An Implementation of Incremental Disambiguation with Conceptual Hierarchy

Thanaruk Theeramunkong Manabu Okumura
Hozumi Tanaka

Department of Computer Science,
Tokyo Institute of Technology
2-12-1, O-okayama, Meguro-ku, Tokyo 152, Japan
Tel. (03)726-1111 (Ext. 4175)
E-mail {ping,dora,tanaka}@cs.titech.ac.jp

Abstract

Word sense ambiguity is one of the most important problems of natural language semantic interpretation. In traditional method, there is a combinatorial explosion problem due to the fact that the propriety of each word sense of an ambiguous word is independently checked with each word sense of other ambiguous words during the disambiguation process. In this paper, we describe an 'incremental semantic disambiguation' model which uses two representations called 'generalized discrimination network'[Okumura *et al.*, 1990] and 'bit-vector-formed conceptual hierarchy representation'. By these representations, word senses of an ambiguous word are represented in the expression that can be used immediately in disambiguation process by not individualizing these word senses. Consequently, the complexity in semantic disambiguation process is improved, independent of the number of word senses of a word. Operations for the disambiguation process in our model are simply equivalent to OR operation. Finally, we introduce the tree-like representation to compact the bit vector style representation. Then, the reduction rate of space using the compaction is evaluated.

1 Introduction

The meaning of a word is ambiguous because it cannot be uniquely determined unless the information of other parts in the sentence is obtained. There are three roughly-classified classes of word sense ambiguities that occur in natural language. We define words that have many concepts, called *polysemy*, to be the first kind of them. It includes words such as bank (river bank, money bank, sequence bank). The second one is called *homonymy*. This includes words that have various meanings (views) but the same superficial word (such as, children (for one's own children or general children)). The last one is the kind of *underspecific* ambiguity where some kinds of information (more details) are not defined, such as, 'human' has more general meaning than 'man' and will be restricted to 'man' when there is some information showing that 'human' is male.

There are currently incremental style and non-incremental style for resolving word sense ambiguity. In non-incremental style, the meanings of words in the sentence are determined at the end of the sentence. However, when the sentence is long, this strategy is impracticable because it will encounter a combinatorial explosion. In incremental style, towards the first and second ambiguity, [Hirst, 1987],[Hirst and Words, 1988] had suggested 'Polaroid words' that represent all word senses in list-like representation. Then when some information is obtained, all word senses in the list will be examined and the improper ones will be eliminated. This method is the incremental disambiguation process, however there are enormous checks in the process of ambiguity resolution. For example, between two words, the number of checks is the product of the number of word senses of these two words and between three words, the number of checks in the worst case is the product of

the number of word senses of these three words, and so on. Therefore, this method is inefficient because of the large computational complexity. Towards the third ambiguity, [Sowa, 1984] had suggested conceptual graph (that is canonical) formation rule for restricting the concept to more specific one.

Towards the above mentioned problem, we propose our computational model based on "incremental disambiguation" [Mellish, 1985] where the process is considered to be the immediate refinement of ambiguous results of semantic processing by newly obtained constraints. In our model, two representations called generalized discrimination network (GDN for short) [Okumura *et al.*, 1990] and bit vector representation that characterizes conceptual hierarchy (CH-representing bit vector for short) are used to the semantic disambiguation process more efficient.

In the generalized discrimination network, which is the extension of a discrimination network to deal with an a priori-fixed order problem[Okumura *et al.*, 1990], semantic disambiguation process can be equated with the downward traversal of the network. Indeterminates which represent word sense ambiguities of words were introduced in this model. When some more information is obtained, the meanings of word will be incrementally determined. The indeterminates are not the list-like representation by enumeration, but are represented as the states in the generalized discrimination network. The merit of this network is that instead of selection through a set of multiple candidate word senses (a linear search) that takes time $O(n)$ (n is the number of candidate word senses), the discrimination network's search algorithm (downward traversal in the network) takes time $O(l)$, where l is the height of the tree, roughly equivalent to $\log n$. The generalized discrimination network is used to represent word sense ambiguity for 'verb' in our model. Here, the selectional restrictions on surface cases, such as subject(S), object(O), prepositions like 'with', 'to' and so on, are used as constraints on the arcs of the network.

We introduce bit vector expression that is sufficient enough to characterize concepts (word senses), especially noun, in conceptual hierarchy. In general, word sense ambiguities of a word, *polysemy*, *homonymy* and *underspecific ambiguity*, are represented by a bit vector in our model. By this bit vector, this ambiguous word (bit vector) can be resolved to be less ambiguous by using the AND operator. We will describe semantic disambiguation in detail in next section. To be mentioned later, the number of bits in a bit vector is equivalent to the number of concepts in a conceptual hierarchy. Therefore, when the number of concepts is large, the long bit vector is not appropriate to be implemented. We introduce the tree-like representation to compact the bit vector. Then, we evaluate the reduction rate of space after the compaction to tree-like representation.

Using generalized discrimination network for 'verb' and conceptual hierarchy for 'noun', it is not necessary to check each of the word senses of 'verb' with each of the word senses of 'noun' but only to check satisfaction between the GDN of 'verb' and the bit-vector-formed representation of 'noun'. The complexity becomes independent of the number of word senses but linearly relative to the number of words.

2 Word sense Ambiguity in Conceptual Hierarchy

In this paper, we use conceptual hierarchy based on the superordinate/subordinate relation to represent ambiguities (especially of noun) that occur in natural language. Conceptual hierarchy refered in this paper is the isa relation of 'noun' where relations between concepts are defined in the ralated way. There are also many researches concerned conceptual hierarchy, especially for noun, such as [Tanaka and Nishina, 1987], [Ishizaki *et al.*, 1987], [Nirenburg and Raskin, 1987], [Isahara and Ishizaki, 1990]. However, in those researches, the way to tackle word sense ambiguity in the semantic disambiguation process is not described. In this section, we show lexical ambiguities expressed in conceptual hierarchy and we will show how it can be used in semantic disambiguation in the section 4.

There are many sources of word sense ambiguities in natural language. We roughly classified them into three kinds of ambiguities. The part of conceptual hierarchy has been shown in figure 1. *Homonymy* refers to words whose various definitions are unrelated, such as 'bank' has at least three meanings ('river bank', 'money bank' and 'sequence bank'). *Polysemy* refers to words whose several meanings are related but in different views[Tokunaga *et al.*, 1989]. For example, there are several views of 'money bank', such as 'place', 'building', 'organization', and so on. *Underspecific ambiguity* refers to words which will be restricted to the more specific ones when some information is obtained, eg. 'place' is restricted to 'river bank' when information like 'a place near river' is acquired. All of these kinds of ambiguities are repre-

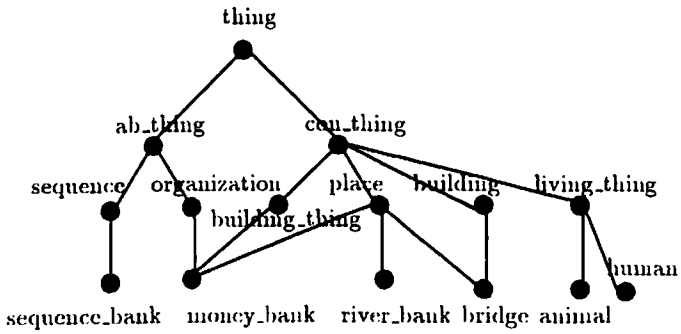

Figure 1: A part of conceptual hierarchy

sented in CH. For example, word 'bank''s ambiguities are shown in figure 2

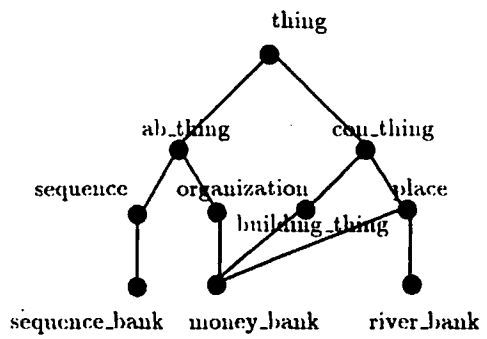

Figure 2: Word Sense Ambiguity

3 Generalized Discrimination Networks

In this section, we describe an incremental disambiguation model called 'generalized discrimination network' that is proposed in [Okumura *et al.*, 1990]. This model is used for representing 'verb' ambiguities. The incremental disambiguation process is considered to be the refinement of ambiguous (undetermined) results of semantic processing by newly obtained constraints. The semantic disambiguation process can be equated with the downward traversal of a discrimination network [Charniak *et al.*, 1980]. However, for a discrimination network, it cannot be traversed unless constraints are entered in an a priori-fixed order. Towards this point, the generalized discrimination network is proposed in [Okumura *et al.*, 1990]. This method 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.

In [Okumura *et al.*, 1990], generalized discrimination network is used to represent semantic ambiguity of 'verb' where the newly obtained constraints are the selectional restrictions on surface cases such as subject(S), object(O), prepositions like 'with', 'to' and so on. The GDN for the verb 'check' is shown in figure 3.

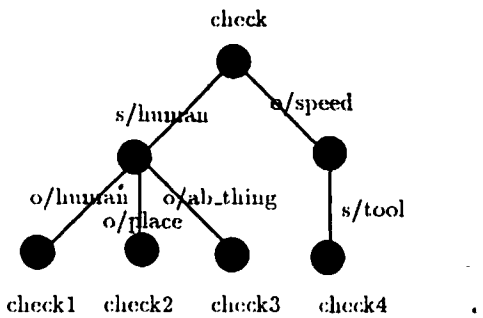

Figure 3: An example of the generalized discrimination network of word 'check'

4 Semantic Disambiguation with CH and GDN

'Verb' ambiguity resolution information is represented in the form of GDN while 'noun' information is represented in the form of graph (a part of conceptual hierarchy). Selectional restrictions that are the constraints (arcs) in GDN are also represented in the form of a part of conceptual hierarchy. The process of semantic disambiguation with CH and GDN occurs by unifying constraints in GDN and input 'noun' occurrence to the result graph. If the result graph is null, the constraint is not satisfied, otherwise the constraint is satisfied and 'noun' ambiguities are partially resolved to be less ambiguous. The following example elaborates the unification between graphs.

Suppose that 'I check the bank.' is currently analyzed. Here, 'check' and 'bank' are ambiguous. The case of 'bank' in this sentence is 'object'. 'Object' constraints in GDN of 'check' are 'human', 'abstract thing' and 'speed'. Unification between these three constaints and 'bank' acts as the disambiguation process. When unification of a constraint with 'bank' produces a result graph that is not null, this constraint satisfies restriction in GDN and word 'bank' is resolved to be less ambiguous. Figure 4 shows the unification between 'abstract thing' and 'bank' and the result of the unification (less ambiguous 'bank').

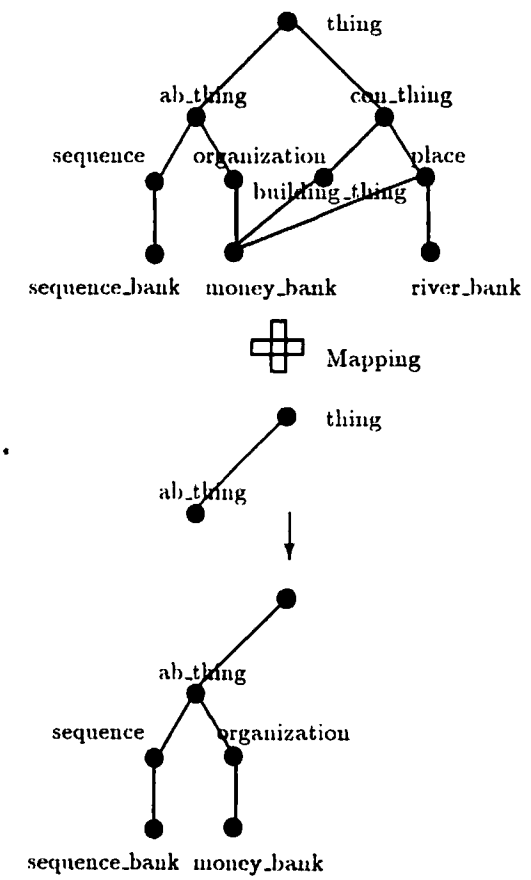

Figure 4: Graph unification as ambiguity resolution

As described, in the traditional method, the meanings of words in the sentence are determined by checking all of the possible combinations of word senses. However, when the number of words that are used to determine word sense of the word, are large, it causes combinatorial explosion. Semantic disambiguation process of our model, can be viewed as the transaction between two word representations, GDN and CH. Since all of ambiguities are not individualized but integrated into one expression, the complexity of our model is independent of the number of word senses in words but linearly relative to the number of words. That is, the complexity is $O(N)$ when N is the number of words. This shows that our model has less complexity than the previous method in which complexity is also dependent to the number of word senses in words.

5 Bit Vector Representation and Operations

In this section, we describe the way to represent CH by bit vector[1] that can characterize CH graph. Ambiguous word can be represented by ORing the bit vector of a set of concepts that the word means. The unification between ambiguous words as semantic disambiguation corresponds to AND operation between the bit vector of those words.

5.1 Linearly ordered relation

Conceptual hierarchy is a set of upper/lower relation between any two nodes. This relation is partial order $\leq$ relation. To assign a bit vector to any concept, we find $<$ relation called linear order of CH by supposing that if $a \leq b$ then $a < b$. The algorithm to find linear order is called **topological sorting algorithm** [Kolman and C.Busby, 1987]. For instance, topological sorting of CH in the figure 1 is shown in figure 6. After sorting linear order of CH, we assign a bit vector of a concept by using the algorithm in figure 5 and then the result is shown in figure 6.

Concept's flag and concept's identifier are used to represent any one concept in CH. Concept's flag points the bit that represents that concept, as concept's identifier shows the relation between that concept and other concepts. Note that if concept a has upper/lower relation with concept b, the bit representing concept b in identifier of concept a is 1, otherwise 0.

5.2 Representation of word sense ambiguities

Let W_1 be an ambiguous word that has two possible concepts, C_1, C_2. Let $BI(C_1)$ and $BI(C_2)$ be

[1] series of 0 and 1

Marking Identifier Algorithm :

1. Using the topological sorted order to mark the unique flag to any one concept, mark the smallest flag to the top concept and next bigger flag to the next concept and so on as shown in figure 6

2. Find concept identifier by ORing that concept's flag with its **subordinate** concepts' flags and its **precedence** concepts' flags. The example identifiers are shown in figure 6.

Figure 5: Marking identifier algorithm

No.	concepts	flag	identifier
1	thing	1	111111111111111
2	con_thing	10	1001111111111
3	living_thing	100	11111
4	human	1000	1111
5	animal	10000	10111
6	building_thing	100000	1000101100011
7	building	1000000	1100011
8	place	10000000	1001110000011
9	bridge	100000000	110100011
10	river_bank	1000000000	1010100011
11	ab_thing	10000000000	111110000000001
12	organization	100000000000	1110000000001
13	money_bank	1000000000000	1110010100011
14	sequence	10000000000000	110010000000001
15	sequence_bank	100000000000000	110010000000001

Figure 6: An Augmented Topological Sorting

the bit vector identifiers of C_1 and C_2 respectively. Let $BF(C_1)$ and $BF(C_2)$ be the bit vector flags of C_1 and C_2 that are derived by the algorithm shown in figure 5, respectively. The bit vector identifier and flag of word , WI_1 and WF_1 are calculated by ORing $BI(C_1)$ with $BI(C_2)$. $BI(C_1) \vee BI(C_2)$ and ORing $BF(C_1)$ with $BF(C_2)$. $BF(C_1) \vee BF(C_2)$, respectively. For example, 'bank' in figure 2 has three concepts (money_bank, river_bank, sequence_bank). So the identifier and flag of the word 'bank' can be calculated by ORing the identifiers and flags of money_bank, river_bank and sequence_bank. Here, the identifier and the flag is '111111010100011' and '101001000000000', respectively.

5.3 Operations on concept bit vectors for Semantic Disambiguation

In this subsection, semantic disambiguation with bit vector representation is decribed. Suppose that there is a 'verb' generalized discrimination network G and a concept, C, is being checked; satisfaction with constraints in the GDN. The disambiguation algorithm is shown in the figure 7

Disambiguation Algorithm :

1. Choose a constraint, Con, from generalized discrimination network G.

2. Suppose the constraint identifier of Con is $BI(Con)$ and the identifier and flag of the concept C are $BI(C)$ and $BF(C)$, respectively. Make ANDing $BI(Con)$ and $BI(C)$ and ANDing $BI(Con)$ and $BF(C)$ and let the result identifier and flag be $BRI(Con, C)$ and $BRF(Con, C)$, respectively.

3. If $BRF(Con, C)$ is 0, that means constraint satisfaction fails, then goto next step otherwise the concept C satisfies the constraint and the concept ambiguity is resolved to be $BRI(Con, C)$.

4. repeat 1, until no constraint left in G.

Figure 7: Disambiguation algorithm

For instance, the analyzed sentence is 'I check the bank'. One of the 'object' constraints of 'check' is 'place'. After ANDing the identifiers and flags of the two concepts, 'place' and 'bank' in figure 6, the result identifier obtained by ANDing '1001110000011' and '111111010100011' is '001001010000011'. Likewise, the result flag obtained by ANDing '1001110000011' and '101001000000000' is '001001000000000'. Here, because the result flag is not 0, the word 'bank' that is ambiguous satisfies this constraint 'place' and can be bounded to be less ambiguous. Note that 14th bit that represents 'bank3' disappears. That means that 'bank3' does not satisfy the constraint 'place'.

6 Tree-like Bit Vector Representation

In the previous subsection, we described about how to assign bit vector to represent a concept in CH. Here, the number of bits in one concept's identifier is the number of concepts in CH. In general, the number of concepts in CH is huge, so it's not efficiency to use this common bit vector representation, because it will be such a long vector. Here, we introduce the idea of tree representation by converting the long bit vector to the tree-like representation by

the algorithm that is shown in figure 8.

Converting to Tree-like Representation

1. Divide long bit vector into 8-bit groups from right to left and let it be 0th level. (n = 0)
2. For each group from right to left do
   If there are all 0's in a group, then compact them to '0' in the n+1th level else compact them to '1' in the n+1th level. See example in figure 9.
3. From (3) we obtain the n+1th level bit vector. if the number of bits in n+1th level are less than 8 then stop. else Divide the n+1th level bit vector into 8-bit groups from right to left and let n = n+1 and then goto 2

Figure 8: Bit vector to Tree-like representation Algorithm

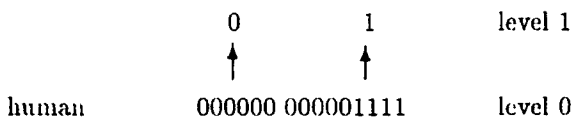

Figure 9: Tree-like representation of 'human'

In figure 9. the tree-like representation of 'human' is shown. Since there aren't many concepts in CH. there are only two levels in this case. However, when the number of concepts in CH is large, there will be many levels in tree-like representation. For example. figure 10 shows 3 levels of tree-like representation. In tree-like representation. only non-zero 8-bit vectors are accumulated. A zero 8-bit vector is represented by one 0 in the higher level. such as in figure 10. 0 in level 1 represents a zero 8-bit vector in level 0, etc. . Therefore, the space used in this representation decreases. Here. we use only 7 bytes (56 bits) to represent this concept, in the other hand. 8^2 bytes (8^3 bits = 512 bits) are necessary used to represent for common bit vector representation case. Here, the more there are 0's in the bit vector, the less number of bits are used. In the general case. almost all bit in identifier bit vector are 0's.

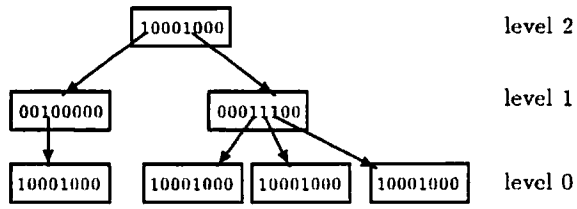

Figure 10: An example of Tree-like representation

7 Operation on Tree-like Representation

The operation in the tree-like representation is similar to the bit vector representation but instead of one AND operation, the AND operation takes place in each level of tree-like structure from top level to the lower levels.

The number of calculation that occurs between concepts depend on those concept identifiers. However. in the worst case. when all of bits in the level 1 are 1, there is only 14.3 % more calculation than by long bit vector representation. In the best case. there are only $\log_8 N$ operations necessary. where N is the number of concepts in CH (the number of bits in bit vector).

8 Evaluation of Tree-like Representation

We have coded concepts in conceptual hierarchy that consists of 69938 concepts into tree-like bit vector style. There are 69938 bits used for a concept in experimental CH in the common bit vector representation. Tree-like representation makes it realistic and economic to represent concepts. The space used in tree-like representation of experimental CH is shown in figure 11.

The maximum space and minimum space that are used for representing concepts in experimental CH are shown in figure 11. The sum of space for representing all concepts in experimental CH and the average space for representing a concept in experimental CH are also shown in figure 11.

There are just only about 20 bytes (160 bits) used for one concept's identifier to be represented and the depth of this tree-like representation is 6. Using tree-like representation when 0's are in the neighborhood

Maximum	9995	bytes.
Minimum	6	bytes.
Sum	1380056	bytes.
Average	19.7326	bytes.
no.of.concepts	69938	concepts.

Figure 11: space used after compacting bit vector to tree-like representation

positions we can represent by only one '0' in the higher level, so when there are many 0's in one vector bit, it looks efficient to use this representation. The reduction of this conceptual hierarchy indicates that there are actually many 0's existing in neighborhood position in the concept identifier in experimental CH.

The reduction can be calculated by the ratio between the number of bits that used in tree-like representation and the number of bits that used in common bit vector. From figure 11, in the worst case, our system uses 9995 × 8 (79960) bits to represent a concept. In this case, the tree-like representation uses space 14.33 % more than the common bit vector representation does. But as average, the space used in our system is about 0.23 % of that of bit vector representation.

9 Conclusion

Our model uses two representation. GDN is used for 'verb' representation. Bit-vector-formed representation is used for 'noun' representation. Word senses of an ambiguous word are represented in the expression that can be used immediately in disambiguation process by not individualizing these word senses. Semantic disambiguation is equated to AND operation between the bit vector of constraints in GDN and the bit vector of ambiguous words (noun) in CH. The complexity of our model is independent of the number of word senses of words but linearly relative to the number of words. Lastly, we introduce tree-like representation to represent concepts in CH. The tree-like representation makes it possible to compact the bit vector and use much less space than the bit vector representation. Finally, We checked the efficiency of this representation and found that the used space was improved.

10 Acknowledgements

I would like to thank to Takenobu Tokunaga and all members in Tanaka laboratory who gave many helpful comments and Amarit Laorakpong, Surapan Meknavin, Suresh Katare Goralrao and Craig Patrick Hunter who helped with checking this paper.

References

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

[Hirst and Words, 1988] G. Resolving Lexical Ambiguity Computationally with Spreading Activation Hirst and Polaroid Words. *Lexical Amiguity Resolution*. Morgan Kaufmann Publishers, 1988.

[Hirst, 1987] G. Hirst. *Semantic Interpretation and the Resolution of Ambiguity. Studies in Natural Language Processing*. Cambridge University Press, 1987.

[Isahara and Ishizaki, 1990]
H. Isahara and H. Ishizaki. Natural language understanding system with concept hierarchy. In *PRICAI*. 1990.

[Ishizaki *et al.*, 1987]
H. Ishizaki, H. Isahara, K. Hashida, K. Uchida, and S Yokoyama. A method of concept description for contexual analysis. *Japan Information Processing Society. SIG Reports*, 87-NL-64, 1987. (in Japanese).

[Kolman and C.Busby, 1987] Bernard Kolman and Robert C.Busby. *Discrete Mathematical Structures For Computer Science*. Prentice-Hall.Inc., 2 edition, 1987.

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

[Nirenburg and Raskin, 1987]
S. Nirenburg and V. Raskin. The subworld concept lexicon and the lexicon management system. *Computational Linguistics*, 13(3-4):276-289, 1987.

[Okumura *et al.*, 1990] M. Okumura, M. Surapan, and H. Tanaka. Towards incremental disambiguation with a generalized discrimination network. In *AAAI*, 8 1990.