Decision Tree Learning Algorithm with Structured Attributes: Application to Verbal Case Frame Acquisition

Hideki Tanaka
NHK Science and Technical Research Laboratories
1-10-11 Kinuta, Setagaya-ku
Tokyo, 157, Japan
tanakah@strl.nhk.or.jp

Abstract

The Decision Tree Learning Algorithms (DTLAs) are getting keen attention from the natural language processing research community, and there have been a series of attempts to apply them to verbal case frame acquisition. However, a DTLA cannot handle structured attributes like nouns, which are classified under a thesaurus. In this paper, we present a new DTLA that can rationally handle the structured attributes. In the process of tree generation, the algorithm generalizes each attribute optimally using a given thesaurus. We apply this algorithm to a bilingual corpus and show that it successfully learned a generalized decision tree for classifying the verb "take" and that the tree was smaller with more prediction power on the open data than the tree learned by the conventional DTLA.

1 Introduction

The group of Decision Tree Learning Algorithms (DTLAs) like CART (Breiman et al., 1984), ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) are some of the most widely used algorithms for learning the rules for expert systems and has been successfully applied to several areas so far.

These algorithms are now getting keen attention from the natural language processing (NLP) research community since the huge text corpus is becoming widely available. The most popular touchstone for the DTLA in this community is the verbal case frame or the translation rules. There have already been some attempts, like (Tanaka, 1994) and (Almuallim et al., 1994).

The group of DTLAs, however, was originally designed to handle "plain" data, whereas nouns are "structured" under a thesaurus. Although handling such "structured attributes" in the DTLA was described as a "desirable extension" in the book of Quinlan (Quinlan, 1993), the problem has received rather limited attention so far (Almuallim et al., 1995).

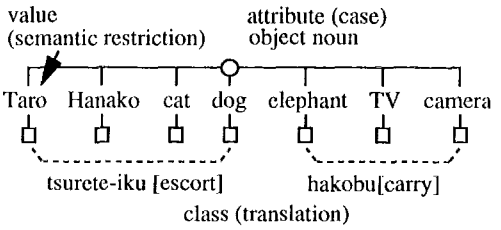

Figure 1: Case Frame Tree Learned by DTLA

There have been several attempts to solve the problem in the NLP community, such as (Tanaka, 1995b), (Almuallim et al., 1995). These attempts, however, are not always satisfactory in that the handling of the thesaurus is not flexible enough.

In this paper, we introduce an extended DTLA, LASA–1 (inductive earning Algorithm with Structured Attributes), which can handle structured attributes in an optimum way. We first present an algorithm called T*, which can solve the sub-problem for structured attributes and then present the whole algorithm of LASA–1. Finally, we report an application of our new algorithm to verbal case frame acquisition and show its effectiveness.

2 The Structured Attribute Problem

Figure 1 shows an example decision tree representing a case frame for the verb "take." This decision tree was called the case frame tree (Tanaka, 1994) and we follow that convention in this paper, too. One may recognize that the restrictions in figure 1 are not semantic categories but are words: this tree was learned from table 1 which contains word forms for the values. Although the tree has some attractive features mentioned in (Tanaka, 1994), it suffers from two problems.

- weak prediction power
  A case frame tree with word forms does not have high prediction power on the open data

Table 1: Single Attribute Table for "take"

ON: Object Noun	Japanese Translation
Taro	tsurete-iku (escort)
Hanako	tsurete-iku (escort)
cat	tsurete-iku (escort)
dog	tsurete-iku (escort)
elephant	hakobu (carry)
elephant	hakobu (carry)
TV	hakobu (carry)
camera	hakobu (carry)

(the data not used for learning). The nouns are the most problematic. There will be many unknown nouns in the open data.

- low legibility
  If we include many different nouns in the training data (the data used for learning), the obtained tree will have as many branches as the number of nouns. The ramified tree is hard for humans to understand.

Introducing a thesaurus or a semantic hierarchy in a case frame tree seems a sound way to ameliorate these two problems. We can replace the similar nouns in a case frame tree by a proper semantic class, which will reduce the size of the tree while increasing the prediction power on the open data. But how can we introduce a thesaurus into the conventional DTLA framework? This is exactly the "structured attributes" problem that we mentioned in section 1.

3 The Problem Setting

3.1 Partial Thesaurus

The DTLA takes an attribute, value and class table for an input [1]. Although the table usually includes multiple attributes, the algorithm evaluates an attribute's goodness as a classifier independently of the rest of the attributes. In other words a "single attribute table" as shown in table 1 is the fundamental unit for the DTLA. This table shows an imaginary relationship between an object noun of the verb "take" and the Japanese translation. We used this table to learn the case frame tree in figure 1 and it suffered from the two problems.

Here, we can assume that the word forms of the ON are in a thesaurus (We call this thesaurus the original thesaurus) and we can extract the relevant part as in figure 2. We call this tree a partial thesaurus T [2]. If we replace "Taro" and "Hanako" in table 1 by "*human" in T, for example, and assign the translation "tsurete-iku" to "*human," the learned case frame tree will reduce the size by one (two leaves in figure 1 are replaced by one leaf). If we replace "Taro," "Hanako," "cat," "dog" and "elephant" by "*mammal," and assign the translation "tsurete- iku" to "*mammal" (The majority translation under the node "*mammal" in T. We are going to use this "majority rule" for the class assignment.), then the learned case frame tree will reduce the size by four. But the case frame tree will produce two translation errors ("hakobu" for "elephant") when we classify the original table 1. In both cases, the learned case frame trees are expected to have reinforced prediction power on the open data thanks to the semantic classes: the replacement in the table generalizes the case frame tree. We want high-level generalization but low-level translation errors; but how do we achieve this in an optimum way?

[1] We are going to mainly use the terms attribute, value, and class for generality. They actually refer to the case, restrictions for the case, and the translation of the verb respectively in our application. In this paper, we use these terms interchangeably.

[2] The scores at each node will be explained in section 3.3.

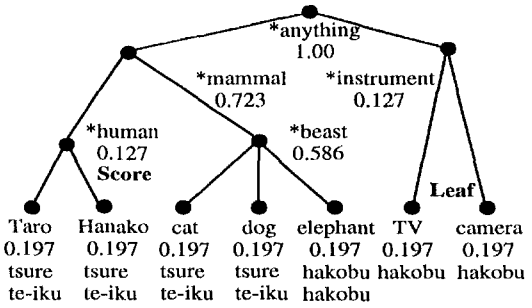

Figure 2: Partial Thesaurus T

Table 2: Notations 1

T	partial thesaurus
r	root node of T
p	any node in T, take subscripts i, j
P	any node set in T
N	set of all nodes in T
$\hat{L}(p)$	set of leaf under p

3.2 Unique and Complete Cover Generalization

One factor we have to consider is the possible combinations of the node set in T which we use for the generalization of the single attribute table. In this paper, we allow to use the node sets which cover the word forms in the table uniquely and completely. These two requirements are formally defined below using the notations in table 2.

Definition 1: For a given node set $P \subset N$, P is called the unique cover node set if $\hat{L}(p_i) \cap \hat{L}(p_j) = \phi$ for $\forall p_i, p_j \in P$ and $i \neq j$.

Definition 2: For a given node set $P \subset N$, P is called the complete cover node set if $\bigcup_{p_i \in P} \hat{L}(p_i) = \hat{L}(r)$.

Table 3: Notations 2

M	total word count in thesaurus
p'	thesaurus node corresponding to p
$D(p')$	word count under p'
$\hat{C}(p)$	set of class under p
c_i	class
$f(c_i)$	frequency of c_i
A	set of class
$\|A\|$	$\sum_{c_i \in A} f(c_i)$
$H(A)$	entropy of class distribution in A

The node set that satisfy the two definitions is called the *unique and complete cover* (UCC) node set and each such node set is denoted by P_{ucc}. The set of all UCC node set is denoted by $\mathcal{P}$. It should be noted that if we use only the leaves in T for generalization, there will be no actual change in the table and this node set is included in $\mathcal{P}$.

The total number of UCC node sets in a tree is generally high. For example, the number of UCC node set in a 10-ary tree with the depth of 3 is about 1.28×10^{30}. We will consider this problem in section 4.

3.3 Goodness of Generalization

Another factor to consider is the measurement of the goodness of a generalization. To evaluate this quantitatively, we assign a penalty score $S(p)$ to each node p in T as

$$S(p) = \alpha \cdot G_{pen}(p) + E(p), \qquad (1)$$

where α is a coefficient, $G_{pen}(p)$ is the penalty for generality[3] , and $E(p)$ is a penalty for the induced errors by using p.

The node that has small $S(p)$ is preferable. And $G_{pen}(p)$ and $E(p)$ are generally mutually conflicting: high generality node p (with low $G_{pen}(p)$) will induce many errors resulting in high $E(p)$ and vice versa. We measure a generalization's goodness by the total sum of the penalty scores of the nodes used for the generalization. There are several possible candidates for the penalty score function and we chose the formula (2) for this research.

$$S(p) = -\alpha \cdot \log \frac{D(p')}{M} + \frac{|\hat{C}(p)|}{|\hat{C}(r)|} H(\hat{C}(p)) \qquad (2)$$

New notations are listed in table 3 in addition to table 2. The second term in formula (2) is the "weighted entropy" of the class distribution under node p, which coincides Quinlan's criterion (Quinlan, 1993).

We calculated $G_{pen}(p)$ (the first term of formula (2)) based on the word number coverage of p' in the original thesaurus rather than in the partial thesaurus, since the original thesaurus usually contains many more words than the partial thesaurus, and is thus expected to yield a better estimate on the generality of node p. The idea is shown in figure 3. The coefficient α is rather difficult to handle and we will touch on this issue in section 4.3. The figures attached to each node in figure 2 are the example penalty scores given by formula (2) under the assumption that the T and the original thesaurus are the same and $\alpha = 0.007$[4].

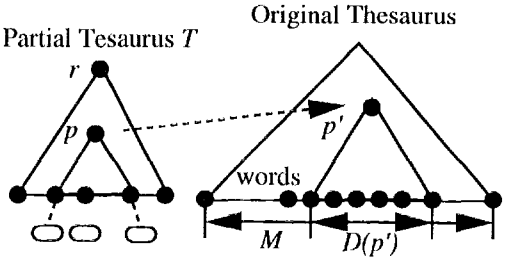

Figure 3: Generality Calculation

With these preparations, we now formally address the problem of the optimum generalization of the single attribute table.

The Optimum Attribute Generalization

Given a tree whose nodes each have a score: Find P_{ucc} that has the minimal total sum of scores:

$$\arg \min_{P_{ucc} \in \mathcal{P}} \sum_{p_i \in P_{ucc}} S(p_i) \qquad (3)$$

4 The Algorithms

4.1 The Algorithm T*

As was mentioned in section 3, the number of UCC node set in a tree tends to be gigantic, and we should obviously avoid an exhaustive search to find the optimum generalization. To do this search efficiently, we propose a new algorithm, T*. The essence of T* lies in the conversion of the partial thesaurus: from a tree T into a directed acyclic graph (DAG) $\mathcal{T}$. This makes the problem into "the shortest path problem in a graph," to which we can apply several efficient algorithms. We use the new notations in table 4 in addition to those in table 2.

The Algorithm T*

```
Tstar(value, class){
  extract partial thesaurus T with
    value and class;
/* conversion of T into a DAG T */
  assign index numbers (1,...,m)
    to leaves in T from the left;
  add start node s to T with
    index number 0
    and e with index number m+1;
  foreach( n ∈ N ∪ {s}){
    extend an arc from n to each
```

[3] If p has low generality, it will have high $G_{pen}(p)$.

[4] This coefficient was fixed experimentally.

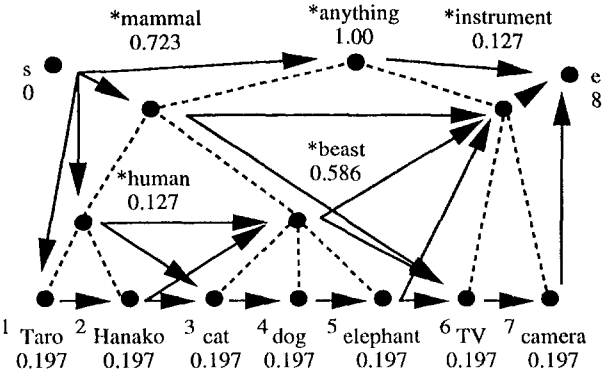

Figure 4: Traversal Graph $\mathcal{T}$

Table 4: Notations 3

$L_{min}(p)$	leaf with smallest index in $\hat{L}(p)$
$L_{max}(p)$	leaf with biggest index in $\hat{L}(p)$

element in the set H_n
defined by (4);}
delete original edges appeared in T;
/* search for shortest path in $\mathcal{T}$ */
opt_node_set = find_short($\mathcal{T}$);
return *opt_node_set*;
}

$$H_n = \{x \mid x \in N \cup \{e\}, \; L_{min}(x) - 1 = L_{max}(n)\} \quad (4)$$

This algorithm first converts T in figure 2 into a DAG $\mathcal{T}$, as in figure 4. We call this graph a *traversal graph* and each path from s to e in the traversal graph a *traverse*. The set of nodes on each traverse is called a *traversal node set*.

Here we have two propositions related to the traversal graph.

Proposition 1: A traversal graph is a DAG.

Proposition 2: For any $P \subset N$, P is a UCC node set if and only if P is a traversal node set.

Since proposition 2 holds, we can solve the optimum attribute generalization problem by finding the shortest traverse [5] in the traversal graph. By applying a shortest path algorithm (Gondran and Minoux, 1984) to figure 4, we find the shortest traverse as ($s \rightarrow$ *human $\rightarrow$ *beast $\rightarrow$ *instrument $\rightarrow e$) and get the optimally generalized table as in table 5 and the generalized decision tree as in figure 5.

4.2 Correctness and Time Complexity

We will not give a full proof for propositions 1 and 2 (correctness of T*) because of the limited space, but give an intuitive explanation of why the two propositions hold.

[5]The sum of the scores in the traversal node set is minimal.

Table 5: Optimally Generalized Single Attribute Table for "take"

ON	Translation	Freq	Error
*human	tsurete-iku	2	0
*beast	tsurete-iku	4	2
*instrument	hakobu	2	0

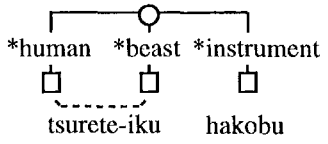

Figure 5: Optimally Generalized Decision Tree for "take"

Let's suppose that we select "*human" in figure 2 for a UCC node set P_{ucc}; then we cannot include "*mammal" in the P_{ucc}: there will be leaf overlap between the two nodes, which violates the unique cover. Meanwhile, we have to include nodes that govern L_{max}(*human) + 1, i.e. "cat," to satisfy the complete cover. In conclusion, we have to include "cat" or "*beast" in the P_{ucc}, which satisfies formula (4). The T* links all such possible nodes with arcs, and the traversal node sets can exhaust $\mathcal{P}$.

One may easily understand that the traversal graph will be a DAG, since formula (4) allows an arc between two nodes to be spanned only in the direction that increases the index number of the leaf. Since proposition 1 holds, the time complexity of the T* can be estimated by the number of arcs in a traversal graph: there is an algorithm for the shortest path problem in an acyclic graph which runs with time complexity of $O(M)$, where M is the number of arcs (Gondran and Minoux, 1984). Then we want to clarify the relationship between the number of leaves (data amount, denoted by D) and the number of arcs in the traversal graph. Unfortunately, the relationship between the two quantities varies depending on the shape of the tree (partial thesaurus), then we consider a practical case: k–ary tree with depth d (Tanaka, 1995a). In this case, the number of arcs in the traversal graph is given by

$$\frac{k(k+1)}{(k-1)^2}k^d - d^2 - \frac{2k}{k-1}d - \frac{k(k+1)}{(k-1)^2}. \quad (5)$$

Since the number of leaves D in the present thesaurus is k^d , the first term in formula (5) becomes $\frac{k(k+1)}{(k-1)^2}D$, showing that T* has $O(D)$ time complexity in this case.

Theoretically speaking, when the partial thesaurus becomes deep and has few leaves, the time complexity will become worse, but this is hardly the situation. We can say that T* has approximately linear order time complexity in practice.

4.3 The LASA–1

The essence of DTLAs lies in the recursive "search and division." It searches for the best classifier attribute in a given table. It then divides the table with values of the attribute.

The goodness of an attribute is usually measured by the following quantities (Quinlan, 1993) (The notations are in table 3.). Now let's assume that a table contains a set of class $A = \{c_1, \ldots, c_n\}$. The DTLA then evaluates the "purity" of A in terms of the entropy of the class distribution, $H(A)$.

If an attribute has m different values which divide A into m subsets as $A = \{B_1, \ldots, B_m\}$, the DTLA evaluates the "purity after division" by the "weighed sum of entropy," $WSH(attribute, A)$.

$$WSH(attribute, A) = \sum_{B_j \in A} \frac{|B_j|}{|A|} H(B_j) \quad (6)$$

The DTLA then measures the goodness of the attribute by

$$gain = H(A) - WSH(attribute, A). \quad (7)$$

With these processes in mind, we can naturally extend the DTLA to handle the structured attributes while integrating T*. The algorithm is listed below. Here we have two functions named makeTree() and Wsh(). The function makeTree() executes the recursive "search and division" and the Wsh() calculates the weighted sum of entropy. T* is integrated in Wsh() at the first "**if** clause." [6] In short, we use T* to optimally generalize the values of an attribute at each tree generation step, which makes the extension quite natural.

The LASA–1

```
place all classes in input table under
root;
makeTree(root, table);
makeTree(node, table){
  A: class set in table;
  find attribute which maximizes
    H(A) - Wsh(attribute, table);
/* table division part follows*/
}

Wsh(attribute, table){
  if(attribute is structured){
    node_set = Tstar(value, class);
    replace value with node_set;
  }
  return WSH(attribute, A) (6)
}
```

We have implemented this algorithm as a package that we called LASA- 1(inductive Learning Algorithm with Structured Attributes). This package has many parameter setting options. The most important one is for parameter α in formula (2). Since it is not easy to find the best value before a trial, we used a heuristic method. The one used in the next section was set by the following method.

We put equal emphasis on the two terms in formula (2) and fixed α so that the traverse via the root node of $\mathcal{T}$and the traverse via leaves only would have equal scores. At the beginning, LASA–1 calculated the value for each attribute in the original table.

Although this heuristics does not guarantee to output the α that has the minimum errors on open data, the value was not too far off in our experience.

5 Empirical Evaluation

5.1 Experiment

We conducted a case frame tree acquisition experiment on LASA–1 and the DTLA [7] using part of our bilingual corpus for the verb "take." We used 100 English-Japanese sentence pairs. The pairs contained 15 translations (classes) for "take," whose occurrences ranged from 5 to 9. We first converted the sentence pairs into an input table consisting of the case (attribute), English word form (value), and Japanese translation for "take" (class). We used 6 cases for attributes [8] and some of these appear in figure 6.

We used the Japanese "Ruigo–Kokugo–Jiten" (Ôno, 1985) for the thesaurus. It is a 10–ary tree with the depth of 3 or 4. The semantic class at each node of the tree was represented by 1 (top level) to 4 (lowest level) digits. To link the English word forms in the input table to the thesaurus in order to extract a partial thesaurus, we used the Japanese translations for the English word forms. When there was more than one possible semantic class for a word form, we gave all of them [9] and expanded the input table using all the semantic classes.

We evaluated both algorithms with using the 10–fold cross validation method(Quinlan, 1993).

The purity threshold for halting the tree generation was experimentally set at 75% [10] for both algorithms.

A part of a case frame tree obtained by LASA–1 is shown in figure 6. We can observe that both semantic codes and word forms are mixed at the

[6] Without this clause, the algorithm is just a conventional DTLA.

[7] Part of LASA–1 was used as the DTLA.

[8] adverb (DDh1), adverbial particle (Dh1), object noun (ONh1), preposition (PNf1), the head of the prepositional phrase (PNh1), and subject (SNh1).

[9] We basically disambiguated the word senses manually, and there were not a disastrously large number of such cases.

[10] If the total frequency of the majority translation exceeds 75% of the total translation frequency, subtree generation halts.

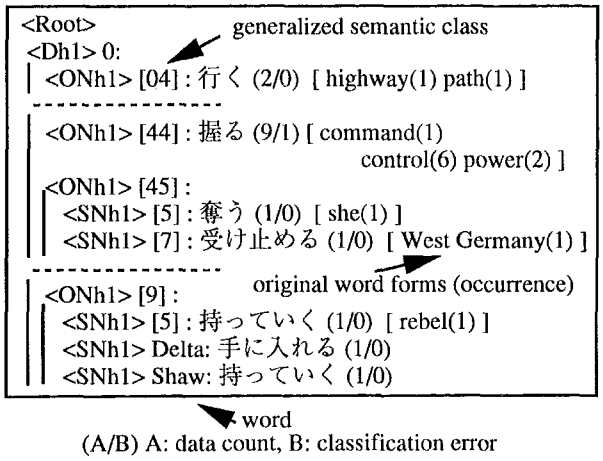

Figure 6: Case Frame Tree Learned by LASA-1

Table 6: Classification Results on Open Data(%)

	LASA(120)		DTLA(100)	
	correct	err.	correct	err.
complete	59.2	20.0	47.0	7.0
incomplete	6.7	14.2	4.0	42.0
total	65.8	34.2	51.0	49.0
leaf size	50.9		57.9	

same depth of the tree. We can also observe that semantically close words are generalized by their common semantic code.

Table 6 shows the percentage of each evaluation item. We have 120 open data, not 100, for LASA–1, because the data is expanded due to the semantic ambiguity. The term "incomplete" in the table denotes the cases where the tree retrieval stopped mid-way because of an "unknown word" in the classification. Such cases, however, could sometimes hit the correct translation since the algorithm output the most frequent translation under the stopped node as the default answer.

In table 6, we can recognize the sharp decrease in incomplete matching rate from 46.0 % (DTLA) to 20.8 % (LASA–1). The error rate also decreased from 49.0 % (DTLA) to 34.2 % (LASA–1).

The average tree size (measured by the number of leaves) for DTLA was 57.9, which dropped to 50.9 for LASA-1.

These results show that LASA–1 was able to satisfy our primary objectives: to solve the two problems mentioned in section 3, "weak prediction power" and "low legibility."

5.2 Discussion

The shape of the decision tree learned by LASA–1 is sensitive to parameter α and the purity threshold. There is no guarantee that our method is the best, so it would be better to explore for a better criterion to decide these values.

The penalty score in this research was designed so that we get the maximum generalization if the error term in formula (2) stays constant. As a result, the subtrees in the deep part are highly generalized. In those parts, the data is sparse and the high-level generalization is questionable from a linguistic viewpoint. Some elaboration in the penalty function might be required.

6 Conclusion

We have proposed a decision tree learning algorithm (inductive Learning Algorithm with the Structured Attributes: LASA–1) that optimally handles the structured attributes. We applied LASA–1 to bilingual (English and Japanese) data and showed that it successfully leaned the generalized decision tree to classify the Japanese translation for "take." The LASA–1 package still has some unmentioned features like the handling of the words unknown to the thesaurus and different α parameter setting. We would like to report those features at another opportunity after further experiments.

References

Hussein Almuallim, Yasuhiro Akiba, and Takefumi Yamazaki. 1994. Two methods for learning alt–j/e translation rules from examples and a semantic hierarchy. In *Proc. of COLING94*, volume 1, pages 57–63.

Hussein Almuallim, Yasuhiro Akiba, and Shigeo Kaneda. 1995. On handling tree–structured attributes in decision tree learning. In *Proc. of 12th International Conference on Machine Learning*, pages 12–20.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. 1984. *Classification and Regression Trees.* Chapman & Hall.

Michel Gondran and Michel Minoux. 1984. *Graphs and Algorithms.* John Wiley & Sons.

Susumu Ôno. 1985. *Ruigo-kokugo-jiten.* Kadokawa Shoten.

John Ross Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1:81–106.

John Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Hideki Tanaka. 1994. Verbal case frame acquisition from a bilingual corpus: Gradual knowledge acquisition. In *Proc. of COLING94*, volume 2, pages 727–731.

Hideki Tanaka. 1995a. A linear–time algorithm for optimal generalization of language data. Technical Report NLC-95-07, IECE.

Hideki Tanaka. 1995b. Statistical learning of "case frame tree" for translating english verbs. *Natural Language Processing*, 2(3):49–72.