상품 속성정보를 이용한 분류체계 자동생성

장 두 석[†] · 전 종 훈[††]

요 약

온라인상에서 거래되는 상품들을 분류하고 관리하기 위해서는 많은 시간과 비용을 들여 상품분류체계를 유지하여야 한다. 일반적으로 상품을 다루는 모든 분야에서 분류체계는 분류전문가에 의하여 수동으로 관리되고 있으며 이는 경제적인 측면, 시간적인 측면에서 많은 낭비를 초래하게 된다. 현대사회에서는 산업의 급속한 발전으로 상품의 다양화 융합화 등이 활발하게 이루어져 상품을 효율적으로 관리하기 위한 분류체계의 필요성은 더더욱 증가하고 있다. 따라서 상품분류체계를 자동화 하고자 하는 연구들이 많이 진행되어 왔으며 이런 연구의 일환으로 본 논문에서는 분류체계를 자동으로 생성하는 방안을 제안한다. 각각의 상품은 속성의 집합이다 라는 관점에서 출발하여 각 상품, 즉 속성집합 간 존재하는 포함관계를 활용하여 계층 트리구조의 분류체계를 자동으로 생성하는 알고리즘을 제시하고 구현하였으며, 실험을 통하여 제안한 알고리즘의 실효성을 입증하였다.

키워드 : 분류체계, 상품 데이터베이스, 전자상거래, 전자카탈로그, 상품 속성

Automated Classification Scheme Generation using Product Attribute Information

Duseok Jang · Jonghoon Chun

ABSTRACT

In order to classify and manage on-line trading goods, the product classification scheme must be maintained. In most systems for handling product information, the classification scheme is managed manually by experts, which in general incurs a lot of time and cost. Effective management of classification system becomes more important as rapid development of industry expedites diversity and convergence of goods and services. There have been many researches on developing classification scheme, and continuing in this line of research, this paper proposes a new method for automatic generation of product classification scheme. Our main idea starts from the concept that a product is a set of attributes, and we propose a novel algorithm for automatically creating hierarchical classification scheme by utilizing inclusive relationships between products. We then prove the effectiveness of proposed algorithm by conducting an experiment.

Key Words : Classification Scheme, Product Database, e-Commerce, Electronic Catalogues, Product Attribute

1. 서 론

현대사회에서는 다양한 산업분야가 새로이 발생하면서 이에 따른 상품 종류수가 급격히 증가하고, 산업분야 간의 융합화가 가속화되어 상품의 기능이 복합적이고 다양화되어 상품을 체계적으로 관리하기가 점점 더 어려워지고 있다. 또한 사용자는 다양하고 복잡한 상품정보로 부터 쉽게 상품에 대한 정보를 획득하고자함은 물론, 상품간의 비교분석 등과 같은 요구사항을 가지기도 한다. 이런 이유로 모든 상거래 분야, 특히 전자상거래에서는 상품을 효율적이고 체계적으로 관리하여야 하는 요구사항이 생기게 되며, 따라서 사용자가 원하는 정보를 쉽게 얻을 수 있도록 상품들을 일정한 기준을 가지고 분류하여 관리하는 분류체계를 구축하는 일은 더욱 중요하게 되었다.

분류체계를 생성하기 위해서는 전체적인 상품의 구조 및 특성을 파악하는데 많은 경험과 시간, 노력이 필요하며 따라서 분류체계를 생성하는 작업은 상품 및 분류에 대한 많은 노하우를 가진 전문가에 의해 이루어진다. 하지만 산업발전의 속도가 가속화되고, 이에 따른 상품의 종류가 급격히 증가하며, 상품의 속성이 다양해지고, 상품간의 속성들이 서로 융합화 되어 전혀 다른 특성을 가질 수도 있는 등, 엄청나게 증가하는 상품의 종류수와 속성을 고려할 때 실질적으로 분류전문가 개인의 능력만으로 분류체계를 생성한다는 것은 많은 시간과 노력이 들뿐만 아니라 분류체계의 일관성도 결여된다는 단점이 있다. 물론 UNSPSC등 표준 상품분류체계가 존재하나, 대부분의 국제표준들은 코드기반 분류체계로 되어있어 새로운 분류항목에 대한 추가 및 변경은

† 정 회 원 : 명지대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 명지대학교 컴퓨터공학과 교수
논문접수 : 2007년 1월 9일, 심사완료 : 2007년 4월 23일

상품 분류체계 관리기관에 요청하여 심사에 의해 이루어지기 때문에 새로운 분류 항목을 바로 적용하기가 어렵고 경우에 따라서는 하나의 상품이 1개 이상의 분류군에 포함되어야 하는 등 모호한 경우도 발생한다. 반면 기업체 내부에서 자체 구축하여 사용하는 분류체계는 국제표준보다 쉽게 수정하고 추가할 수 있으며 필요에 의해 언제든지 재생성도 가능하여 아직까지 많은 기업체들은 자신만의 고유한 분류체계를 가지고 상품을 분류하고 있으며 국제 표준이 필요한 경우 자사의 분류체계와 국제표준을 연결하여 사용하고 있는 실정이다.

일반적으로 B2C나 B2B 순수 온라인 전자상거래 업체의 경우 상품분류체계를 유지하고 관리하는데 전체 비용의 70% 정도가 소요되는 것으로 알려져 있다[1]. 또한 수작업에 의한 분류체계 생성은 분류체계를 생성하는 담당자의 능력에 따라 분류체계가 달라짐으로 인해 분류체계의 일관성 및 신뢰성이 떨어진다. 분류체계 변경이나 추가사항이 발생할 경우 전체적인 분류체계를 숙지하고 있는 전문가의 도움이 반드시 필요하여 유지보수에 많은 비용이 들게 든다. 따라서 특정 전문가의 능력에 의존적으로 생성되는 분류체계가 아닌 자동으로 상품분류체계를 생성하는 것이 필요하며 결과적으로는 상품분류체계 유지 관리에 소요되는 시간과 비용 절감을 가능하게 한다.

본 논문에서는 상품의 분류체계를 자동으로 생성하는 방안을 제시하고자 한다. 주어진 상품의 속성정보들은 본 논문에서 제시하는 분류체계 자동생성 알고리즘에 의하여 상호간의 포함관계가 파악되며, 이를 기반으로 분류체계가 자동으로 생성된다. 자동으로 생성되는 분류체계는 깊이 우선의 계층적 트리구조를 가지게 되며 사용자의 개입에 의해 원하는 형태로 변환되어 사용될 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 분류 체계, 전자상거래, 의미적 분류모형 등 관련 이론적 배경을 살펴보고 3장에서는 본 논문의 주된 내용인 상품 속성정보에 기반한 상품 분류체계 자동생성 알고리즘을 제안하고 4장에서는 실험결과에 대해 언급하며 5장에서는 결론을 맺는다.

2. 분류체계 관련 지식

2.1 분류체계

분류란 여러 가지 사물이 뒤섞여 있는 가운데에서 비슷한 것끼리 묶어서 경계를 인식하는 활동이다. 분류를 통해 사물에 대한 본질을 파악하고 정보를 조직하여 막연히 아는 것을 분명하게 알게 하고, 전체를 골고루 빠짐없이 알게 하며, 사물의 특성을 빠르게 판단하도록 해준다[2]. 분류체계는 분류의 기초가 되는 것으로 분류 기호 및 그것에 대응하는 용어를 사용하여 문헌 또는 데이터를 구조화하여 표현하는 체계이다. 분류와 분류체계는 정보를 조직화하기 위해서는 반드시 필요하고 시스템으로 구현할 때도 분류와 분류체계는 두 가지 다 구현되어야만 의미가 있다.

현재 통용되고 있는 국제 표준 분류 코드체계로서는 HS (Harmonized System, 통일상품분류체계), SITC(Standard International Trade Classification, 국제표준무역분류), UNCCS (United Nations Common Coding System, UN공용분류시스템), UN/SPSC(UN/Standard Products and Services Classification, UN 표준상품서비스분류), NICE(상품서비스국제분류표), SKTC(Standard Korean Trade Classification, 한국표준무역분류)등이 있으며, 컴퓨터 데이터베이스에서 상품과 서비스를 식별하기 위한 식별코드체계로서는 EAN/UCC(Europian Article Number/Uniform Code Council), UPC(Universal Product Code), UNDP(United Nations Development Programme)등이 있다.

UNSPSC는 UN의 UNCCS와 세계적인 기업 신용평가 기관인 D&B사의 SPSC(Standard Product and Services Codes) 코드가 결합되어 만들어진 것으로서 전자상거래를 위한 상품 및 서비스용 분류표준으로 개발된 가장 광범위하고 체계적인 분류체계라 할 수 있다. 분류 구조는 Segment, Family, Class, Commodity 4단계의 계층구조를 이루며, 각 단계가 두 자리 수로 구성되어 총8자리수의 분류코드로 이루어져 있다[4]. UNSPSC가 글로벌 e마켓플레이스 분야에서 사용되는 상품과 서비스를 분류하는 표준들 중에서 가장 많이 사용하는 상품분류코드라는 점과 B2B 전자상거래에서 발생하는 거래의 다양한 품목을 업종이나 특성에 따라 체계적으로 분류할 수 있는 특성을 가지고 있다는 점이다[3].

NAICS는 1997년 미국, 캐나다, 멕시코에서 제정된 산업분류 표준으로 국내에서는 사용되지 않으며 북미 표준으로 국내의 한국 표준 산업 분류와 유사한 형태이다, UDDI에서는 이 코드를 표준으로 삼는다[5].

AN/UCC는 상품 및 유통 정보의 바코드화를 통한 중앙집중적인 관리를 목적으로 1997년 EAN International 과 UCC에 의해 제정되었으며 현재 100개 이상의 국가의 90만개 이상의 기업이 사용하고 있으며 특징으로는 각 상품에 대해 고유한 번호를 부여하여 바코드를 통해 상품 정보 및 유통 정보를 얻을 수 있으며 국내에서는 소비재 부분에 대해 분류코드가 정의되어 있으며, 약 30만개의 단품이 등록되어 있다[6].

UPC는 UCC에서 관리하며, 미국과 캐나다에서 사용되고 있는 12자리 공통상품코드로 넘버시스템 캐릭터 1자리, 제조업체코드 5자리, 상품품목코드 5자리, 체크디지트 1자리로 구성되어 있다. 12자리 코드체계는 UPC-A로서 표준형이고, 8자리의 UPC-E라는 단축형도 존재한다[7].

HS는 수출입 상품을 위한 체계, 관세 분석을 목적으로 1988년 CCC(관세협력이사회)에 의해 제정되었으며 현재 176개 국가의 국가에서 사용함으로써 세계 무역의 98% 이상을 담당하고 있으며 특징으로는 국제 무역의 필요에 의해 제정된 전통적 분류체계로 무역관련 업체에 의해 많이 사용되고 있으며 전자 카탈로그에도 쉽게 적용할 수 있다[8].

다양한 분야에서 다양한 용도로 상품 분류에 대한 국제표준분류체계를 제정하여 운용하고 있지만 기업체에서는 여전히 표준 분류체계보다는 업체의 특성에 맞는 자사의 상품 분류체계를 운영하고 있다.

2.2 전자 카탈로그

전자상거래란 기업과 소비자(B2C), 기업과 기업(B2B), 기업과 정부(B2G)간의 정보나 재화와 용역의 서비스를 전자적인 매체 또는 개방 네트워크를 통해 서비스 하는 것이다[9]. 전자카탈로그란 전자상거래를 위하여 상품의 고유 속성 정보를 나타내는 상품정보(상품명, 규격 등), 상품들을 구분하기 위해 필요한 식별정보(분류명, 분류코드, 제품번호 등), 판매정보(재고유무, 판매단위, 포장단위, 유통단위, 가격정보 등)를 전자적인 형태로 저장하고 교환하기 위한 일종의 전자문서를 의미한다. 전자카탈로그 구축의 궁극적인 목적은 다양한 비즈니스 주체(개인, 기업, 정부 등)간의 카탈로그 정보의 공유에 있다.

현재 많은 단일 벤더 온라인 마켓이 운영되고 있으며 또한 온라인 거래시장만을 제공하고 수많은 벤더에 의해 운영되는 오픈마켓이 점점 더 활성화되어가고 온라인상에서 거래되는 상품의 수가 증가할수록 전자카탈로그 시스템은 점점 더 많은 유지보수와 생성이 필요하다. 따라서 전자상거래의 기본인 전자카탈로그를 보다 효율적으로 관리하고 조회하려는 연구[10, 11, 12]들이 진행되었다.

전자상거래의 기본인 전자카탈로그와 전자카탈로그를 온라인상에서 효율적으로 관리하기 위한 방법 즉 분류체계의 필요성은 전자상거래가 활성화되면 될 수록 점점 더 중요해진다.

2.3 분류체계 자동생성을 위한 연구

상품을 체계적으로 분류하고 관리하기 위해서는 분류체계는 필수적이다. 이미 국제적으로 통용되는 상품분류 체계가 있지만 항목의 추가, 변경 등 표준화코드의 한계성으로 인해 아직까지 많은 기업들이 자사의 고유한 상품분류체계를 생성하고 운영하고자 한다. 다루는 상품수와 속성이 그리 많지 않던 시절에는 수작업으로 분류체계를 생성하고 관리하였으나 상품수가 다양해지고 여러 상품들의 기능이 융합되어 다양한 상품속성을 가지는 현재의 상황에서 수작업으로 분류체계를 생성하는 것은 많은 시간 및 비용이 소요된다. 이런 낭비적인 요소를 제거하고자 상품분류체계를 자동으로 생성하려는 연구들이 많이 이루어졌다. 관련 연구로는 분류체계를 자동으로 생성하기 위해 기계학습 기법을 이용하는 방법[13], 벡터 스페이스 모델과 상품의 특징을 나타내는 단어들을 이용하여 상품분류체계를 자동으로 생성하고자 한 Autocat[14], Naive-Bayesian Classifier을 확장하여 사용자의 개입 없이 자동으로 유사한 카탈로그를 분류하는 방법[15, 16], UNSPSC, UNDP[17]등 널리 알려진 코드 기반의 국제 상품분류체계와 매핑이 필요한 경우 상품에 내포되어 있는 의미를 파악하여 의미적인 연관관계를 찾아 매핑할 수 있도록 분류체계를 조정하고 재 생성하는 기법에 관한 연구[18], 그리고 [20]에서는 B2B에서 공급자와 발주자 사이의 상품정보 공유 시 문제점에 대한 언급과 문제점을 해결하는 방안으로 상품의 속성들을 구조적으로 정의하는 방안을 제시하였다. 또한 [21]은 온톨로지 기반의 상품분류체계를 생성하는 방법을 제안하였으며, 상품 속성 이면에 숨어 있는 의미를 찾아 상품의 속성들 간에는 구조적 관계가 형성하므로[22] 상품속성들 간의 의미적인 계층구조를 생성하고 하나의 정의된 상품분류는 의미적으로 여러 개의 다양한 분류군에 포함될 수 있도록 하는 의미적 분류 모형을 제안하였다[19, 23]. 기존의 국제표준분류체계 시스템들이 코드기반 분류체계로서 수시로 변하는 상품의 특성을 제대로 반영하지 못하는 문제점의 대안으로 상품 속성에 포함되어 있는 고유한 의미를 활용하여 분류체계를 구축하는 분류체계 모형[23, 24]이 있다. [25]는 온라인상에서 상품을 검색 할 때 공급자에 의해 제공되는 획일적인 정보 검색의 문제점을 보완하기 위해 정보검색과 상품 분류체계 생성을 위해 기계학습을 이용한 GoldenBullet이 제시되었다. [14]는 기존의 정보검색 알고리즘과 유사하며 각 상품의 속성을 나타내는 특징 단어들 간의 연관관계를 벡터를 이용하여 유사도 계산을 하여 UNSPSC 코드체계처럼 카테고리화 하였으며, [25]는 상품의 특성 명칭들을 전처리과정을 거쳐 정제화한 후에 각 특징 단어들 간의 벡터스페이스모델, 나이브 베이시안 분류기등을 이용하여 UNSPSC코드 체계처럼 4개의 분류그룹으로 분류체계를 생성 하였다. [14, 25]는 각 상품의 특징을 나타내는 단어들만 가지고 단어들 사이의 유사도 계산에 의한 방법으로 분류체계를 생성하였기에 모든 상품을 100%완벽하게 포함하는 분류체계를 형성하지는 못한다.

상품을 구성하는 요소 중 가장 기본이 상품의 특징을 나타내는 속성이며, 속성은 상품에 대한 많은 정보를 가지고 있다. 따라서 상품을 분류하기 위해서는 속성에 대한 정보를 반드시 알아야 하며, 수작업에 의한 분류체계를 생성한다고 하더라도, 상품의 속성을 참조할 수밖에 없다. 이처럼 중요한 상품속성을 간과하고 분류체계를 생성한다는 것은 의미가 없다. 그러므로 본 논문에서는 상품을 구성하는 가장 기본인 속성을 이용하는 분류체계를 생성하고자 한다. 물론 상품의 분류체계를 자동으로 생성하려는 다양한 연구들이 진행되었으나 많은 연구들이 단순히 상품 속성 명칭을 이용하여 클러스터링이나 마이닝 기법 등, 유사도 기반 검색을 하는 방향으로 연구들이 진행되었으며, 본 논문에서처럼 상품을 속성의 집합으로 해석하여 속성들 사이의 포함관계를 찾아, 모든 속성들이 포함되는 분류체계를 자동으로 생성하는 연구는 없었다.

3. 알고리즘

3.1 개요

모든 상품에는 각각 자신의 고유한 특성을 나타내는 속성정보가 존재하므로, 이러한 속성정보들 간에 내포되어 있는 포함관계를 활용하면 상품들 간의 계층구조를 찾을 수 있으며, 이는 곧 상품 분류체계를 자동으로 생성할 수 있음을 의미한다.

따라서 본 장에서는 각 상품의 속성정보를 이용하여 상품 속성들 간의 포함관계를 찾아내고, 계층구조를 생성하는 방안에 대해 제안하고자 한다.

(그림 1)에서 ①은 외부 파일로 작성된 상품과 상품속성

```
BEGIN
   load CE                          ——①
   do
      COMPUTE_GCA(CE)               ——②
   while(EOF of CE)
   sort GCA                         ——③
   do
      COMPUTE_RCA(GCA)              ——④
   while (EOF of GCA)

   COMPUTE_PCR(RCA)                 ——⑤
   Create TREE                      ——⑥
END
```

(그림 1) 상품 분류체계 자동 생성 알고리즘

정보를 메모리에 로딩하는 단계이며, 메모리에 로딩된 상품과 상품속성을 ②단계에서는 각각의 상품을 상품 속성집합으로 분류하고, 상품명은 고유한 상품아이디로 표현하며, ③은 ②에서 생성된 속성집합을 ④에서 효율적으로 사용하기 위해 일반적으로 사용하는 정렬 알고리즘을 이용하여 정렬한다. 정렬된 속성집합을 ④에서는 같은 성질의 속성집합을 하나의 속성집합으로 통합하고, ⑤는 속성들 간의 포함관계(부모-자식)를 찾아내고, ⑥은 ⑤에서 생성된 포함관계를 이용하여 계층트리를 생성한다.

알고리즘에 사용되는 용어들의 정의는 다음과 같다.

N = {x | x는 자연수}

A = {A_i | A_i는 상품의 속성, $A_i \neq A_{i-k}$}

P = {P_i | P_i는 상품명, $\forall P_i \supset A_j(A_j$: 모든 상품의 공통 속성)}

CE = {(P_i, A_{j1}, A_{j2}, A_{j3}...A_{jk}) | $P_i \in P$, $A_{jk} \in A$, 모든 A_{jk}는 P_i의 속성 }

GCE = {CE_i | CE_i는 GCE의 i번째 CE}

$GPID$ = {(P_i, Pid_j) | $P_i \in P$, Pid_j는 P_i의 고유 아이디 $Pid_j \in N$ }

CA = {(A_i, $CntOfPid_l$, $SumOfPid_l$, $MaxOfPid_l$, Pid_{j1}, Pid_{j2}, Pid_{j3} .... Pid_{jk}) | $A_i \in A$, $Pid_{jk} \in GPID$, 모든 Pid_{jk}는 A_i를 포함하는 상품아이디, $CntOfPid_l = n(Pid_{jk})$, $SumOfPid_l = \sum(Pid_{jk})$, $MaxOfPid_l = Max(Pid_{jk})$ }

GCA = {CA_n | CA_n는 GCA의 n번째 CA }

RA = {CA_i | $CA_i = CA_{i-1}$ 이면 CA_i, $CA_i \in GCA$, $CA_{i-1} \in GCA$, CA_i는 중복이 제거된 CA}

RCA = {RA_i | RA_i는 RCA의 i번째 RA }

PR = {(RA_i, RA_{i-k}) | $RA_i \in RCA$, $RA_{i-k} \in RCA$, RA_i는 자식노드, RA_{i-k}는 부모노드, $RA_i = RA_{i+k}$ 이면 Root 노드 }

PCR = {PR_i | PR_i는 PCR의 i번째 PR}

알고리즘의 제한 조건으로는 계층트리를 생성하기 위해, 모든 상품 P는 최소한 하나 이상의 공통된 속성(A_1)을 포함하고 있으며, 상품 속성 A_i는 의미적으로 전체를 통틀어서 유일해야하며, 상품 속성이 가지고 있는 데이터베이스적인 특징(길이, 타입 등)도 동일하다고 본다.

제안한 알고리즘은 상품과 상품속성으로 이루어진 상품집합을 속성으로 분리하여, 속성들 사이에 포함되어 있는 관계를 찾아, 계층구조로 표현하는 알고리즘이다. 따라서 속성간의 포함관계를 찾는 방식은 집합을 이용하여 정의한다.

정의 1. *상품 속성별로 분류된 두 집합 RA_i와 RA_{i-k}사이에 $RA_i \subset RA_{i-k}$ 이고 $RA_i \neq RA_{i-k}$이면 RA_i와 RA_{i-k}은 연관관계가 있으며, RA_i는 RA_{i-k}의 하위노드가 된다.*

속성 RA_i, RA_{i-k}의 원소들은 상품 아이디로 구성되어 있으며, 상품 아이디의 개수가 많은 속성일수록 더 많은 상품의 공통속성이 된다. 따라서 공통속성이 많은 속성일수록 계층구조의 상단에 위치한다. □

본 알고리즘은 속성간의 포함관계를 얼마나 빠르게 찾을 것이냐, 속성간의 동치 집합을 얼마나 효율적으로 통합하느냐에 따라 성능이 좌우된다. 따라서 본 논문에서는 속성집합은 하나로 통합하고, 속성간의 포함관계를 찾는데 소요되는 연산시간을 줄이기 위해, 속성집합의 원소를 이용하여 계산할 수 있는 3개의 함수(Count, Sum, Max)를 이용하는 방안을 제시한다. 상품 아이디를 이용하여 연산 가능한 Count, Sum, Max가 그 자체만으로는 의미가 없지만, 속성집합 간에 동치를 이룰 수 있는지를 파악하고, 속성간의 포함관계를 찾는 연산에서, 포함관계를 형성할 수 있는 속성을 찾는데 주요한 판단기준으로 사용한다. 예를 들어 속성 A={1, 3, 5}, B={1, 5}, C={1, 3, 5}, D={1, 2, 6}, E={2, 3, 6}일 때 A와 동치인 집합을 구하려면 A-B, A-C, A-D, A-E 간의 원소를 모두 비교하여 동치여부를 파악해야 한다. 집합의 개수가 작고, 원소의 개수가 적은 경우에는 별 문제 없겠지만, 집합과 원소의 개수가 증가하면 할수록 더 많은 연산시간이 필요 하다. 소요되는 연산시간의 대부분은, 연산을 할 필요가 없는 것 까지도 연산을 함으로써 발생한다. 따라서 본 논문에서는 속성집합의 원소를 비교를 할 필요가 없는 속성들을 걸러내어, 불필요한 연산시간을 줄이고자 한다. 앞의 예 에서 3개의 함수를 사용하여 A와 동치를 이루는 집합을 구하는 과정은 다음과 같다.

① 각 집합별로 원소의 개수, 합, 최대값을 구한다.
$COUNT_A = 3$, $SUM_A = 9$, $MAX_A = 5$
$COUNT_B = 2$, $SUM_B = 6$, $MAX_B = 5$
$COUNT_C = 3$, $SUM_C = 9$, $MAX_C = 5$
$COUNT_D = 3$, $SUM_D = 9$, $MAX_D = 6$
$COUNT_E = 3$, $SUM_E = 11$, $MAX_E = 6$ 이다.

② A와 동치를 이루기 위한 집합은 각각의 집합을 구성하는 원소의 개수는 동일해야한다. 따라서 $COUNT_A = 3$이므로 원소의 개수가 3인 집합만이 동치집합을 이룰 수 있는 후보 집합이 된다. ②에 의해 생성된 후보 집합은 {C, D, E}이다.

③ $SUM_A = 9$이므로 원소의 합이 9인 집합이 동치를 이룰 수 있다. 따라서 후보 집합 중 $SUM_E = 11$이므로 동치를 이룰 수 없어 후보 집합에서 제거 되어 후보 집합은 {C, D}로 줄어든다.

④ $MAX_A = 5$ 이므로 각 원소의 최대값은 5이어야 하는데, $MAX_D = 6$이므로 후보 집합에서 제거되어 결국 후보 집합은 {C}가 된다. 따라서 A와 C는 동치를 이룰 가능성이 매우 크며 정확히 동치집합인지를 파악하기 위해 집합의 원소를 비교하면 된다. 물론 속성간

의 포함관계를 찾는 경우에도 동일한 방법을 적용하면 된다. 이렇게 하면 모든 원소를 비교하지 않고도 동치나 포함관계를 이룰 수 있는 속성집합만을 비교하기 때문에 충분히 연산시간을 줄일 수 있다.

알고리즘에 대한 기술은, 상품집합 P = {P_1, P_2, P_3, P_4}와 상품속성 집합 A = {A_1, A_2, A_3, A_4 ,A_5, A_6, A_7, A_8}로 구성된, 다음의 상품 데이터를 사용하여 자세히 설명하고자 한다.

P_1 = {A_1, A_3, A_4}

P_2 = {A_1, A_2, A_5, A_8}

P_3 = {A_1, A_3, A_7}

P_4 = {A_1, A_2, A_6, A_8}

3.2 상품속성 분류

알고리즘의 첫 번째 단계로 상품 속성을 중심으로 속성이 속해있는 상품의 Pid들을 집합으로 묶어냄으로서 상품속성별로 분류하는 과정이다.

집합 Z = {1, 2, 3, 4}일때 n(Z)은 집합을 구성하는 원소의 총 개수로 정의하며, ∑(Z)은 Z를 구성하는 원소들의 합이라 정의하고, Max(Z)은 집합의 원소들 중 최대값이라 정의한다. 따라서 집합 CA_n의 CntOfPid, SumOfPid, MaxOfPid를 정의하면 다음과 같다.

CntOfPid = $n(CA_n)$

SumOfPid = $\sum(CA_n)$

MaxOfPid = $Max(CA_n)$

(그림 2)의 상품속성별 분류 과정은 CE_i의 상품이 상품테이블(GPID)에 등록되어 있지 않으면, GPID에 상품명과, 고유한 상품아이디(Pid)를 부여하여 저장하고, 상품이 저장되어 있다면 저장된 상품의 Pid을 구한다. 속성별 분류는 CE_i의 A_{jk}가 속성그룹테이블(GCA)에 등록되어 있으면, 등록된 CA_n의 마지막에 Pid를 추가하고, CntOfPid, SumOfPid, MaxOfPid를 재계산하여 수정한다. 만일 존재하지 않는다면 CA_n에 A_{jk}를 등록하고, 3개의 함수에 초기 값을 할당하고, $CA_n(Pid_{jk})$에 Pid를 추가하여 CA_n을 생성한 후 GCA에 등록한다. 하나의 속성집합 CA_n는 속성(A_i)과 $CntOfPid_i$, $SumOfPid_i$, $MaxOfPid_i$, 속성을 포함하는 상품아이디(Pid_{jk})로 구성된다. 따라서 전체 상품집합을 순차적으로 한번만 읽어서, 모든 상품을 상품의 고유아이디로 변환하고, 속성별로 상품을 분류한다.

본 논문에서는 상품명을 직접 사용 하는 것이 아니라, 상품명을 자연수의 고유한 상품아이디로 변경하여 사용함으로써, 속성집합을 통합하고 속성간의 포함관계를 효율적으로 찾기 위해, 제시한 3개의 함수(CntOfPid, SumOfPid, MaxOfPid)를 효율적으로 계산하고 사용할 수가 있다. 속성집합의 원소로 분류된 상품정보가 아이디로 관리됨으로써, 속성간의 원소들을 비교 연산할 경우에도 연산시간을 단축한다. 또한 속성집합의 원소들은 상품인데, 상품명을 문자 형태로 저장 하는 것이 아니라 자연수를 저장함으로써 메모리도 절약된다.

3.1의 예제에서 상품을 상품속성으로 분류하는 과정은 다음과 같다. 먼저 상품 P_1과 P_1의 첫 번째 속성 A_1을 읽은 후, <표 1>에서 ProductName이 P_1인 것을 찾는다. 찾은 결과 P_1이 <표 1>에 존재하지 않기 때문에 P_1을 ProductName으로 지정하고, Pid는 1로 하여 <표 1>에 저장한다. Pid가 할당이 되었으면, 다음으로 A_1이 <표 2>의 AttName에 존재하는지 확인한다. 확인 결과 A_1은 존재하지 않으므로, A_1을 AttName에 할당하고, Pid는 <표 1>에 등록된 1을 지정하고, CntOfPid는 1, SumOfPid는 1, MaxOfPid도 1로 할당한 후, <표 2>에 등록한다. 상품 P_1의 두 번째 속성 A_3을 읽어 <표 1>에서 P_1이 등록되어 있는지 확인한다. 확인결과 P_1은 이미 등록되어 있으므로, ProductName P_1의 Pid 1을 취한다. Pid를 구한 후 AttName A_3가 <표 2>에 등록되어 있는지 확인하면, A_3는 등록되어 있지 않으므로 AttName은 A_3

```
COMPUTE_GCA(GCE)
입력 : 상품집합(GCE)
출력 : 상품속성별로 분리된 속성집합(GCA)

GPID ← null, Pid ← 0
Do
  if CEi(Pi) not Exist in GPID then
     Pid ← Pid + 1
     GPIDm(Pi)    ← CEi(Pi)
     GPIDm(Pidj) ← Pid
  else
     Pid ← GPIDm(Pidj)
  end if

  if CEi(Ajk) not Exist in GCA then
     CAn(Ai)          ← CEi(Ajk)
     CAn(CntOfPidl)  ← 1
     CAn(SumOfPidl) ← Pid
     CAn(MaxOfPidl) ← Pid
     CAn(Pidjk)       ← Pid
     GCA ← GCA + CAn

  else
     n ← find(CAi(Ai)=CEi(Ajk))
     CAn(CntOfPidl)  ← CAn(CntOfPidl) + 1
     CAn(SumOfPidl) ← CAn(SumOfPidl) + Pid
     CAn(MaxOfPidl) ← Max(Pid,CAn(MaxOfPidl))
     CAn(Pidjk) ← CAn(Pidjk) + Pid
   end if
While(EOF of GCE)

     CE   : 한 개의 상품에 대한 집합
     GCE  : 상품 집합, CE 집합
     CA   : 한 개의 속성 집합
     GCA  : 속성 집합 , CA집합
     Pid  : 상품 고유 아이디
     GPID : 상품별 아이디 집합
     CntOfPid : CA의 원소를 구성하는 Pid의 개수
     SumOfPid : CA의 원소를 구성하는 Pid의 합
     MaxOfPid : CA의 원소를 구성하는 Pid의 최대값
```

(그림 2) 상품속성별 분류

<표 1> 상품 아이디

ProductName	Pid
P_1	1
P_2	2
P_3	3
P_4	4

〈표 2〉 상품속성별 분류

Att Name	CntOf Pid	SumOf Pid	MaxOf Pid	Pid	Pid	Pid	Pid
A_1	4	10	4	1	2	3	4
A_2	2	6	4	2	4		
A_3	2	4	3	1	3		
A_4	1	1	1	1			
A_5	1	2	2	2			
A_6	1	4	4	4			
A_7	1	3	3	3			
A_8	2	6	4	2	4		

〈표 3〉 정렬된 상품속성 집합

Att Name	CntOf Pid	SumOf Pid	MaxOf Pid	Pid	Pid	Pid	Pid
A_4	1	1	1	1			
A_5	1	2	2	2			
A_7	1	3	3	3			
A_6	1	4	4	4			
A_3	2	4	3	1	3		
A_2	2	6	4	2	4		
A_8	2	6	4	2	4		
A_1	4	10	4	1	2	3	4

으로 지정 하고, Pid는 <표 1>에 획득한 1을 지정하고, CntOfPid는 1, SumOfPid는 1, MaxOfPid도 1로 할당한 후, <표 2>에 등록한다. 속성이 이미 <표 2>에 등록되어 있는 경우, 즉 상품 P_3의 속성 A_3을 분류해보면, 상품 P_3은 <표 1>에 이미 등록되어 있으므로, Pid는 3을 획득하고, A_3도 이미 <표 2>에 존재하므로 A_3에 Pid 3을 추가 후, A_3에 포함되어 있는 모든 Pid을 이용하여 CntOfPid, SumOfPid, MaxOfPid를 계산한 결과 2, 4, 3로 수정하여 저장한다. 상품 P_1, P_2, P_3, P_4를 위와 같은 방법으로 모두 분류하면 <표 2>와 같다.

상품속성별로 분류된 집합을 3.3절 상품속성 집합 통합에서 효율적으로 사용하기 위해 속성집합을 구성하는 함수, CntOfPid, SumOfPid, MaxOfPid순으로 오름차순 정렬을 한다. 본 논문에서는 별도의 정렬 알고리즘을 구현하지 않고 일반적인 알고리즘 중 가장 빠르다는 퀵 소트 알고리즘을 그대로 적용한다.

<표 2>의 예제를 CntOfPid, SumOfPid, MaxOfPid순이 정렬하면 <표 3>과 같다.

3.3 동일한 상품속성 집합 통합

이 절에서는 속성집합을 구성하는 Pid들이 동일한 속성들은, 속성을 구성하는 원소는 동일하고, 속성명만 다르므로,

```
COMPUTE_RCA(GCA)
입력 : 상품속성별로 분리된 속성집합(GCA)
출력 : 중복된 속성을 제거한 속성집합(RCA)

DO
 if CAn(CntOfPid)  = CAn+1(CntOfPid) and
    CAn(SumOfPid) = CAn+1(SumOfPid) and
    CAn(MaxOfPid) = CAn+1(MaxOfPid) then
    if CAn(CntOfPid) =  1 then
       CAn(Ai) ← CAn(Ai) + CAn+1(Ai)
       remove(CAn+1)
    else
       compareAttGroup()  //집합의 원소 비교
    end if
 else
     RAi ← CAn
     RCA ← RCA + RAi
 end if
While(EOF of GCA)

       GCA  : 정렬된 속성집합
       RCA  : Pid가 같을 경우 통합된 속성집합
```

(그림 3) 동일한 상품속성 집합 통합

하나의 속성집합으로 통합할 수 있다. 통합하는 과정은 (그림 3)과 같다.

속성집합 CA_n와 CA_{n+1}이 동치집합이 되기 위해서는

① $n(CA_n) = n(CA_{n+1})$

② $CA_n \subset CA_{n+1} \wedge CA_n \supset CA_{n-1}$

③ $\forall CA_n \in CA_{n-1}$ 이어야 한다.

본 논문에서 제시한 CntOfPid, SumOfPid, MaxOfPid을 이용하여 CA_n와 CA_{n+1}이 동치가 될 수 있는 속성의 조건은

$CA_n(CntOfPid) = CA_{n+1}(CntOfPid) \wedge$

$CA_n(SumOfPid) = CA_{n+1}(SumOfPid) \wedge$

$CA_n(MaxOfPid) = CA_{n+1}(MaxOfPid)$이다.

모든 속성집합은 3.2절에서 CntOfPid, SumOfPid, MaxOfPid 순으로 오름차순으로 정렬이 되어 있다. 따라서 비슷하거나, 같은 속성들은 인접하여 위치하고 있기 때문에, 속성간의 동치여부는 바로 인접한 속성끼리만 비교하도록 한다. 인접한 두 속성집합의 동치 여부를 파악하기 위해서 속성의 모든 원소들을 전부 비교하지 않고, CntOfPid, SumOfPid, MaxOfPid가 동일한 속성인 경우에만 원소들을 비교함으로써 불필요한 비교 연산시간을 제거한다. 만일 CntOfPid, SumOfPid, MaxOfPid 중 어느 하나라도 일치하지 않으면, 두 속성은 동치관계를 형성하지 않는다. 왜냐하면 CntOfPid, SumOfPid, MaxOfPid는 속성을 포함하는 Pid을 이용하여 계산된 값들이므로, 속성을 구성하는 Pid가 다르면 원소의 개수, 합, 최대값 중 하나이상은 무조건 달라지기 때문에 절대 동치를 이룰 수 없다.

(그림 3)에서 두 속성집합 CA_n와 CA_{n+1}의 CntOfPid가 1이고 SumOfPid가 동일하면 두 속성은 동일한 집합이다. 두 속성이 가지고 있는 SumOfPid은 각 상품의 고유 아이디의 합 즉 Pid들의 합이고, 상품의 Pid가 같으면, 동일 상품의 특성에 포함되는 속성이다. 따라서 CntOfPid가 1인 속성들은 단 1회의 비교연산으로 동치집합 여부를 확인할 수 있다. CntOfPid가 2이상인 경우에는 CA_n와 CA_{n-1}를 구성하는 Pid는

〈표 4〉 통합된 상품속성 집합

ROW ID	Att Name	CntOf Pid	SumOf Pid	MaxOf Pid	Pid	Pid	Pid
1	A_4	1	1	1	1		
2	A_5	1	2	2	2		
3	A_7	1	3	3	3		
4	A_6	1	4	4	4		
5	A_3	2	4	3	1	3	
6	A_2	2	6	4	2	4	
7	A_1	4	10	4	1	2	3

달라도 CntOfPid, SumOfPid, MaxOfPid는 같을 수 있어, 동일한 값을 가진다고 하더라도 반드시 동치 집합을 형성하지는 않는다. 따라서 속성간의 Pid을 비교하는 연산(compareAttGroup)이 필요하다. Pid를 비교연산 할 경우에도 속성의 모든 Pid을 비교 하는 것이 아니라, 동치집합의 정의를 이용한다. 동치 집합의 정의에 의하면 두 집합이 동치를 이루기 위해서는 $CA_n = CA_{n-1}$이다. 따라서 CA_n의 원소 중 하나라도 CA_{n+1}에 존재하지 않는다면 동치를 형성할 수 없으므로, CA_{n+1}과의 비교 연산을 종료하면 되고, CA_{n+1}에서 CA_n의 Pid를 찾았다면 더 이상 비교연산을 할 필요 없이 CA_n의 다음 Pid를 비교하면, Pid 전체를 비교 하는 것 보다 연산 시간을 획기적으로 줄일 수 있다.

<표 3> 에서 A_4, A_5, A_7, A_6은 CntOfPid가 1이므로 SumOfPid가 같다면 AttName은 하나로 통합할 수 있다. 다음으로 CntOfPid가 2이상인 속성 중 CntOfPid, SumOfPid, MaxOfPid가 동일한 AttName들의 원소들을 비교 하면, A_2와 A_8는 동치 집합을 형성하므로, 두 AttName을 하나의 AttName(A_2)으로 통합할 수 있다. 이런 과정을 모든 AttName에 순차적으로 적용하여 정리하면 <표 4>와 같다.

3.4 상품 속성간의 포함관계 계산

본 장에서는 속성들 간의 내포되어 있는 포함관계를 효율적으로 찾아내는 과정이다. 포함관계를 찾기 위한 알고리즘은 (그림 4)와 같다.

두 속성집합 $RA_i \subset RA_{i+k}$ 이고 $RA_i \neq RA_{i+k}$이면 RA_i는 RA_{i+k}의 진부분집합이다. 따라서 RA_{i+k}은 RA_i의 모든 원소를 포함한다. 그러므로 RA_i 가 RA_{i+k}에 포함되기 위해서 $RA_i \subsetneq RA_{i+k}$가 되어야 한다.

속성 RA_i가 RA_{i+k}에 포함될 가능성이 없는 경우는

① RA_i(CntOfPid) > = RA_{i+k}(CntOfPid)

② RA_i(SumOfPid) > = RA_{i+k}(SumOfPid)

③ RA_i(MaxOfPid) > RA_{i+k}(MaxOfPid) 이다. 즉 3개의 경우 중 하나라도 해당 되면 RA_i는 RA_{i+k}에 절대로 포함되지 않는다.

속성 RA_i의 부모노드가 되기 위한 후보집합 CANDP는 다음과 같이 정의한다.

CANDP = *{RA_k | RA_k는 RCA의 k번째 RA, RA_i와 포함관계를 형성할 가능성이 있는 속성}*

CANDP가 되기 위한 RA_k는 다음과 같다.

① RA_i(CntOfPid) < RA_k(CntOfPid)

② RA_i(SumOfPid) < RA_k(SumOfPid)

③ RA_i(MaxOfPid) < = RA_k(MaxOfPid)

즉 RA_k의 원소의 개수는 RA_i 보다 커야하고, RA_k의 원소들의 합은 RA_i의 원소들의 합보다 커야하며, RA_k의 원소의 최대값은 RA_i의 원소의 최대값 보다 크거나 같아야 한다.

두 속성 RA_i가 RA_{i+k}에 포함되기 위한 조건은 반드시 후보집합에 포함되고, $\forall RA_i \in RA_k$이다. 따라서 RA_i의 모든 원소는 반드시 RA_k에 존재해야 한다.

속성 RA_i가 포함될 수 있는 부모 노드를 찾는 방법은 RA_{i+1} 부터 RA_n 까지 RA_i가 포함될 수 있는 RA_{i+k}을 찾으면 된다. 이런 방법으로 하면 자료가 적은 경우에는 별 차이가 없겠으나 자료가 증가하면 연산횟수는 $a(a-1)^2*P^2$(a:속성수, p:상품수)이다. 연산횟수의 대부분은 포함관계를 형성할 수 없는 속성들을 비교하는데 소요된다. 따라서 (그림 4)에서는 불필요한 연산을 제거하기 위해 CntOfPid, SumOfPid, MaxOfPid을 이용하여, RA_i와 포함관계를 형성할 가능성이 있는 속성들을 추출하여, 후보집합 CANDP를 생성한다. RA_i의 부모노드를 찾는 방법은 RA_i의 Pid와 후보집합에 속한 속성들의 Pid들을 비교하여 최종적으로 포함관계를 찾아낸다. 후보집합을 사용함으로써, 포함관계를 형성할 수 없는 속성과는 비교 연산을 수행하지 않기 때문에 소요되는 연산시간을 획기적으로 줄인다.

<표 4>의 예제에서 AttName A_4의 부모노드를 찾기 위해 제일 먼저 A_4의 CntOfPid+1, SumOfPid+1, MaxOfPid의 값 2, 2, 1을 조건으로 하여 후보집합을 생성한다. 생성된 후

```
COMPUTE_PCR(RCA)
입력 : 중복된 속성을 제거한 속성집합(RCA)
출력 : 부모-자식관계가 정의된 속성집합(PCR)

n ← 0, findRslt ← null
For (i = 1 to RCA.length)
  for (k = i + 1 to RCA.length )
     RAi(CntOfPid) <  RAk(CntOfPid) and
     RAi(SumOfPid) <  RAk(SumOfPid) and
     RAi(MaxOfPid) < = RAk(MaxOfPid) then
     CANDP ← CANDP + RAk
  next
  findRslt ← findParent(CANDP, RAi)
  if findRslt = True then
     PRi[Cname] ← RAi[name]
     PRi[Pname] ← findRslt
     PRi[Ccode]  ← curCode
     PRi[Pcode]  ← parentCode
  end if
  PCR  ← PCR + PRi
Next

     RCA : Pid가 같을 경우 통합된 속성집합
     PCR : 포함관계(부모-자식) 테이블
     CANDP : 부모 노드가 될 수 있는 후보 속성 집합
```

(그림 4) 상품속성별 포함관계 연산

보집합 CANDP = {A_3, A_2 ,A_1}이다. 후보집합이 생성된 후 A_4와 후보집합에 속한 속성들의 Pid를 차례대로 비교하여 포함관계를 찾는다. 비교 연산결과 A_4의 모든 Pid들이 A_3의 Pid들에 존재하므로, AttName A_3는 A_1의 부모노드가 된다. 부모노드를 찾으면 <표 5>에 A_4의 AttName과 ROWID를 CNode와 CCode로 지정하고, A_3의 AttName과 ROWID를 PNode와 PCode로 등록한 후, A_4의 부모노드를 찾는 연산을 종료한다. 만일 A_4와 후보집합의 첫번째 속성과 포함관계를 형성하지 못하면, 후보 집합의 다음 속성 A_2와 비교연산 한다. 위와 같은 방법으로 모든 속성의 Pid를 비교하면 부모노드를 반드시 찾게 된다. A_1의 부모노드를 찾으면 후보집합으로 부터 부모노드를 찾는 연산을 중지하고, 다음 AttName A_5의 부모노드를 찾는다. 동일한 방법으로 마지막 AttName까지 부모 노드를 찾는 연산을 수행하면 모든 속성은 자신의 부모 노드를 가지게 된다.

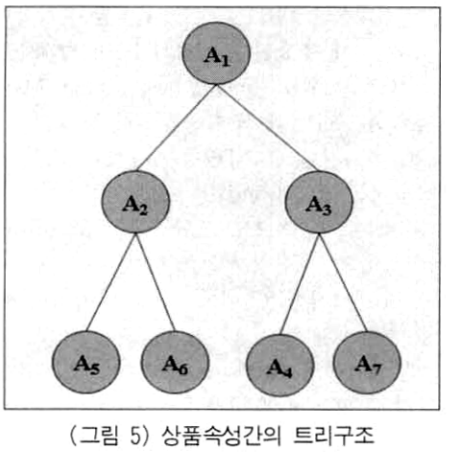
3.5 트리 생성

마지막 단계로 3.4에서 생성한 부모-자식의 관계를 이용하여 계층구조로 표현한다. 계층구조는 일반적으로 사용하는 트리 알고리즘을 적용하여 표현한다.

<표 5>의 자식노드 코드 CCode와 부모노드 코드 PCode을 이용하여 트리구조를 생성하면 (그림 5)와 같이 상품속성별 계층트리 구조를 이룬다.

〈표 5〉 부모-자식 노드

CNode	PNode	CCode(ROWID)	PCode(ROWID)
A_4	A_3	1	5
A_5	A_2	2	6
A_7	A_3	3	5
A_6	A_2	4	6
A_3	A_1	5	7
A_2	A_1	6	7
A_1	A_1	7	7



(그림 5) 상품속성간의 트리구조

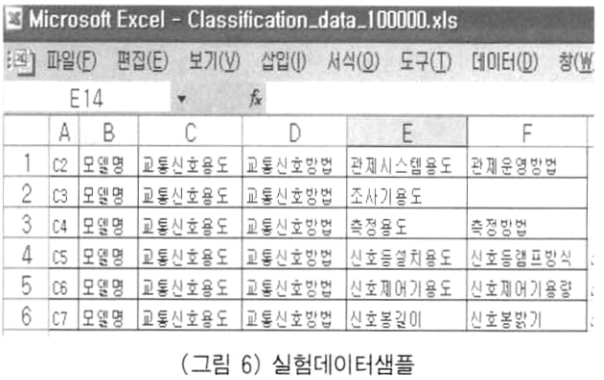
4. 실험

본 논문에서는 무작위로 작성한 총 100,000건의 상품 데이터를 24M 바이트 액셀 파일의 형태로 저장하여 실험 데이터로 사용 하였다. 한 개의 상품은 평균 12개의 속성을 가지고 있으며, 상품속성 당 평균 크기는 한글 12자, 즉 24바이트 가량이다. 상품 당 속성의 개수는 최대 16개, 최소 8개로 다양한 종류의 상품을 다루고 있는 데이터이다[26]. 실험은 펜티엄4 2.1GHz의 CPU와 1G 바이트 메모리를 갖춘 PC 환경에서 JDK1.4로 구현하여 실행하였다. (그림 6)은 실험에 사용한 데이터의 일부 표본이다.

실험은 10만건의 상품 데이터로부터 무작위로 1만건 단위의 상품 데이터를 추출하여 각 상품건수별로 3회씩 총 30회 실행하였다. 데이터 파일을 로딩하는 시점에서부터 최종 결과물이 생성될 때까지의 연산시간을 측정하였으며 그 결과는 <표 6>과 같다.

실험결과 생성되는 분류체계는 (그림 7)과 같이 트리구조의 분류체계를 생성한다. (그림 8)은 연산시간의 평균값을 상품 데이터 건수별로 그래프로 표현한 것이다.

본 논문에서 제안하는 방식의 우수성을 입증하기 위하여 상품의 모든 속성들의 조합을 전부 비교하는 단순 비교 방식으로도 실험을 실행하여 비교하려 시도 하였으나, 너무 많은 시간이 소요되어 결과를 생성하는데 실패하였다. 따라


Microsoft Excel - Classification_data_100000.xls

파일(F) 편집(E) 보기(V) 삽입(I) 서식(O) 도구(T) 데이터(D) 창(W

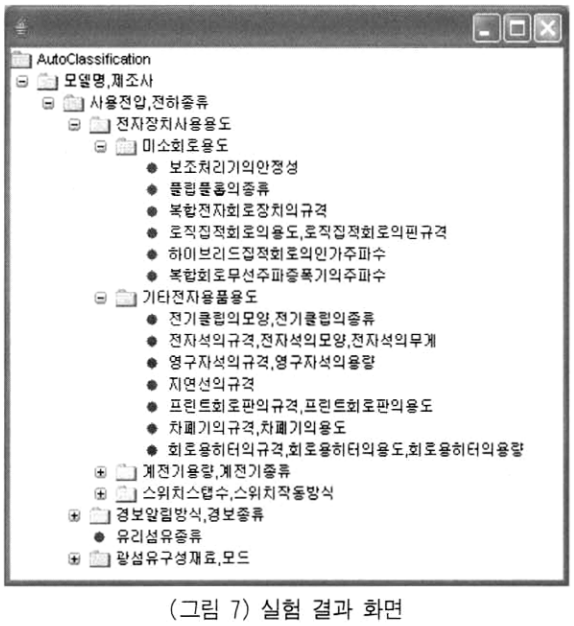
E14

	A	B	C	D	E	F
1	C2	모델명	교통신호용도	교통신호방법	관제시스템용도	관제운영방법
2	C3	모델명	교통신호용도	교통신호방법	조사기용도	
3	C4	모델명	교통신호용도	교통신호방법	측정용도	측정방법
4	C5	모델명	교통신호용도	교통신호방법	신호등설치용도	신호등램프방식
5	C6	모델명	교통신호용도	교통신호방법	신호제어기용도	신호제어기용량
6	C7	모델명	교통신호용도	교통신호방법	신호봉길이	신호봉밝기

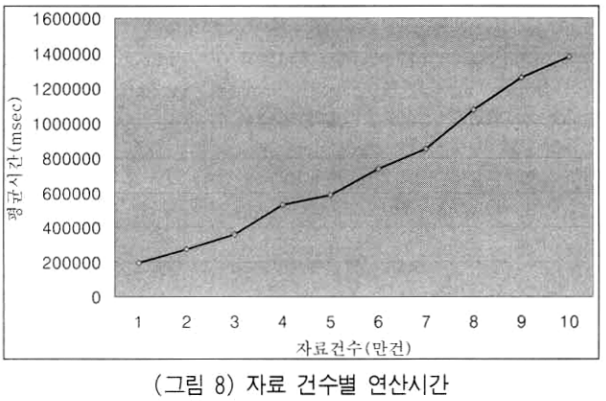
(그림 6) 실험데이터샘플

〈표 6〉 자료 건수별 연산시간

건수	1차(msec)	2차(msec)	3차(msec)	평균
10000	195156	197092	194202	195,483
20000	267031	280390	270609	272,677
30000	358000	340205	376589	358,265
40000	521109	530101	541863	531,024
50000	585015	591028	587821	587,955
60000	738610	730190	742094	736,965
70000	854328	832093	869894	852,105
80000	1076078	1056778	1098783	1,077,213
90000	1261282	1249835	1272375	1,261,164
100000	1380375	1360892	1399848	1,380,372



(그림 7) 실험 결과 화면



(그림 8) 자료 건수별 연산시간

〈표 7〉 단순비교방법과 제안알고리즘의 연산시간

처리단계	단순비교	제안알고리즘
상품정보로딩	$p * a$	$p * a$
상품속성분류	$p(p-1) * a$	$p * a$
상품속성소팅	$a * log\ a$	$a * log\ a$
상품속성통합	$p(p-1) * a^2$	$log\ p * 2a$
속성포함관계	$p(p-1) * a^2$	$(p*log\ p)*(a*log\ a)$

서 이를 분석적으로 비교하고자 한다. 상품수를 p 상품속성수를 a 라 할 때 상품속성간의 포함관계 연산을 위한 단순비교방법과 제안된 알고리즘간의 비교연산회수를 계산해보면 <표 7>과 같다.

<표 7>에 의해 100,000건의 상품정보를 처리하기 위한 연산시간을 계산하면 단순 비교에 의한 방법은 91,999시간+이 소요되고 제안된 알고리즘에 의해서는 0.41시간++이 소요되어 단순비교방법보다 222,389배의 성능향상을 보임을 알 수 있다.

본 알고리즘은 상품을 속성으로 분리하여 속성간의 포함관계를 찾는 구조이며, 모든 상품은 최소한 하나 이상의 공통속성을 가지게 되므로, 속성으로 분류하여 트리구조로 표현하면 반드시 하나의 루트노드가 존재한다. 또한 속성의 다양화 및 복잡화로 인하여 트리의 깊이가 깊어지는 한이 있더라도, 모든 속성은 반드시 부모-자식 관계를 형성하며, 부모노드에 있는 속성은 자식노드에 있는 속성을 포함하는 계층구조의 트리를 생성한다.

본 연구는 실험을 통하여 상품속성 정보를 기반 정보로 활용함으로서 대량의 상품에 대한 분류체계를 적절한 시간내에 자동으로 생성할 수 있음을 보였다. 그러나 제안된 알고리즘은 가장 하위의 상품속성에서 시작하여 자신과 포함관계를 형성하는 상위의 상품속성을 찾으면 즉시 부모 노드로 설정하는 형태로 구현되어 있어 결과적으로 분류체계는 깊이우선순위 트리구조의 형태로 생성된다는 문제가 있다. 또한 상품속성간의 포함관계를 찾는데 소요되는 연산시간이 전체의 89%을 차지하므로, 이 부분을 개선한다면 연산시간을 획기적으로 단축할 수 있을 것으로 기대된다.

5. 결론 및 향후 연구

제시된 자동분류체계 생성 알고리즘은 새로운 분류체계를 생성하거나 기존의 분류체계를 상품의 속성 정보를 이용하여 새로운 분류체계로 마이그레이션할 경우 유용하게 사용할 수 있다. 대부분의 온라인 판매업체에서 취급하는 상품건수는 100,000건 이하이나 증가추세에 있으며, 급격히 성장하고 있는 오픈마켓일 경우 거래되는 상품건수가 빠르게 증가(100만건)하고 있고, 상품의 교체주기도 상당히 짧다[27]. 실질적으로 100만건 이상이 되는 상품을 수작업으로 파악하고 분류하여 분류체계를 생성하기에는 너무 많은 시간이 소요되며 관리도 불가능하다. (그림 8)에서 보듯이 현재 10만건의 상품자료를 연산하는데 소요되는 시간은 자료의 건수에 비례하여 선형증가모형을 이룸으로써 상품건수에 따른 분류체계를 생성하는 시간을 예측할 수 있으며 100만 건 단위의 상품수라 할 경우에도 분류체계 생성시간을 충분히 예측하고 자동으로 생성하는데 의의가 있다.

본 논문에서는 상품과 상품에 대한 속성만으로 각 상품들간에 계층구조를 생성할 수 있음을 확인하였다. 그러나 현 알고리즘은 단말 노드에서 검색을 시작하여 자신의 부모노드를 찾으면 검색을 중단하는 형식으로 되어 있어 깊이 우선 트리구조의 분류체계를 생성한다. 따라서 상품의 구조가 복잡하고 건수가 많아질수록 트리의 깊이가 깊어지는 특성을 가진다. 그러나 현실적으로 트리의 깊이가 4단계를 초과하는 경우는 실용적이지 못하다. 예를 들어 UNSPSC와 같은 분류체계는 4단계로 이루어져 있고 일반적으로 자체 분류체계를 사용하는 경우에도 대부분 3~4단계의 구조로 되

+ $p(p-1)*a^2$ = 100,000*99,999*144 = 1,439,985,600,000회
= 1,439,985,600,000회 * 0.23msec = 91,999시간

++ plogp*aloga = 100,000*5*12*1.07 = 6,475,087회
= 6,475,087회* 0.23msec = 0.41시간

어있다. 그러므로 분류체계를 적정한 수준에서 조정하는 작업이 필요하게 되며, 따라서 향후에는 자동 분류 시스템이 제시한 트리구조에 사용자가 트리의 깊이나 부모 자식간의 관계를 조정하는 작업을 하여 원하는 형태의 트리를 생성할 수 있도록 하여야 한다. 자동으로 생성된 트리가 사용자의 의도와 일치하지 않거나 조정이 필요하게 되므로 이동, 삭제, 변경, 추가 등의 사용자 오퍼레이션이 요구 되며 이에 대한 연구가 추가적으로 필요하다. 더불어서 현 방식에서는 상품 속성들 간의 포함관계를 계산하는 연산시간이 전체시간의 89%를 차지하는데 이를 단축할 수 있는 방안이 고려되어야 한다. 또한 오퍼레이션에 의해 변형된 트리는 모두 동일한 계층구조를 가진 트리임을 증명하고, 최적화된 분류체계 트리를 바탕으로 분류체계 관리를 위한 데이터베이스 스키마를 자동으로 생성하여[13] 데이터베이스 전문가가 아닌 일반사용자가 분류체계 생성 및 관리에 대한 모든 업무를 할 수 있도록 하는 연구를 추가적으로 수행해야 한다.

참 고 문 헌

[1] Amy Roger, "Tool to Manage E-Catalog Content," Computer Reseller NEWS, Apr 5, 1999.

[2] 김건오, "검색엔진과 분류," 경영과컴퓨터 2005년 3월호, 2005.

[3] e비지니스정보센터, www.kebic.or.kr.

[4] UNSPSC, "United Nations Standard Products and Services Classification," White paper, http://www.unspsc.com/, 2001.

[5] NAICS, "North American Industry Classification System", http://www.census.gov/epcd/www/naics.html.

[6] EAN/UCC, "Europian Article Number/Uniform Code Council," http://www.ean-int.org/.

[7] UPC, "Universal Product Code," http://en.wikipedia.org/wiki/Universal_Product_Code.

[8] HS, "Harmonized Commodity Description and Coding System," \http://sunsite.icm.edu.pl/untpdc/eto/standards/hs/index.html

[9] 김학명, "전자상거래와 환경 비즈니스," 환경기술정보지 제 13호, 2002.

[10] Ajit Patankar and Arie Segev, "An Extensible Catalog Management Framework," CITM working paper:02-WP1051, September 2002.

[11] Christoph Quix, Mareike Schoop, Manfred Jeusfeld, "Business Data Management for Business-to-Business Electronic. Commerce," ACM SIGMOD 2002, volume 31, 49-54, 2002.

[12] 정지혜, "전자상거래를 위한 확장된 디지탈 카 탈로그 및 질의 모델 제안," 서울대학교 대학원, 학위논문(석사) 2000.

[13] Dongkye Kim, Sang-goo Lee, Jonghoon Chun, Sangwook Park and Jaeyoung Oh, "Catalog Management in E-Commerce System," proc. of Computer Science & Technology, 2003.

[14] Ben Wolin, "Automatic Classification in Product Catalogs," ACM SIGIR '02. August 11-15, 2002.

[15] 김성환, 김현철, 이태희, 이상구, "Extending Data Model of Naive-Bayesian Classifier in e-Catalog Classification," 한국컴퓨터종합학술대회 2005 논문집, Vol.32, No.1(B) 2005.

[16] Young-gon Kim, Tachee Lee, Jonghoon Chun, Sang-goo Lee, "Modified Naive Bayes Classifier for E-Catalog Classification," DEECS 2006, LNCS 4055, pp.246-257, 2006.

[17] UNDP, "United Nations Development Programme," http://www.undp.org/

[18] Domenico Beneventano and Stefania Magnani, "A framework for the classification and the reclassification of electronic catalogs," ACM SAC'04, March 2004.

[19] Dongkye Kim, Sang-goo Lee, Jonghoon Chun, Juhnyoung Lee, "A Semantic Classification Modelfor e-Catalogs," IEEE International Conference on E-Commerce Technology 85-92, 2004.

[20] Fensel, Omelayenko, Ding, Schulten, Botquin, Brown, Hlett, "Product Data Integration in B2B E-Commerce," IEEE Intelligence System, 2001.

[21] Dongkye Kim, Sang-goo Lee, Junho Shim, Jonghoon Chun, Zoonky Lee, and Heungsun Park, "Practical Ontology System for Enterprise Application," ASIAN 2005, LNCS 3818, pp.79-89, 2005.

[22] 이상구, "Design & Implementation of an e-Catalog Management System," DASFAA 2004 Tutorial, 2004.

[23] 김동규, 이상구, 최동훈, 전종훈, "전자 카탈로그를 위한 의미적 분류 모형," 정보과학회논문지: 데이터베이스 33권 1호, 102-116, 2006.

[24] 김동규, 이상구, 최동훈, "상품 데이터베이스의 동적 특성을 지원하는 분류모형", 정보처리학회 논문지 제12-D권 제1호, 165-178, 2005.

[25] Y.Ding, M.Korotkiy, B. Omelayenko,V.Kartseva, V. Zykov, M.Kelein, E.schulten, andD.Fensel, "GoldenBullet: A Automated Classification of Product Data in E-commerce," Withold Abramowicz(ed), Business Information System, Proceedings of BIS2002, Poznan, Poland, 2002.

[26] 상품별 속성정보, http://ghost.mju.ac.kr, 2006.

[27] 정한민, 김평, 성원경, "전자상거래 검색 기술동향," ITFIND, 주간기술동향, 1273호, 2006.

장 두 석

e-mail : dsjang@mju.ac.kr

1995년 한국방송대학교 전자계산학과 (학사)

2000년 경기대학교산업정보대학원 전자계산학과 (이학석사)

2005년 명지대학교 대학원 컴퓨터공학과 (박사과정수료)

1994년~현 재 (주)이건창호시스템

관심분야 : 전자 카탈로그, 데이터베이스, CRM

전 종 훈

e-mail : jchun@mju.ac.kr

1986년 University of Denver 전산과학과 (학사)

1992년 Northwestern University 전산과학과 (석사, 박사)

현재 명지대학교 컴퓨터공학과 교수

2001년~현 재 ㈜프람트 대표이사 사장

관심분야 : 전자상거래, 의료정보, CRM, 디지털 라이브러리