟩, ⟨ 1231,0110 ⟩}. Operations are performed on each pair with the current state. As for ⟨ 111,010 ⟩, the identifiers are not in a prefix relation, so the analysis fails. Therefore, the result is necessary only for pair ⟨ 1231,0110 ⟩. The resultant identifier is 1231 from identifiers 123, 1231. The resultant bit vector is 0100 from the conjunction of bit vectors 0101(the length-adjusted vector) and 0110. Bit vector 0100 shows that constraint a/a_2(corresponding to identifier 12) is still unsatisfied.

Finally, when constraint a/a_2 is obtained, operations are performed between ⟨ 1231,0100 ⟩ and ⟨ 12,00 ⟩. The resultant state is ⟨ 1231,0000 ⟩. Because all vector bits are 0, all constraints are satisfied and the discrimination network can be traversed to node 1231 unconditionally.

Conclusion

We have proposed a method which can traverse the discrimination network according to the order in which constraints are obtained incrementally during the analytical process. This order is independent of the a priori-fixed order of the network. This approach is considered to be an implementation of the incremental disambiguation model with a discrimination network. The operation between identifiers is regarded as the extended unification on class hierarchy[Dahlgren and McDowell, 1986, Sowa, 1984] in that the mutual reachability of two entered nodes is checked and, if successful, the resulting subordinate node is returned. Our approach is based on the notion of constraint logic programming[Dincbas, 1986]: it gives the table between constraints and corresponding identifiers, and reduces the possibilities of identifiers by unification be-

[3]The leftmost bit has no corresponding constraint and makes the vector length the same as that of the identifier.

[4]When constraints are obtained in the regular order of the network, bits of the vector are changed to 0 from left to right in turn.

tween them, and renders the search space incrementally smaller.

The problem of constraint order freedom in the discrimination network traversal has been countered in the past by transforming the network into a lattice. Our approach allows us to resolve the problem using the original network. This approach seems to have the same expressive power as a taxonomic lattice, but doesn't require as much additional memory space. We think it is also possible to apply the generalized discrimination network to the compilation of production rules[Forgy, 1982].

References

[Adriaens and Small, 1988] G. Adriaens and S.L. Small. Word expert parsing revisited in a cognitive science perspective. In S.L. Small, G.W. Cottrell, and M.K. Tanenhaus, editors, *Lexical Ambiguity Resolution : Perspectives from Psycholinguitics, Neuropsychology, and Artificial Intelligence*, pages 13–43. Morgan Kaufmann Publishers, 1988.

[Aho *et al.*, 1983] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.

[Bobrow and Webber, 1980] R.J. Bobrow and B.L. Webber. Knowledge representation for syntactic/semantic processing. In *Proc. of the 1st National Conference on Artificial Intelligence*, pages 316–323, 1980.

[Charniak *et al.*, 1980] E. Charniak, C.K. Riesbeck, and D.V. McDermott. *Artificial Intelligence Programming*. Lawrence Erlbaum Associates, 1980.

[Dahlgren and McDowell, 1986] K. Dahlgren and J. McDowell. Kind types in knowledge representation. In *Proc. of the 11th International Conference on Computational Linguistics*, pages 216–221, 1986.

[Dincbas, 1986] M. Dincbas. Constraints, logic programming and deductive databases. In *Proc. of the France-Japan Artificial Intelligence and Computer Science Symposium 86*, pages 1–27, 1986.

[Forgy, 1982] C.L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1):17–37, 1982.

[Hirst, 1987] G. Hirst. *Semantic interpretation and the resolution of ambiguity*. Cambridge University Press, 1987.

[Jacobs, 1988] P.S. Jacobs. Concretion: Assumption-based understanding. In *Proc. of the 12th International Conference on Computational Linguistics*, pages 270–274, 1988.

[Lytinen, 1988] S.L. Lytinen. Are vague words ambiguous? In S.L. Small, G.W. Cottrell, and M.K. Tanenhaus, editors, *Lexical Ambiguity Resolution : Perspectives from Psycholinguitics, Neuropsychology, and Artificial Intelligence*, pages 109–128. Morgan Kaufmann Publishers, 1988.

[Mellish, 1985] C.S. Mellish. *Computer Interpretation of Natural Language Descriptions*. Ellis Horwood, 1985.

[Moerdler and McKeown, 1988] G.D. Moerdler and K.R. McKeown. Beyond semantic ambiguity. In *Proc. of the 7th National Conference on Artificial Intelligence*, pages 751–755, 1988.

[Sowa, 1984] J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, 1984.

[Woods, 1978] W.A. Woods. Taxonomic lattice structures for situation recognition. In *Theoretical Issues in Natural Language Processing 2*, pages 33–41, 1978.

[Matsumoto *et al.*, 1983] Y. Matsumoto, et.al. BUP:a bottom-up parser embedded in Prolog. *New Generation Computing*, 1(2):145–158, 1983.