연구논문집 제 10권 제 1호 p.9~15 1979

情報要素에 관한 Binary Term-Property Matrix로 부터 木構造로의 變換에 관한 Algorithm

金 熙 昇
電子計算學科

〈要 約〉

情報의 要素들이 여러가지 性質들을 가지는 度數에 따라, 情報要素들의 相互關係를 木構造로 구성하는 종래의 定量的 方法을 개선하여, 情報의 要素들이 어러가지 性質들을 가지는 有無에만 의존하여, 情報 要素들의 相互關係를 木構造로 구성하는 定性的 方法을 연구하였으며, 一般的으로 情報要素들이 가지는 性質들에 관하여 우선순위를 매길수 있는 경우에 대해서, 木構造 形式으로 표현하는 方法과 Algorithm을 제시하고, 그 타당성을 증명하였다.

An Algorithm about Hierarchical Arrangement of Information Terms based on Binary Term-Property Matrix

Kim Heui Seung
Dept of Computer Science

〈Abstract〉

Some algorithms for a hierarchical arrangement of terms from a given term-property matrix was known. The elements of such matrix indicate property frequency of the corresponding terms. While this term-property matrix is counted as quantitative, a binary term-property matrix is counted as qualitative, for the elements of this matrix is represented by 1 or 0 corresponding to the presence or absence of property. I will show an algorithm which converts a binary term-property matrix into a tree structure.

I. 서 론

Information unit들에 관한 상호관계와 성질로부터, 體系化된 구조를 밝혀 내는 한 方法으로써, Term-Property matrix가 주어지고, 이것으로부터 hierarchical structure를 구성하는 문제가 다루어져 왔다. [1]

Term-Property matrix $V=[v_{ij}]$는

	P_1	P_2	P_3 …… P_m
T_1	v_{11}	v_{12}	v_{13} …… v_{1m}
T_2	v_{21}	v_{22}	v_{23} …… v_{2m}
T_3	v_{31}	v_{32}	v_{33} …… v_{3m}
⋮	⋮	⋮	⋮
T_n	v_{n1}	v_{n2}	v_{nm}

와 같이 표시되어 information item을 T_i로, 그리고 각 T_i에 대한 property로써 P_j가 주어졌을때, information item T_i가 property P_j를 가지는 度

數가 v_{ij}이다.

이러한 Term-Property matrix로 부터 hierarchical structure로 변환시키는데 대하여는 similarity coefficient와 cut-off value 결정에 의하여 구성하는 방법이 있다. [2], [3]

Term T_i에 관한 property 系數들을 element로 하는 vector를

$$v^i=(v_{i1},\ v_{i2},\ v_{i3},\cdots\cdots v_{im})$$

라 한다면, Term T_i와 T_j의 상호상관계수 C_{ij} (similarity coefficient)는

$$C_{ij}=\frac{\sum_K \min(v_k^i, v_k^j)}{\sum v_k^i}$$

로 주어진다. 만약 두 vector v^i와 v^j가 同一하다면 C_{ij}는 1이 되고, 함께 같은 property를 가지지 않으면, C_{ij}는 0이 된다.

적당한 cut-off value K에 대하여 $C_{ij}\geq K$이고 $C_{ji}\geq K$이면, 두 term T_i와 T_j는 synonymous term의 관계이고, $C_{ij}<K$이고 $C_{ji}<K$이면, T_i와 T_j는 unrelated term의 관계이며, $C_{ij}<K$이고 $C_{ji}\geq K$이면, T_i와 T_j는 parent-son의 관계이다.

한가지 cut-off value K에 대하여, 일반적으로 term들의 parent-son의 관계가 다 밝혀지지 않고, 따라서 term들의 hierarchical structure를 구성할 수 없으므로, cut-off value K를 변동시켜 가면서 term들의 상호관계를 결정짓고, 그것으로부터 term들의 hierarchical structure를 구성한다.

예컨데 Term-Property matrix가 다음과 같을 때 즉

$V=$

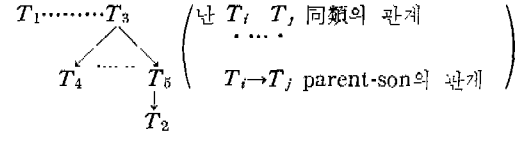
	P_1	P_2	P_3	P_4	P_5	P_6
T_1	3	0	0	5	1	0
T_2	0	1	3	0	1	0
T_3	2	0	1	5	2	0
T_4	1	0	1	3	2	0
T_5	2	1	2	2	2	0

이고, 이것의 similarity coefficient의 matrix $C=[C_{ij}]$는

$C=$

	T_1	T_2	T_3	T_4	T_5
T_1	1	1/9	8/9	5/9	5/9
T_2	1/5	1	2/5	2/5	4/5
T_3	8/10	2/10	1	7/10	7/10
T_4	5/7	2/7	7/7	1	6/7
T_5	5/9	4/9	7/9	6/9	1

이며, 여기에 cut-off value K를 0.20, 0.40, 0.60, 0.80, 1.00을 차례로 적용시키므로써, 다음의 hierarchical structure



를 얻는다.

이 방법은 Term들이 property를 가지도 빈도(度數)에 主眼點을 두고 있다.

즉, $C_{ij}<K$이고 $C_{ji}\geq K$ 일때 T_i가 T_j의 parent라는 관계는

$$C_{ij}=\frac{\sum\min(v^i, v^j)}{\sum v^i}<K$$

이고

$$C_{ji}=\frac{\sum\min(v^i, v^j)}{\sum v^j}\geq K$$

이므로

$$\sum v^i > \frac{\sum\min(v^i, v^j)}{K}\geq \sum v^j$$

이니까, 결국 $\sum v^i > \sum v^j$ 일때 T_i가 T_j의 parent임을 말한다. 즉 term들의 상호관계는, 그 term들의 property를 가지는 빈도수에 의하여 결정이 되고, 그렇게 구성된 hierarchy도 빈도수에 의한 hierarchy이다.

Term들이 어떤 property를 가지는 지의 여부만으로 hierarchy를 구성한다면, 우리는 그 term들의 상호관계를 定性的인 側面으로 생각할 수 있을 것이나. Information unit들의 상호관계는 다음과 같이 생각할 수 있다.

(i) A unit는 B unit가 가지는 모든 성질 이상의 것을 포함한다면 前者는 後者의 parent의 관계로,

(ii) A unit는 B unit가 가지는 성질을 적갈이 포함한다면, 같은 group의 관계로,

(iii) A unit는 B unit가 가지는 성질을 다 갖추지 못하면서, 또한 B unit가 가지지 않는 성질을 갖출 때, A와 B는 Brother의 관계로 규정한다.

Term T_i가 어떠한 성질 P_j를 가지면 V_{ij}를 1, 가지지 않으면 V_{ij}를 0으로 定하므로써, Term T들의 property P에 대한 Binary matrix를 만들 수 있다. 이러한 Binary Term-Property matrix로 부터 hierarchical structure(Tree Structure)를 만드는 Algorithm을 다음에 제시하고자 한다.

Ⅱ. Binary TermProperty matrix로부터 Tree를 구성하는데 있어서의 문제점

Binary Term-Property Matrix A는

	P_1	P_2 ····· P_m
T_1	a_{11}	a_{12} ····· a_{1m}
T_2	a_{21}	a_{22} ····· a_{2m}
⋮	⋮	⋮
T_n	a_{n1}	a_{n2} ····· a_{nm}

과 같이 표시할 수 있고, 이때 T_i가 Property P_j를 가지면 a_{ij}는 1이고, T_i가 Property P_j를 가지지 않으면 a_{ij}는 0이다.

(규정 1) Binary Term-Property matrix와 Tree의 관계는 다음과 같이 규정한다. Binary Term-Property matrix $A=[a_{ij}]$에 대하여, Property에 관한 Index Set을 $J=\{1,2,\cdots m\}$, Term에 관한 Index Set을 $I=\{1,2,\cdots n\}$이라 하고, 각 Term 간의 상호관계의 집합을 $R=\{.SG.,.PS.,.BB.\}$라 하면, 임의의 두 Term T_k, T_l은

(i) 모든 $j\in J$에 대하여 $a_{kj}=a_{lj}$이면 $T_k.SG.T_l$이다.

(ii) 모든 $j\in J$에 대하여 $a_{kj}\geq a_{lj}$이며 $T_k.PS.T_l$이다.

(iii) 어떤 $j\in J'\subset J$에 대하여는 $a_{kj}>a_{lj}$이며, 어떤 $j\in\overline{J'}$에 대하여는 $a_{kj}<a_{lj}$이면, $T_k.BB.T_l$이다.

(iv) 어떤 Term T_l에 대해서 $T_k.PS.T_l$되는 T_k는 하나 이상이어서는 안된다.

上記 규정에서 $T_k.SG.T_l$은 T_k와 T_l이 같은 group임을 의미하고, $T_k.PS.T_l$은 T_k가 T_l의 parent이고, T_l은 T_k의 son 임을 의미하며, $T_k.BB.T_l$의 관계는 T_k와 T_l이 서로 brother의 관계 즉, 무관함을 의미한다.

Tree의 도표상, $T_k.SG.T_l$이라 함은 T_k와 T_l이 한개의 같은 node에 대응됨을 말하며, $T_k.PS.T_l$이라 함은 $(T_k\rightarrow T_l)$의 Parent-Son의 관계로 대응시킬수 있음을 말하며, $T_k.BB.T_l$이라 함은 T_k와 T_l이 서로 다른 branch에 소속됨을 말한다.

예를들어 Binary Term-Property matrix가 (표 1)과 같은 때, 그것의 Tree는 上記의 규정 (i), (ii), (iii)의 조건만을 적용시키면 그림 (1)과 같이 표시된다.

(표 1)

	P_1	P_2	P_3
T_1	1	1	0
T_2	1	1	1
T_3	1	0	0
T_4	1	0	1
T_5	0	1	0
T_6	0	0	1
T_7	0	1	1
T_8	0	0	0

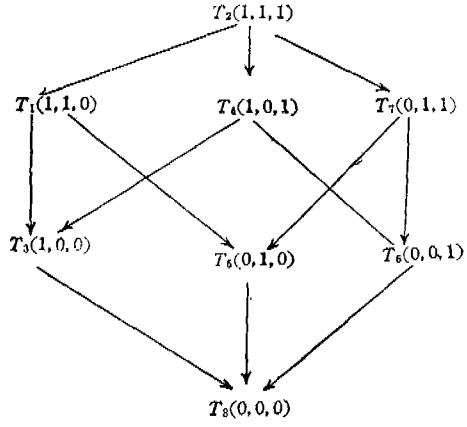


그림 1.

B-T-P matrix로 부터 Tree의 구성에 있어서, 첫번째의 문제는 단일한 parent를 가지게 하는 것이다. (Tree의 규정(iv)) 예컨데 (그림 1)에서 T_3, T_5, T_6는 각각 2개씩의 parent를 가지며, T_8은 3개의 parent를 가진다. 이들 parent의 수를 單一하게 하기 위해서는 이들 parent 중에서 한개를 선택해야 한다는 것이 그 한가지 방법이다.

만약, $P_1, P_2, \cdots P_m$의 각 property가 $T_1, T_2, \cdots T_n$의 분류에 영향을 주는 기여도가 크기 순서로 지정될 수 있다면, 난잎 parent를 가지는 member들의 Tree로 구성할 수 있다.

T에 주는 P_j의 기여도를 P_j의 weight(W_{P_j})라 하자. W_{P_j}의 크기 순으로 column을 변환시킨 matrix를 A'이 하면, A'는

	P_1'	P_2' ····· P_m'
T_1		
T_2		$(a_{ij})'$
⋮		
T_n		

와 같다.

이 matrix의 weight들은

$$W_{P_1'} > W_{P_2'} > W_{P_3'} \cdots\cdots > W_{P_m'}$$

이다. 특히 Binary Term-Property matrix의 모든 element는 0 혹은 1 이므로

$$W_{P_1'} = 2^{(m-1)},\ W_{P_2'} = 2^{(m-2)}, \cdots W_{P_m'} = 2^0$$

으로 둔다면, T_i vector는 한개의 binary number로 대치 할 수 있다.

즉,

$$\begin{aligned} \mathrm{val}(T_i) &= a_{i1}W_{P_1'} + a_{i2}W_{P_2'} + \cdots a_{im} < W_{Pm'} \\ &= a_{i1}2^{(m-1)} + a_{i2}2^{(m-2)} + \cdots a_{im}2^0 \\ &= \sum_{j-1}^{m} a_{ij}2^{(m-j)} \end{aligned}$$

이다.

그리고

$$\begin{aligned} \Delta_{kl} &= \mathrm{val}(T_k) - \mathrm{val}(T_l) \\ &= \sum_{j-1}^{m} (a_{kj} - a_{lj}) \cdot 2^{(m-j)} \end{aligned}$$

라 하자.

이제, 어떤 한 member T_q가 T_k와 T_l의 두 member를 parent로 가진다면

$$\begin{aligned} \Delta_{kq} &= \mathrm{val}(T_k) - \mathrm{val}(T_q) \\ &= (a_{k1} - a_{q1})2^{(m-1)} + (a_{k2} - a_{q2}) \cdot 2^{(m-2)} + \cdots \\ &\quad + (a_{km} - a_{qm}) \cdot 2^0 \end{aligned}$$

이고,

$$\begin{aligned} \Delta_{lq} &= \mathrm{val}(T_l) - \mathrm{val}(T_q) \\ &= (a_{l1} - a_{q1}) \cdot 2^{(m-1)} + (a_{l2} - a_{q2}) \cdot 2^{m-2} + \\ &\quad \cdots\cdots + (a_{lm} - a_{qm}) \cdot 2^0 \end{aligned}$$

이며, 만약 $\Delta_{kq} > \Delta_{lq}$라면, T_k보다 T_l이 더 적은 weight의 Term에서 T_q와 차이가 나므로, T_k보다는 T_l이 더 많은 관계를 T_q와 가진다고 볼 수 있으므로, T_l을 T_q의 parent로 삼는 것이 타당하다.

요약하면,

(v) $T_k.PS.T_q$이고 $T_l.PS.T_q$ 인데 $\Delta_{kq} > \Delta_{lq}$이면 $T_l.PS.T_q$의 관계만을 선택한다.

上記 (v)의 관계를 이용하면, (표 1)에서 $W_{P_1} > W_{P_2} > W_{P_3}$로 배열 되었다 하면, (그림 1)의 T_3는 T_1과 T_4의 두 parent를 가지나

$$\begin{aligned} \Delta_{13} &= \mathrm{val}(T_1) - \mathrm{val}(T_3) \\ &= (110)_2 - (100)_2 \\ &= (010)_2 \\ \Delta_{43} &= \mathrm{val}(T_4) - \mathrm{val}(T_3) \\ &= (101)_2 - (100)_2 \\ &= (001)_2 \end{aligned}$$

곧, $\Delta_{13} > \Delta_{43}$이므로 $T_1.PS.T_3$보다는 $T_4.PS.T_3$의 관계로 선택된다. 같은 방법으로 나머지 Term들의 parent도 單一化하면 (그림 Ⅰ)의 Tree는 (그림 Ⅱ)의 Tree로 된다.

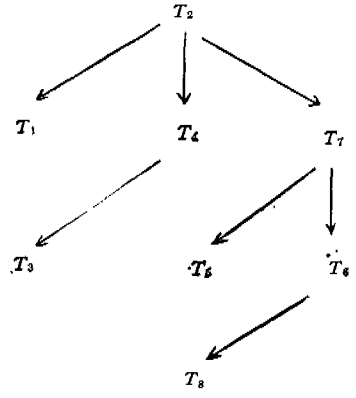


그림 Ⅱ.

Term T_k와 T_l의 각 property의 attribute를 vector의 한 成分으로 생각하면,

$$T_k = (a_{k1}, a_{k2}, \cdots\cdots\cdots a_{km})$$

이고

$$T_l = (a_{l1}, a_{l2}, \cdots\cdots\cdots a_{lm})$$

이다. vector T_k와 vector T_l의 anding을 T_k & T_l로, oring을 $T_k|T_l$로 하여 다음과 같이 導入한다.

$$T_k \ \&\ T_l = (a_{k1} \ \&\ a_{l1},\ a_{k2} \ \&\ a_{l2}, \cdots a_{km} \ \&\ a_{lm})$$

$$T_k|T_l = (a_{k1}|a_{l1}, a_{k2}|a_{l2}, \cdots a_{km}|a_{lm})$$

&은 binary anding operation, |은 binary oring operation을 의미한다.

이러한 operation을 利用하면 (규정 Ⅰ)의 (i), (ii), (iii)는 각각 다음의 (i'), (ii'), (iii')로 바꾸어 놓을수 있다.

(i') $T_k = T_l$이면 $T_k.SG.T_l$이다.

(ii') $T_k \ \&\ T_l = T_l$이고, $T_k|T_l = T_k$이면, $T_k.PS.T_l$이다.

(iii') $\mathrm{val}(T_k|T_l) > \mathrm{val}(T_k \ \&\ T_l)$이면, $T_k.BB.T_l$이다.

(증명) 모든 $j \in J$에 대하여 $a_{ij} = a_{lj}$이면 $T_k = T_l$이며 또 $T_k = T_l$이란 모든 $j \in J$에 대하여 $a_{kj} = a_{lj}$ 임을 의미하므로 (i)은 (i')와 같이 바꾸어 놓을 수 있다.

또, 모든 $j \in J$에 대하여 $a_{kj} \geq a_{lj}$이면, 이것은 a_{lj}

가 0일때는 a_{kj}가 1 혹은 0, a_{lj}가 1일때는 a_{kj}가 1임을 의미한다. 따라서 첫번째 경우,

$$a_{kj} \,\&\, a_{lj}=a_{lj},\ a_{kj}|a_{lj}=a_{kj}$$

이며, 두번째 경우,

$$a_{kj} \,\&\, a_{lj}=a_{lj},\ a_{kj}|a_{lj}=a_{kj}$$

이므로,

$$T_k\&T_l=(a_{k1}\&a_{l1},\ a_{k2}\&a_{l2},\cdots a_{km}\&a_{lm})$$
$$=(a_{l1}, a_{l2},\cdots\cdots a_{lm})$$
$$=T_l$$

이며,

$$T_k|T_l=(a_{k1}|a_{l1}, a_{k2}|a_{l2},\cdots\cdots a_{km}|a_{lm})$$
$$=(a_{k1}, a_{k2},\cdots\cdots a_{km})$$
$$=T_k$$

이다. 즉, 모든 $j\in J$에 대하여 $a_{kj}\geq a_{lj}$이라는 것은 $T_k\&T_l=T_l$이고 $T_k|T_l=T_k$임을 의미하므로 (ii)는 (ii′)로 바꾸어 놓을 수 있다.

또, 어떤 $j\in J'\subset J$에 대하여는 $a_{kj}\geq a_{lj}$이며, 어떤 $j\in\overline{J'}$에 대하여는 $a_{kj}<a_{lj}$이라는 것은,

$$\mathrm{val}(T_k|T_l)=\mathrm{val}(a_{k1}|a_{l1}|a_{l1},\cdots\cdots a_{k2}|a_{l2}, a_{km}|a_{lm})$$
$$>\mathrm{val}(a_{k1}, a_{k2},\cdots\cdots a_{km})\ \text{ or }\ \mathrm{val}(a_{l1}, a_{l2},\cdots\cdots a_{lm})>$$
$$\mathrm{val}(a_{k1}\&a_{l1}, a_{k2}\&a_{l2},\cdots\cdots a_{km}\&a_{lm})$$

이다. 즉 $\mathrm{val}(T_k|T_l)>\mathrm{val}(T_k)$ or $\mathrm{val}(T_l)>\mathrm{val}(T_k\ \&T'_l)$이다. 따라서 (iii)은 (iii′)로 바꾸어 놓을 수있다.

(증명 끝)

(iv)와 (v)는 다시 다음의 (iv′)로 바꾸어 놓을 수 있다.

(iv′) $T_k.PS.T_l$되는 T_k가 여러개 존재할때, 그 것들을 $T_{K_1}, T_{K_2},\cdots T_{K_n}$라 하며, $\Delta_{K_i l}$이 minimum되는 T_{K_i}를 선정하여 $T_{K_i}.PS.T_l$ 만을 인정한다.

Binary Term-Property matrix로 부터 Tree를 구성함에 있어서 또 다른 문제는, 어떤 Top T_j들에 대하여 parent가 존재치 않는 경우이다. 이때 만약, dummy term T_D를

$$T_D=(\oplus a_{i1}, \oplus a_{i2},\cdots\cdots \oplus a_{im})$$

$(\oplus a_{ij}=a_{1j}|a_{2j}|a_{3j}\cdots\cdots|a_{nj}$이고 ⊕는 or operation을 의미한다)

로 두면, T_D는 모든 term들의 parent이다. 왜냐하면, 임의의 term T_l에 대해서

$$T_D\&T_l=T_l$$

이고,

$$T_D|T_l=T_D$$

이므로, $T_D.PS.T_l$이기 때문이다. T_D를 parent가 없는 term들 (T_l)의 parent로 삼아서, tree를 만든 후에 T_D를 제거하면, T_l들은 각각 독자적 Head parent(Top)이 되게 된다.

Ⅲ. *B-T-P* matrix로 부터 Tree Structure를 구성하는 Algorithm

이제껏 논증한 방법에 의거하여 *B-T-P* matrix로 부터 Tree를 만드는 algorithm은 다음과 같다.

(a) *B-T-P* matrix의 배열 : Binary Term-Property matrix를 nxm array로 Source area에 배열시킨다.

(그림 Ⅲ)의 source area array의 element를 S_{ij}라 하고 *B-T-P* matrix의 element를 a_{ij}라 하고 ←을 assign 기호라 하면,

$$((S_{ij}\leftarrow a_{ij})\ i=1,2,\cdots n,\ j=1,2,\cdots m)$$

으로 요약할 수 있다.

(b) property column 변환 : Source area의 *B-T-P* matrix를, column변환에 의하여, Property의 weight 順으로 再配列시킨다. 즉, ↔을 두 element. 상호교환을 표시하게 한다면,

$$(W_{P_j}<W_{P_k}\text{인 경우 } S_{ij}\leftrightarrow S_{ik},\ i=1,2,\cdots n)\ j=1,2,\cdots m,\ k=j+1, j+2,\cdots m)$$

을 시행한다.

(c) degree계산 : 각 term들의 degree를 degree $(T_i)=\sum_{j=1}^{m} S_{ij}$에 의하여 구한다. Source area에는 다음과 같이 한다.

$$\left(\left(\mathrm{degree}(T_i)\leftarrow \sum_{j=1}^{m} S_{ij}\right)\ i=1,2,\cdots n\right)$$

(d) Top 작성 : Source area에서 degree (T)가 최대인 term들의 집합을

$$\tau_m=\{T_i|i\in I,\ \mathrm{degree}(T_i)\geq\mathrm{degree}(T_k),$$
$$k=1,2,\cdots\cdots n\}$$

라 할때, τ_m의 갯수가 하나인 경우에 그 T_i를 Top으로 삼고, τ_m의 갯수가 그 이상이면

$$T_D=(\oplus a_{i1},\ \oplus a_{i2},\cdots\cdots\oplus a^i_m)$$

인 T_D를 만들어 Top으로 삼는다.

즉,

$\mathrm{degree}(\tau_m)=1$이면 Top←$T_i$,

$\mathrm{degree}(\tau_m)\geq 2$이면 Top←$T_D$

(e) T의 Tree area이전 : source area에 삭제되지 않고 남아있는 term들중에서 degree (T)가 가장 큰 term들의 집합을

$\tau=\{T_i|i\in I,\ \text{degree}(T_i)\geq\text{degree}(T_k),\ K\subseteq I,$

K는 source area에 남아있는 T_i들의 index}

라하면, τ중의 T_i들을 source area에서 Tree area로 옮기고, source area로 부터는 삭제한다. 즉 degree 순위가 p이고, p degree 순위 중, q Term unit라면,

$((TN(p,q)\leftarrow T_q,)T_q\in T,\ q=1,2,\cdots\text{degree}(T))$

이고, TN은 Term unit record의 term name 난이다. τ의 각 element T_q를 $TN(p,q)$에 assign할 때마다 $q\leftarrow q+1$로 하고, T의 T를 모두 assign 시킨 후에는 $p\leftarrow p+1$로 하여, 다음 degree의 Tree area를 가리키고 있도록 한다.

(f) parent-son의 관계 조성 : Tree area에서 (e) 단계에서 들어온 term들의 집합을 τ_p라하고 그 전에 들어온 higher degree term들의 집합을 τ_{p-1}이라 하면

$\tau_p=\{T_k|K$는 $TN(\text{degree}\ (\tau_p),q)$이 가리키는 T들의 index}

이고

$\tau_{p-1}=\{T_l|l$은 $TN(\text{degree}(T_{p-1}),q)$이 가리키는 T들의 index}

이다.

$T_q,T_r\in\tau_p$에 대하여

$T_q=T_r$이면, $T_q.SG.T_r$이므로

$SG(T_q)\leftarrow T_r$,

$SG(T_r)\leftarrow T_q$

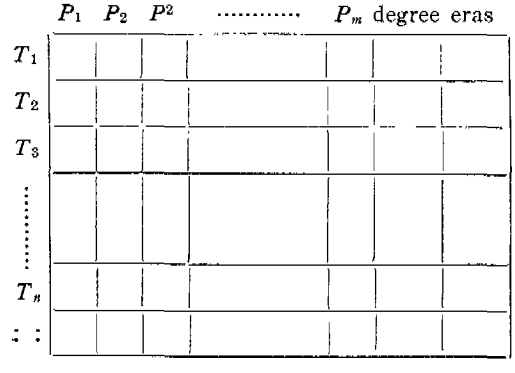
되게하고,

$T_k\in\tau_{p-1}$, $T_l\in\tau_p$인 T_k,T_l에 대하여

$T_k\&T_l=T_l$이고, $T_k|T_l=T_k$이면서

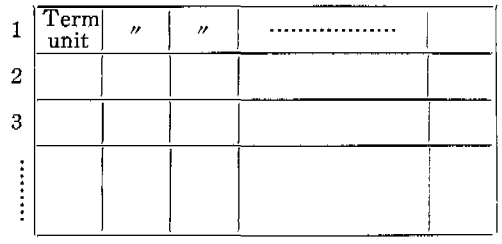
Δ_{kl}이 minimum이 되는 T_k,T_l은

Source area



Tree area

degree순위



Term unit(TU)

Term name TN	Same group term SG	parent term PA	son term SO

그림 Ⅲ.

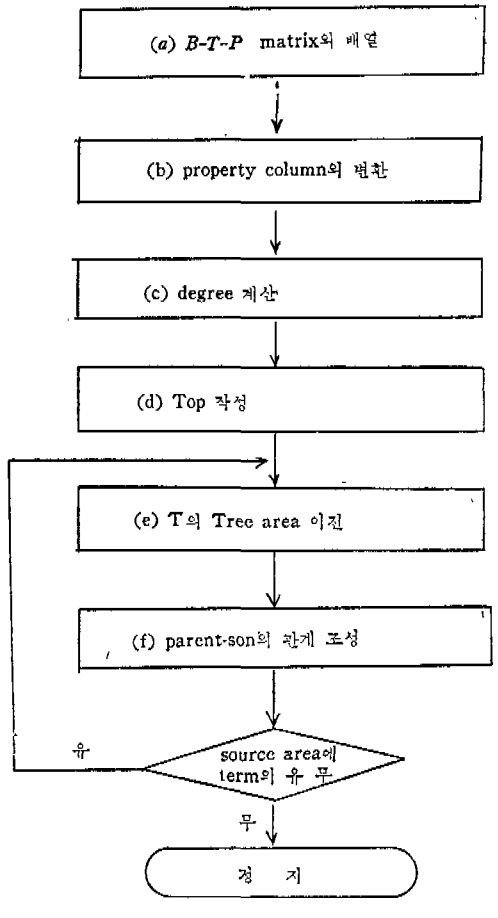


그림 Ⅳ.

$T_k.PS.T_l$이므로

$PA(T_l) \leftarrow T_k$,

$SO(T_k) \leftarrow T_l$

되게 한다.

(g) 정지 : source area에 term이 남아 있으면 (e)로 보내고, 그렇지 않으면 정지시킨다.

Source area와 Tree area및 Term unit record는 (그림 Ⅲ)과 같고, 上記 Algorithm에 대한 Block diagram은 (그림 Ⅳ)와 같다.

Ⅳ. *B-T-P* matrix로 부터 만들어진 Tree structure의 성질 및 결론

Tree structure 上에 T_i가 T_j의 parent로 표시되었다면, T_i는 T_j가 가지는 모든 property를 포함하며, T_i와 T_j가 같은 node 상에 놓여졌다면 T_i와 T_j는 定性的인 面에서 同一하며, T_i와 T_j가 tree structure 上에 parent-son과 同一 node 以外의 것으로 나타났다면, T_i는 T_j의 property를 一部 혹은 全部를 가지고 있지 않음을 의미하게 된다.

情報의 表現은, 定性的으로 생각한다면 *B-T-P* matrix로 규정짓고 이를 Tree structure로 구성하여 體系化할수 있음이 보여졌다.

참 고 문 헌

1. Automatic Information Organization and Retrieval p. 34~64, Gerard Salton, Mc Graw-Hill.
2. Techniques for Thesaurus Organization and Evaluation, Abraham, C.T., Scarecrow Press. Inc., Metucher, N.J., 1965.
3. An Experimental Investigation of Automatic Hierarchy Construction, Blomgren, G., A. Goodman, and L. Kelly, Report ISR-11, Cornell Univ. June, 1966.
4. Data Structure and manugement, I. Flores, Prentice Hall, 1970.
5. Data Structure Theory and Practice, A.T. Berztiss, Academic Press, 1971.
6. The Art of Computer Programming Vol1. Donald E. Knuth, Addison Wesley, 1973.