사용자 요구에 기반한 맞춤형 분류체계 생성기법 구현

Implementation of an User-guided Classification Tailoring System

장두석(Duseok Jang)*, 전종훈(Jonghoon Chun)**

초 록

분류체계가 현업에 유용하게 사용되기 위해서는, 다양한 특성을 가진 기업체나 조직의 사용목적에 적합하도록 만들어져야 한다. 분류체계 생성과정을 자동화함으로써 분류체계 시스템의 운용의 효율성과 편의성은 향상될 수 있으나, 실질적으로 업무에 적용하기 위해서는 분류체계 생성 단계에서부터 사용자가 적극적으로 개입하여 요구사항을 반영할 수 있어야 한다. 본 연구에서 제안하는 분류체계 생성 알고리즘은 사용자가 원하는 분류단계를 입력받아 이에 맞는 분류체계를 맞춤형으로 생성한다. 또한, 일차적으로 생성한 분류체계를 사용자가 원하는 형태로 변환할 수 있도록 분류항목을 조작하는 연산자를 제안하고 구현한다.

ABSTRACT

In order for a classification system to be useful in every day business environments, it has to reflect peculiar characteristics of enterprises and organizations. By automating the generation process of classification, the overall operation of classification system becomes effective and efficient. However, in order for it to be really useful, human input and interventions are needed from the very beginning phase of the classification generation. This paper proposes an user-guided classification generation algorithm, in which the user inputs a specific level of the classification that she/he wants. Also we propose and implement manipulation operators to tailor the initial classification into the shape that user wants.

키워드 : 상품속성, 분류체계, 분류체계 연산자, 사용자 지정, 맞춤형 분류체계, Product Attribute, Classification scheme, Classification Operation, user-guided, Tailoring Classification System

본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT연구센터 육성·지원사업(IITA-2006-C1090-0603-0031)의 연구결과로 수행되었음.

* 명지대학교 대학원 컴퓨터공학과
** 명지대학교 컴퓨터공학과 교수

1. 서 론

21세기 정보의 사회의 도래와 함께 전자상거래의 활성화로 인터넷은 사이버 공간상에서 각종 비지니스의 장이 되고 있으며, 점점 더 많은 영역으로 확대되고 있다[10]. 또한 최근에 급격한 증가 추세에 있는 오픈 마켓의 경우에도 취급하는 상품의 수가 적게는 수십만 건에서 수백만 건에 이르며 점점 더 증가 추세에 있다[5]. 이처럼 온라인상에서 전자적으로 거래되고 있는 엄청난 상품이나 콘텐트의 자료가 홍보되고 관리되기 위해서는, 효율적인 관리 체계가 필요하며, 사용자 입장에서는 쉽고 빠르게 원하는 정보를 찾을 수 있어야 한다. 상품이나 정보를 체계적으로 관리하고, 검색 할 수 있는 방법은 효율적인 분류체계의 생성 및 관리이다. 따라서 분류체계는 기업이나 조직의 목적 및 용도에 적합하도록 설계되고 생성하여 관리되어야만 한다. 이미 HS[18], UNSPSC[22], UPC[23], EAN/UCC[16]등 국제 표준 분류체계가 존재하나, 이들은 특정 목적에 맞도록 설계되고 분류되어 있어, 이용자의 사용목적에 맞지 않는 경우가 많다. 따라서 기업이나 조직은 각자의 사용목적에 적합한 고유한 분류체계를 생성하고자 한다. 분류체계 생성은 업무에 능통한 전문가에 의해 수작업으로 체계화 되어왔으나, 엄청난 상품의 수와 복잡하고 다양한 기능들의 추가로 인해 수작업에 의한 분류체계 생성은 점점 더 많은 시간과 비용을 필요로 한다. 따라서 이런 낭비적인 요소를 제거하여, 빠르고 쉽게 분류체계를 자동으로 생성하고자 하는 연구들이 많이 진행되었다. 그러나 자동으로 생성된 분류체계가 기업이나 조직의 사용목적에 적합하지 못하여 실제 업무에 적용하는 사례는 많지 않다. 또한 대부분의 자동화된 분류체계들이 자동 생성에 중점을 두고 연구되어 분류체계의 유연성이 떨어진다. 따라서 효율적인 분류체계를 구축하는 방법은 시스템에 의해 자동으로 생성된 분류체계를 기반으로 하여, 사용자가 개입하여 조정하는 방법을 병행하면, 빠른 시간 안에 사용목적에 적합한 분류체계를 생성할 수 있다.

이전 연구 [8, 9]에서는 상품을 상품속성집합으로 해석하여 상품속성들 간의 내포되어 있는 포함관계를 찾아 계층구조의 분류체계를 생성함을 보였다. 자동으로 생성된 분류체계의 특성은 단말노드의 속성에서 검색을 시작하여 자신과 포함관계를 형성하는 부모노들의 속성을 찾으면 검색을 중단하는 형태로 구현되어 있어, 상품의 수가 많아지고, 기능이 복잡하고 다양화될수록 분류단계가 깊어질 수 있다. 그러나 전자상거래 업체나 UNSPSC등 일반적으로 사용하는 분류체계는 보통 3~4단계로 구성되고 관리됨으로써 제안된 알고리즘에 의해 생성된 분류체계가 현실적이지 못할 수 있다. 또한 조직의 사용목적에 따라 분류단계가 달라질 수 있어, 특정 분류단계로 제한하여 생성 하는 것은 분류체계의 적용성을 떨어뜨린다. 그러므로 분류단계를 조직의 사용목적에 맞도록 조정할 수 있어야한다. 또한 시스템에 의해 자동으로 생성된 분류체계는 사용자의 의도와 반드시 일치 하는 것은 아니므로, 분류체계가 보다 현실적이고, 사용 용도에 적합해지기 위해서는 분류항목에 대한 조작기능(추가, 삭제, 이동 등)이 제공되어야만 한다. 따라서

본 연구는 사용자가 생성하고자 하는 분류 단계를 지정하여 분류체계를 생성하고, 자동으로 생성된 분류체계의 분류항목을 조작할 수 있는 기능을 제공한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 분류체계 구현방법, 관련연구, 이전연구 등 이론적 배경을 살펴보고, 제 3장에서는 알고리즘의 개요와, 사용자가 트리의 깊이를 조정하여 분류체계를 생성하는 알고리즘 및 자동으로 생성된 분류항목을 사용자가 조작할 수 있는 방법을 제안하며 제 4장에서는 알고리즘을 구현하여 실험한 결과를 기술하고 제 5장에서는 결론을 맺는다.

2. 이론적 배경

2.1 분류체계

체계적인 상품 관리를 위해서 효율적인 분류체계는 필수이다. 분류체계는 분류항목, 분류구조, 분류규칙으로 구성된다. 분류항목이란 분류하고자 하는 상품의 개념을 명명하는 말이며, 분류구조란 분류항목간의 계층적인 관계를 나타내는 말이며, 분리 규칙이란 상품이 어떤 분류항목에 포함되어야하는지에 대한 규칙을 이야기한다. 따라서 분류체계는 분류규칙과 분류구조가 통합된 의미이다. 분류체계를 생성하는 방법은 일반적으로 3가지로 구분된다[1].

1) 분류 전문가에 의한 생성: 상품에 대한 전문가적 의견을 가지고 분류체계를 생성함으로 분류체계의 목적, 용도 등을 명확히 정의할 수 있으며, 조직의 목표와 활용도에 최적화된 분류체계를 작성할 수 있다. 그러나 상품수가 방대할 경우 모든 상품에 대한 정보를 알 수 가 없으며, 따라서 각 영역의 전문가 집단을 구성하여 분류체계를 생성해야 하는데 쉽지 않는 작업이며, 수작업에 의한 분류체계 생성으로 오랜 작업시간을 필요로 한다. 이는 비용의 증가 원인이 된다. 또한 정확성을 검증할 방법도 없다. 더불어 분류체계를 생성하는 사람 또는 전문가 집단의 능력에 따라 다양한 분류체계가 생성될 수 있다.
2) 규칙에 의한 자동 생성: 일반적인 자동 생성방법으로는 통계적인 방법, 기계학습 방법을 많이 이용한다. 분류체계를 생성하는 시간이 수작업에 비교할 수 없을 정도로 빠르다. 그러나 자동으로 생성된 분류체계는 일정한 규칙에 의해 생성되므로, 조직의 정확한 목표나 사용처 등을 고려하지 않게 되어 조직의 필요성에 적합하지 못할 수 있다. 또한 자동으로 생성된 분류체계의 품질부분도 검증이 필요하다.
3) 국제/국내 표준으로 작성된 분류기준을 도입: 이미 국내 또는 국제적으로 통용되고 있는 분류체계를 도입하여 적용 하는 것이다. 이는 이미 검증되고 안정된 분류체계를 쉽고 빠르게 생성할 수 있다. 그러나 현실적으로 아직까지 분류체계를 매매하는 시장이 활성화 되지 못해 목적에 적합한 분류체계를 구하기가 쉽지 않다.

효율적인 분류체계 구축방법은 3가지 모

든 방법을 참조하여 최적의 방안을 찾는 것이다. 가장 현실적인 방안으로는, 시스템에 의해 자동으로 생성된 분류체계를 분류담당자가 검증하는 작업을 하면, 수작업에 의한 분류체계를 생성하는 시간 및 비용을 획기적으로 줄일 수 있다. 따라서 본 연구에서는 시스템에 의해 자동으로 생성된 분류체계를 기반으로 하여, 사용자에게 적합한 분류체계를 생성 하는 것이다.

2.2 관련 연구

통계적인 방법, 기계 학습적 방법, 분석적 방법 등을 이용하여 분류체계를 자동화하려는 다양한 연구들이 진행 되었다. 관련 연구로는 [25]는 나이브 베이시안 분류기를 확장하여, 사용자의 개입 없이 자동으로 유사한 카탈로그를 분류하는 방법을 제안하였으며, [13, 24]는 정보검색 기법과 기계학습방법을 이용하여 상품분류체계 생성하는 방법을 제안하였고, [12]는 전자카탈로그와 표준분류체계의 의미적 매핑기법을 제시하였으며, [11]은 벡터 스페이스 모델과 상품의 특징을 나타내는 단어들을 이용하여 상품분류체계를 자동으로 생성하였다. [2, 3, 15]는 모든 상품은 상품 속성 이면에 숨어 있는 의미를 분석해보면 상품의 속성들 간에는 구조적 관계가 형성하므로[20], 속성들 간의 의미적인 계층구조를 생성하고, 하나의 정의된 상품군은 의미적으로 여러 개의 다양한 분류군에 포함될 수 있다는 의미적 분류 모형을 제안하였다. [19]는 도서관 분류시스템에 근거한 문자 분류체계 생성 방안을 제시 하였으며, [5]는 사용자가 입력한 단어를 이용하여 사용자의 의도를 파악하여 계층구조의 의사결정 트리를 생성한다. [17]은 전자카탈로그 생성 후 개인에게 최적화된 뷰를 제공하는 방법에 대한 제안을 하였다. 또한 최근에는 온톨로지 기반의 분류체계에 대한 연구도 진행되었다. [14]는 온톨로지 기반의 상품 분류체계를 자동으로 생성하는 방법을 제안하였으며, [4]는 온톨로지 기반의 분류체계를 생성하고, 생성된 분류체계를 사용자가 조작할 수 있는 방법을 제시 하였다.

대부분의 연구들이 분류체계를 자동으로 생성하는데 역점을 두었으나 본 연구는 자동으로 생성된 분류체계가 기업체나 조직의 사용목적에 적합한 분류체계가 되도록 분류체계의 유연성을 제공하고자 한다.

2.3 이전 연구

모든 상품은 속성을 가지고 있다. 속성은 물건의 특징을 표현하고 다른 상품과 구별짓는 중요한 용어이다. 이처럼 상품을 구분하는 중요한 속성을 체계적으로 분류하여, 속성들 사이에 내포되어 있는 포함관계를 찾아내고 계층구조로 표현할 수 있다면 상품에 대한 분류체계를 자동으로 생성할 수 있다. 따라서 본 연구는 이런 관점에서 접근하여 분류체계를 자동으로 생성하고자 하는 연구를 진행 중이다. [7]에서는 상품속성을 이용하여 분류체계를 자동으로 생성하기위한 기법을 제안하였으며 [8]에서는 [7]에서 제안한 기법을 바탕으로 상품을 속성집합으로 분리하여, 속성간의 내포되어 있는 포함관계를 찾아서 계층구조의 분류체계를 자동으로 생성함을 보였다. 그러나 [7]에서 제시

한 방법으로 [8]에서 구현하여 실험한 결과 상품속성간의 포함관계를 찾아내는 부분의 연산시간이 전체 연산시간의 89%를 차지하였다. 이는 속성간의 포함관계를 찾는 부분이 [7]에서 제시한 알고리즘의 성능을 좌우하는 부분이며, 이 부분을 개선해야만 전체적인 성능향상을 가져올 수 있음을 알 수 있다. 따라서 [9]에서는 첫째로 속성간의 포함관계를 형성할 가능성이 없는 속성들을 배제하기 위한 함수들을 사용하였으며, 두개의 색인테이블을 정의하여 포함관계를 빠르게 찾은 연산을 수행하였고, AND조건 연산자를 사용하여 부모-자식 관계를 빠르게 찾음으로써 속성간의 포함관계를 찾는데 소요되는 시간을 획기적으로 개선하였다. [7, 8, 9]에 의해 자동으로 생성된 트리는 단말노드에서 검색을 시작하여 부모노드를 찾으면 검색을 중단하는 특징을 가지고 있어, 제안한 알고리즘으로 생성되는 트리구조의 분류체계는 깊이 우선 트리의 분류체계를 생성한다. 분류체계의 깊이는 조직의 규모, 분류체계 사용의 목적에 따라 달라지므로 어느 특정한 분류단계로 확정할 수 없다. 그러므로 자동으로 생성되는 분류체계의 깊이는 사용자가 결정할 수 있어야한다. 또한 분류단계에 맞는 분류체계를 자동으로 생성했다 하더라도, 생성된 분류체계가 반드시 사용자의 의도와 일치 하는 것은 아니므로, 사용자에게 적합한 분류체계를 생성하기 위해 분류항목에 대한 조작이 가능해야한다.

3. 맞춤형 분류체계 생성 기법

본 장에서는 [8, 9]의 깊이 우선 트리 생성의 문제점을 해결하기 위해, 알고리즘을 수정하고 추가하여, 사용자가 지정한 분류단계에서 분류체계를 생성할 수 있도록 한다. 3.1에서는 [8, 9]와 공통으로 사용된 알고리즘에 대해 간략히 언급하고, 3.2에서는 사용자가 지정한 분류단계에서 분류체계를 생성하는 방법을 제안하며, 3.3에서는 자동으로 생성된 분류체계를 사용자가 조작할 수 있는 연산자에 대해 기술한다.

3.1 알고리즘 개요

본 알고리즘은 상품을 상품속성집합으로 분류하여, 상품속성의 이면에 숨어 있는 속성집합간의 포함관계를 찾아, 사용자가 지정한 분류단계에서 분류체계를 자동으로 생성한다. 전체 알고리즘은 〈그림 1〉과 같다.

〈그림 1〉에서 ①은 사용자가 지정한 분류

```
begin
    input userDefinedDepth          —— ①
    load CE                         —— ②
    while(EOF of CE) {
        COMPUTE_GCA(CE)             —— ③
    }
    sort GCA                        —— ④
    while(EOF of GCA) {
        COMPUTE_RCA(GCA)            —— ⑤
    }
    MAKE_KEY(RCA)                   —— ⑥
    COMPUTE_PCR(RCA)                —— ⑦
    CREATE_TREE                     —— ⑧
end
```

〈그림 1〉 분류체계 자동 생성 알고리즘

단계에서 분류체계를 생성하기 위해서 사용자로부터 분류단계(userDefinedDepth)을 입력받는다. ②는 외부 파일로 작성된 상품과 상품속성 정보를 메모리에 로딩 하는 단계이며, 메모리에 로딩 된 상품과 상품속성을 ③에서는 각각의 상품을 상품 속성집합으로 분류하고, 상품명은 고유한 상품아이디로 표현한다. ④는 ③에서 생성된 속성집합을 일반적으로 사용하는 정렬 알고리즘을 이용하여 정렬한다. 정렬된 속성집합을 ⑤에서는 같은 성질의 속성집합을 하나의 속성집합으로 통합하고, ⑥은 ⑤를 이용하여 속성간의 포함관계를 쉽게 찾을 수 있도록 2개의 색인테이블을 생성하고, ⑦은 ⑥에서 생성된 색인테이블을 이용하여 속성간의 포함관계를 형성할 수 있는 후보 집합을 구하며 ⑧은 사용자가 지정한 분류단계에서 최대한 균형잡힌 분류체계를 자동으로 생성하고, 생성된 분류체계를 조정할 수 있는 기능을 제공한다.

각 단계별 알고리즘을 간략히 살펴보면 다음과 같다.

userDefinedDepth: 분류담당자가 생성하고자 하는 분류체계의 분류단계를 입력한다.

CE: 분류체계를 자동으로 생성하기 위해 파일로 작성된 상품데이터를 컴퓨터의 메모리로 올리는 과정으로 알고리즘에 대한 입출력은 〈그림 2〉와 같다.

CE
입력 : Excel로 작성된 상품집합
출력 : 메모리에 적재된 상품집합

〈그림 2〉 상품자료 적재 알고리즘의 입출력

COMPUTE_GCA: 파일로부터 메모리에 적재된 상품집합(GCE)을 상품속성별로 분류하는 단계로써 알고리즘의 입출력은 〈그림 3〉과 같다.

COMPUTE_GCA(GCE)
입력 : 상품집합
출력 : 속성별로 분류된 상품속성 집합

〈그림 3〉 속성분류 알고리즘의 입출력

속성간의 포함관계를 쉽고 빠르게 찾기 위해, 각 상품에 자연수의 고유한 상품아이디(Pid)를 부여하고, 속성별로 상품아이디(Pid)를 분류 한다. 상품을 분류하면서 Fn을 계산한다. Fn은 속성집합을 구성하는 Pid를 이용하여 계산할 수 있는 n개의 함수(Count, Sum 등)이며, 속성간의 중복이나 포함관계를 빠르게 파악하기 위함이다.

COMPUTE_RCA: 상품속성별로 분류된 집합들 중 상품속성을 구성하는 상품아이디(Pid)가 동일한 속성은 하나의 속성으로 통합하는 과정이다. 중복 속성을 제거하는 알고리즘의 입출력은 〈그림 4〉와 같다.

COMPUTE_RCA(GCA)
입력 : 속성집합
출력 : 중복이 제거된 속성집합

〈그림 4〉 속성통합 알고리즘의 입출력

MAKE_KEY: 속성집합간의 포함관계를 빠르게 찾기 위하여 2개의 색인을 생성한다. 첫 번째 색인은 속성의 현재 위치(RowID)

와 Pid을 이용하여 계산된 Fn(n=2,3,4....)으로 구성된 색인(ATTIDX)이며, 다른 하나는 Pid별로 속성들이 포함되어 있는 열의 위치(RowID)로 구성된 색인(PIDIDX)이다. 색인을 생성하는 과정의 알고리즘 입출력은 〈그림 5〉와 같다

MAKE_KEY(RCA)
입력 : 중복이 제거된 속성집합
출력 : 2개의 색인(ATTIDX, PIDIDX)
ATTIDX - 상품속성에 해당하는 Fn(Fn : 상품 아이디를 이용하여 계산할 수 있는 함수)
PIDIDX - 상품아이디에 해당하는 열번호(RowID)

〈그림 5〉 색인 테이블 입출력

COMPUTE_PCR: 속성간의 포함관계를 형성할 수 있는 후보 집합을 구한다. 두 속성 집합 RA_i와 RA_{i+k}가 포함관계를 형성하기 위해서는 $RA_i \subset RA_{i+k}$이고 $RA_i \neq RA_{i+k}$이어야 한다. 그러므로 RA_i+1은 RA_i의 모든 원소를 포함한다. 따라서 진부분집합의 정의를 이용하여 속성간의 포함관계를 형성할 수 있는 후보 집합을 생성한다. 후보 집합을 생성하는 알고리즘에 대한 입출력은 〈그림 6〉, 〈그림 6.1〉, 〈그림 6.2〉와 같다.

makeCandidateParent: ATTIDX색인을 이용하여 후보 집합을 생성한다. 각각의 속성들은 Fn을 가지고 있으므로, Fn을 비교하여 속성간의 포함관계를 형성할 가능성이 있는 속성들만을 추출하여 집합을 생성한다. 이렇게 함으로써 속성간의 포함관계를 형성할

COMPUTE_PCR(RCA)
입력 : 2개의 색인(ATTIDX, PIDIDX)
출력 : 속성간의 포함관계
(자식 - 부모 관계(C), 부모 - 자식 관계(P))

〈그림 6〉 포함관계를 찾는 알고리즘의 입출력

가능성이 없는 속성들은 배제가 되어, 포함관계를 찾는 연산시간을 획기적으로 단축할 수 있다. 후보 집합을 생성하는 알고리즘에 대한 입출력은 〈그림 6.1〉과 같다

makeCandidateParent(RA_i)
입력 : 상품속성 집합
출력 : 입력된 속성(RA_i)과 포함관계를 형성할 수 있는 후보집합(candidate)

〈그림 6.1〉 후보 집합 생성 알고리즘의 입출력

findCandParent: makeCandidateParent로 생성된 모든 후보 집합이 반드시 포함관계를 가지는 것은 아니므로, 후보집합 중에서 완벽한 포함관계를 형성하는 속성을 찾는 연산이 필요하다. 이때 색인 PIDIDX을 이용한다. 후보 집합과 PIDIDX을 AND연산을 수행하면 완벽한 포함관계를 형성할 수 있는 후보 집합이 생성된다. 후보 집합 생성 과정에서 각 속성이 가질 수 있는 부모-자식 관계 색인(P)과 자식-부모 관계 색인(C)을 생성한다. 이는 3.2에서 사용자가 지정한 분류단계에서 분류체계를 생성하기 위한 자료로 사용한다. 속성간의 계층구조를 찾아

후보 집합을 완성하는 과정의 알고리즘 입출력은 〈그림 6.2〉와 같다.

findCandParent(candidate, RAi)
입력 : 상품속성(RAi),
후보집합(candidate)
출력 : 후보집합
(자식-부모 관계 색인(C),
부모-자식 관계 색인(P))

〈그림 6.2〉 후보 집합 확정 알고리즘의 입출력

3.2 사용자 지정 트리 생성

본 장에서는 3.1에서 생성된 속성간의 포함관계를 이용하여 하향식 접근방법으로 분류체계를 생성하고자 한다. 자동으로 생성될 분류체계는 사용자가 지정한 분류단계에서 최대한 균형 잡힌 분류체계를 생성한다. 제안한 알고리즘에서는 각각의 속성은 자신의 조상 노드가 될 수 있는 속성집합(C)을 가지고 있으며, 동시에 각 속성이 가질 수 있는 자손 노드의 속성집합(P)도 생성되었다. 따라서 사용자가 지정한 분류단계에서 분류체계를 생성하기 위해 C, P를 활용하여 속성간의 계층관계를 찾는다.

사용자가 지정한 분류단계에서 분류체계를 생성하는 과정은 다음과 같다.

① 루트노드 결정: 모든 속성을 포함하고 있는 속성, 즉 자식노드의 개수가 가장 큰 값이 루트노드가 된다.

② 트리차수 결정: 모든 속성을 사용자가 지정한 분류 단계에 배치하기 위한 트리의 차수를 지수법칙을 이용하여 계산한다.

③ 자식노드 결정: 부모노드를 포함하는 속성들 중 자식노드의 개수가 많은 순서대로 트리의 차수만큼 선택하여, 자식노드로 결정한다. 자식노드로 결정된 속성은 다른 노드의 자식노드가 되지 못하도록 하기 위하여, 자식노드로 선정된 계층을 저장하여 관리한다.

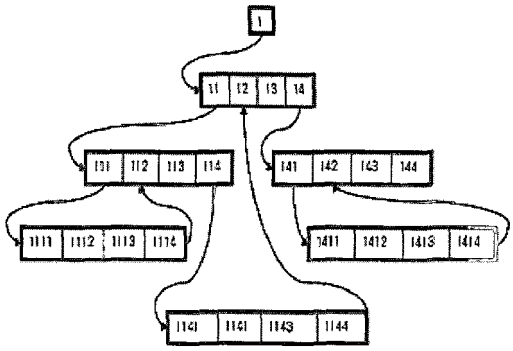
④ 부모노드 설정: ③에 의해 선정된 자식노드들의 첫 번째 노드부터 순차적으로 접근하여 현재 노드를 부모노드로 설정한다.

⑤ 재귀호출 및 트리의 높이 확인: 사용자가 지정한 트리의 높이까지 반복적으로 ③을 실행하고, 현재 트리의 높이가 사용자가 지정한 트리의 높이 이면 ④의 다음 속성을 처리한다.

⑥ 미 분류 속성 처리: 자식노드의 개수를 트리의 차수로 지정하고, 자식노드의 개수가 많은 순서대로 자식노드를 선정했기 때문에 자식노드의 수가 적거나 없는 속성들은 ⑤의 처리 과정까지 분류되지 않고 미 분류 속성으로 남는다. 따라서 미 분류 속성은 ⑤까지 생성된 트리의 범위를 벗어나지 않는 높이에서 자신과 가장 가까운 노드에 추가한다.

사용자가 지정한 분류단계에서 분류체계를 생성하는 과정은 〈그림 7〉과 같다.

알고리즘을 위한 용어의 정의는 다음과 같다.



〈그림 7〉 사용자 지정트리 생성방법

A = {A_i | A_i는 상품의 속성, $A_i \neq A_i+k$}

P = {(P_i, C_{j1}, C_{j2} ...., C_{jk}) | $P_i \in A$, C_{jk} ∈A, 모든 C_{jk}는 P_i의 자식노드가 될 수 있는 속성 집합}

C = {(C_i, P_{j1} P_{j2}, ..., P_{jk}) | $C_i \in A$, P_{jk} ∈A, 모든 P_{jk}는 C_i의 부모노드가 될 수 있는 속성집합}

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}

PCR = {(P_i, C_i) | $P_i \in A$, $C_i \in A$)

분류 체계 생성 과정을 알고리즘으로 표현하면 〈그림 8〉, 〈그림 8.1〉, 〈그림 8.2〉, 〈그림 8.3〉과 같다.

〈그림 8〉은 균형트리를 생성하기 위하여 사용자가 지정한 분류체계의 레벨을 참조하여 트리의 차수를 구한다. 다음으로 루트노드를 찾은 후, 재귀적 호출과 트리의 차수를 활용하여 자식노드를 결정한다. findChild 연산 후 부모-자식관계를 찾지 못한 미 분류 항목에 부모노드를 찾아서 지정하면, 사용자가 지정한 분류단계에서 모든 항목이 반영

```
CREATE_TREE(candParent)

입력: 속성간의 포함관계가 정의된 집합
출력: 계층구조의 분류체계
degreeCnt
← computeDegree(userDefinedDepth)
parentRowID ← root
findChild(parentRowID)
adjustTree(P)
cretae Tree(PCR)
```

〈그림 8〉 계층구조의 분류체계 생성

된 분류체계를 생성한다.

〈그림 8.1〉은 사용자가 지정한 분류단계에서 모든 속성이 반영된 분류체계를 생성하기 위해서 속성의 전체 개수를 파악한 후, 사용자가 지정한 레벨에서 균형트리가 생성될 수 있는 트리의 차수를 계산한다. 전체 속성의 개수를 N, 사용자가 지정한 트리의 높이를 h라 할 때, 균형트리를 이루기 위한

```
computeDegree(userDefinedDepth)

입력 : 전체 속성의 개수, 사용자가
       입력한 분류단계
출력 : 트리의 차수
totAtt ← totalAttribute()
d ← Power(totAtt, 1/(userDefinedDepth - 1)
```

〈그림 8.1〉 트리 차수 계산

```
computeDegree(userDefinedDepth)

입력: 전체 속성의 개수, 사용자가
      입력한 분류단계
출력: 트리의 차수
totAtt ← totalAttribute()
d ← Power(totAtt, 1/(userDefinedDepth - 1))
```

〈그림 8.1〉 트리 차수 계산

트리의 차수 d를 구하는 방법은 지수법칙을 응용하여 계산 한다.

$1+d^{h-1} \geq N$(1: 루트노드개수)

$= d^{h-1} \geq N-1$

$= d \geq \sqrt[h-1]{}\ of\ N-1$

$\therefore d \geq (N-1)^{\frac{1}{h-1}}$(d: 트리차수, 정수)

〈그림 8.2〉는 사용자가 지정한 분류단계에서 분류체계를 생성하기 위한 과정이다. 부모-자식관계를 찾기 위한 알고리즘은 트리의 기본적인 특성을 이용하여 제안한다. 트리의 특성중 하나는 노드가 트리의 상단에 위치할수록 자손노드의 수는 점점 더 많아지며, 루트노드가 가장 많은 자손노드를 가지고 있다. 따라서 자손노드가 많은 것을 계층구조의 상단에 위치시킴으로서 많은 상품에 포함되는 속성이 분류체계의 상단에 위치하게 된다. 분류체계를 생성하는 규칙은 다음과 같다.

① 루트노드는 모든 속성을 포함한다. 따라서 속성의 개수가 가장 큰 값이 루

```
findChild(parentRowID)

입력: 속성별로 자식노드가 될 수
      있는 속성집합
출력: 부모-자식관계가 설정된 집합(PCR)

fcc ← findCandChild(parentRowID)
fcc ← sort(fcc)

for (idx=1 to (idx<degreeCnt and idx<fcc.size()))
   childRowID  ← fcc(idx)
   childNode[idx]  ← childRowID
   PCRi[Pcode]  ← parentRowID
   PCRi[Pname]←
     findNodeName(parentRowID)
   PCRi[Ccode]  ← childRowID
   PCRi[Cname]  ←
findNodeName(childRowID)
   P(level) ← currentDepth
next

for (idx = 0 to childNodesize() )
   if currentDepth < userDefinedDepth
   currentDepth ← currentDepth + 1
   paretRowID  ← fcc(idx)
   findChild(parentRowID)
 else
   currentDepth ← currentDepth - 1
   return
 end if
next

currentDepth ← currentDepth - 1
return
```

〈그림 8.2〉 자식노드 결정

트노드가 된다.

② 루트노드로부터 트리를 생성하는 하향식 접근방법을 이용하여 트리를 생성한다.

③ 자식노드는 조상노드의 모든 속성을 포함한다.

$NODE_i \subseteq NODE_j$이고 $NODE_i \neq NODE_j$이면 $n(NODE_i) < n(NODE_j)$

$\therefore NODE_i$는 $NODE_j$의 부모노드

④ ${}_dC_{Pi} \Leftarrow n(C_{jk})$: 자식노드 결정

자식노드 결정은 부모노드를 포함하는 모든 자식노드들 중 각각의 노드가 가지고 있는 자식노드의 개수가 많은 순서대로 선택한다. 자식노드가 많은 속성을 선택하는 이유는 자식노드가 많으면 트리의 상단에 위치할 확률이 높기 때문이다.

⑤ $n(NODE\ child) \leq d$(d: 트리의 차수)

자식노드의 개수는 최대 트리의 차수를 넘지 않는다.

⑥ $h(T) \leq u$(h: 트리의 높이, T: 트리, u: 분류단계)

트리의 깊이가 사용자가 지정한 레벨보다 크면 자식노드를 찾는 작업을 중지하고 다음 속성의 자식노드를 찾는다. 사용자가 지정한 분류단계에서 단말노드가 생성된다.

전체 속성 수를 N이라 하고, 부모-자식관계를 찾은 속성들을 T 라고 하며, 사용자가 지정한 분류단계는 u라 할 때, 미 분류항목에 대한 처리과정은 다음과 같다.

① 미 분류항목 구분 : N - n(T): 미 분류된 항목들의 개수를 구한다.

② 미 분류항목의 부모노드 찾기

${}_1C_{Ci},\ P_{jk} \in T \wedge P_{jk} < u$

미 분류된 항목은 자신과 가장 가까운 부모노드를 찾는다. 이때 3.1에서 언급한 P, C 색인을 사용한다.

```
adjustTree(P)

입력 : 미 분류된 속성집합
출력 : 부모-자식관계가 설정된
       집합(PCR)

for (idx = 0 to P.size() )
   if ( P(level)  = 0 )
   childRowID     ← P(idx)
   parentRowID  ← findParent(childRowID)
   PCRi[Pcode]  ← parentRowID
   PCRi[Pname]← findNodeName(parentRowID)
   PCRi[Ccode]  ← childRowID
   PCRi[Cname]← findNodeName(childRowID)
 end if
next
```

〈그림 8.3〉 미 분류 속성 분류

〈그림 8.2〉을 수행한 후 모든 속성이 T에 반영 되지는 않는다. 현재 알고리즘이 자식노드의 개수를 기준으로 트리의 단계를 결정한다. 따라서 속성이 너무 세분화되어, 트리의 깊이가 깊어질 경우 잘리는 현상이 발생한다. 따라서 〈그림 8.3〉은 미 분류된 속성을 사용자가 지정한 분류단계에 포함시키기 위한 과정이다. 3.1에서 부모노드가 가질 수 있는 자식노드 집합(P)과, 자식노드가 포함될 수 있는 부모속성 집합(C)을 생성하였다. 따라서 현재 미 분류된 속성이 포함되어야 할 부모노드를 알 수 있으므로, 부모노드 중에서 사용자가 지정한 분류단계를 넘지

않는 속성들 중에서, 자신과 가장 가까운 부모노드를 찾아서 T에 반영한다. 이렇게 함으로써 모든 속성은 반드시 T에 포함되고, 부모 노드를 가지게 된다. 그러나 미 분류항목이 특정 노드에 몰리는 경우가 발생할 수 있으므로, 완벽한 균형트리를 유지하는 않는다.

3.3 분류체계 조작 연산자

상품속성을 이용하여 알고리즘에 의해 분류체계를 자동으로 생성하였다고 하더라도, 반드시 사용자의 의도와 일치하는 것은 아니다. 경우에 따라서는 항목의 추가, 수정, 삭제 등 분류항목에 대해 조작을 할 수 있는 연산자가 제공 되어야만, 보다 현실적인 분류체계가 된다. 따라서 본 장에서는 분류체계를 조작하기 위한 연산자를 제안하고 구현한다.

자동으로 생성된 분류체계가 사용자에게 적합한 분류체계가 되도록 조작하기 위해서 필요한 연산자는 추가, 이동, 삭제, 변경, 분리, 병합, 그룹화이다. 일반적인 계층구조의 분류체계에서 운영될 수 있는 조작 연산자는 [2, 4]에서 제안되었다. [2]에서는 의미적 분류모형의 분류체계를 조작할 수 있는 추가, 삭제, 병합, 분리에 대한 4가지 연산자 모델을 제시하였으며, [4]에서는 다수의 온톨로지를 하나로 병합하는 연산자에 대한 모델을 제시하였다. 따라서 본 논문에서는 [2, 4]에서 제시한 연산자들을 참조하여 본 알고리즘에 맞도록 제안하고, 추가적으로 그룹화, 이동, 변경에 대한 조작 연산자를 제안하여, 7개의 분류체계 조작 연산자를 구현하여 실험하였다. 조작 연산자 알고리즘은

userOperation()

입력 : 자동으로 생성된 계층구조의 분류체계
출력 : 사용자 조작이 반영된 분류체계

Insert_Tree(N_p, Nnew)
Merge_SubTree(N_p, N_i, ... N_k)
Mege_EdgeNode(N_i, N_j, ... Nm)
Split_SubTree(N_p, Nnew1, N_{new2})
Split_EdgeNode(N_i, N_j, ..., N_m)
Group_Tree(N_p, N_i, N_j, Nk)
Delete_Tree(N_p)
Move_SubTree(N_p, N_i)
Move_EdgeNode(N_p, N_i)
Rename(N_p, newname)

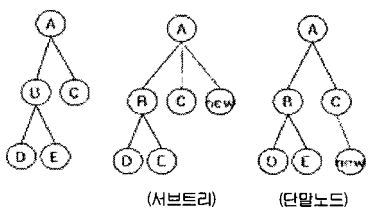
〈그림 9〉 조작 연산자

〈그림 9〉 와 같다.

각각의 연산자들은 다음과 같이 정의한다.

입력(Insert): 부모노드에 새로운 노드를 추가한다.

Insert_Tree(Np, Nnew): 지정한 분류항목 p에 새로운 분류항목 new를 추가한다. 분류항목의 추가는 단말노드에서도

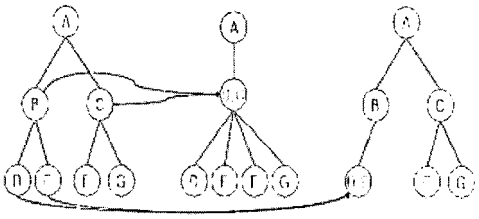


발생할 수 있으며 서브트리에서도 가능하다.

통합(Merge): 2개 이상의 단말노드나 서브트리를 통합하여 하나의 서브트리나 단말노드로 생성한다.

Merge_SubTree(N_p, N_j, ..., N_k): 서브트리 i,...,k를 통합하여 새로운 분류항목 p를 생성한다.

Mege_EdgeNode(N_j, N_j, ... N_m): 단말노드 j,...,m을 노드 i로 통합한다.

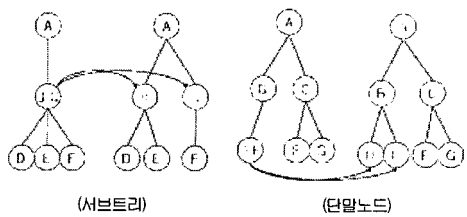


(서브트리) (단말노드)

분리(Split): 하나의 서브트리나 단말노드를 여러 개의 서브트리나 단말노드로 분리하는 연산자이다.

Split_SubTree(N_p, N_{new1}, N_{new2}): 서브트리 p를 서브트리 new1, new2로 분리한다. 서브트리의 자손노드들도 new1, new2에 따라 분리된다.

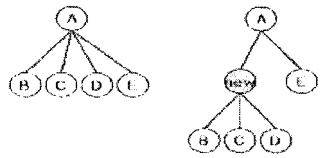
Split_EdgeNode(N_i, N_j, ..., N_m): 단말노드 i를 단말노드 i부터 m으로 분리한다.



(서브트리) (단말노드)

그룹화(Group): 여러 개의 서브트리나 단말노드를 하나의 서브트리로 생성한다.

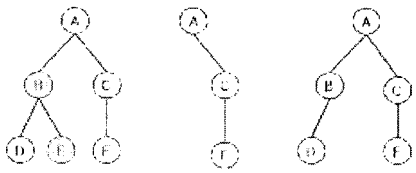
Group_Tree(N_p, N_i, N_j, N_k): i, j, k의 서브트리나 단말노드를 하나의 서브트리 p로 생성한다.



(서브트리)

삭제(Delete): 서브트리나 단말노드를 삭제한다.

Delete_Tree(Np): 노드 p를 삭제한다. 만일 p가 서브트리이면 p의 모든 자손노드들도 삭제한다.



(서브트리) (단말노드)

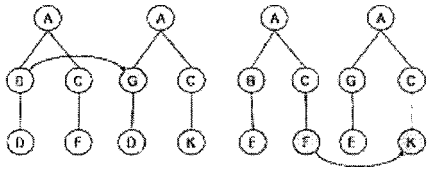
이동(Move): 서브트리나 단말노드를 이동한다.

Move_SubTree(N_p, N_i): 서브트리 i를 p의 하위노드로 이동한다. 서브트리 i의 모든 자손노드들은 p를 조상노드로 갖는다.

Move_EdgeNode(N_p, N_i): 단말노드 i가 단말노드 p의 하위노드로 이동한다.

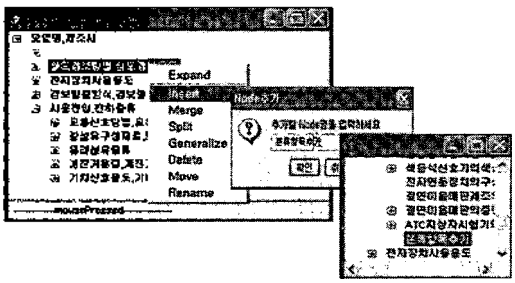
변경(Rename): 서브트리나 단말노드의 이름을 변경한다.

Rename(Np, newname) : 서브트리나 단말 노드 p의 명칭을 newname으로 변경한다.



(서브트리) (단말노드)

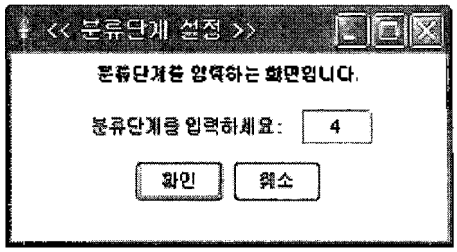
위에서 정의된 7가지의 분류체계 조작 연산자를 구현함으로써, 사용자는 시스템에 의해 자동으로 생성된 분류체계를 기반으로 하여 사용목적에 적합한 분류체계를 완성할 수 있다. 제안한 분류체계 조작 연산자를 구현한 화면은 〈그림 10〉와 같다.



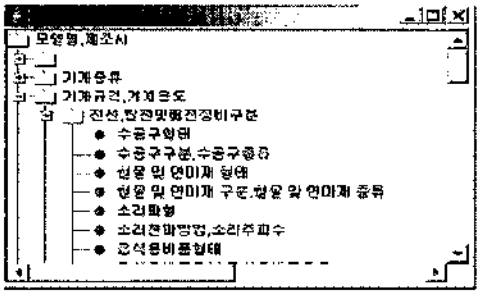
〈그림 10〉 사용자 연산자 구현화면

4. 실험결과

본 논문에서는 Excel로 작성된 100,000건의 상품자료를 실험데이터로 사용하였다[6]. 한 개의 상품이 가지고 있는 속성은 최소 7개에서 최대 15개까지 다양하게 구성되어 있다. 상



〈그림 11〉 분류단계 입력 화면



〈그림 12〉 분류체계 생성화면

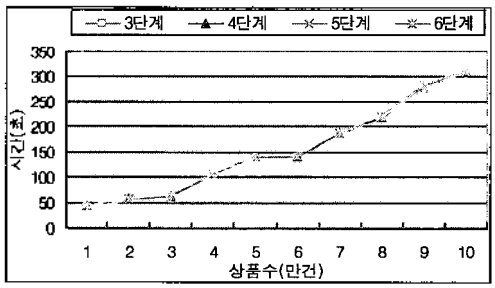
품 속성 당 평균 크기는 24byte이며, 전체 파일크기는 24Mbyte이다. 실험은 2GHz CPU와 2GByte의 메모리를 가진 PC환경에서 JDK1.4를 가지고 알고리즘을 구현하여 실험하였다.

실험방법은 분류단계(3, 4, 5, 6단계)별로 각 상품을 1만건 단위로 분할하여, Excel자료를 읽는 시간부터 계층구조의 분류체계가 생성되기까지 걸리는 연산시간을 측정하고 분석한다. 또한 제시한 알고리즘에 의해 생성된 분류체계가 사용자가 지정한 분류단계에서 정확히 생성되는지 실험한다. 마지막으로 이전연구[9]와 본 연구를 비교 분석한다.

사용자가 분류단계를 지정하는 화면은 〈그림 11〉과 같고 분류단계별 생성화면은 〈그림 12〉과 같다. 각 분류단계별 수행시간은 〈표 1〉, 〈그림 13〉와 같다.

〈표 1〉 분류단계별 수행 시간(/초)

건수(만)	3단계	4단계	5단계	6단계
1	43	44	44	45
2	56	57	57	57
3	62	63	63	63
4	106	107	108	108
5	138	139	139	139
6	140	140	141	141
7	186	188	187	188
8	218	218	219	219
9	278	281	275	283
10	317	311	311	312



〈그림 13〉 분류단계별 수행시간

실험결과 〈그림 12〉에서 보듯이 사용자가 지정한 분류단계에서 모든 속성이 분류됨을 알 수 있다. 이는 시스템에 의해 일방적인 분류단계로 생성되는 분류체계의 단점을 극복하고, 각 조직의 목적에 맞는 분류단계를 지정하여 생성할 수 있음을 보임으로써, 보다 현실적인 분류체계가 된다.

각 분류 단계별 연산시간은 〈표 1〉, 〈그림 13〉과 같이 분류단계별로 연산시간은 거의 변동이 없이 동일한 형태를 보인다. 그러므로 사용자가 지정한 분류단계와 상관없이 상품 건수별로 항상 일정한 속도를 보장하며, 선형 증가 모형을 보임으로써, 다량의 상품에 대해서도 수행시간을 충분히 예측할 수 있다.

〈그림 8.2〉자식노드 결정 알고리즘에서 분리되지 않고 미 분류항목으로 남는 속성들이 발생한다. 미 분류 항목은 사용자가 지정한 단계에서 분류체계가 생성되기 위한 과정에서 발생하며, 이는 속성이 매우 다양하고 복잡하여 상품 속성간의 정확한 포함관계를 찾을 경우에 트리의 깊이가 너무 깊어져 분류체계 생성과정에서 잘림 현상이 발생한다. 제안한 알고리즘의 특성은 자식노드가 될 수 있는 속성의 개수를 기준으로 분류체계를 생성하기 때문에 자식노드의 수가 적거나 트리의 깊이가 깊으면 미 분류 항목이 발생할 수 있다. 발생한 미 분류 항목은 재 분류과정에서 상품속성의 손실 없이 모든 속성이 반영된 분류체계가 생성된다. 자동으로 생성된 분류체계는 본 논문에서 제시한 분류항목 조작 연산자를 이용하여, 사용자나 조직의 의도에 따라 조정 작업을 할 수 있도록 함으로써, 시스템에서 자동으로 생성되는 분류체계의 단점을 극복하고 용도에 적합한 분류체계를 생성할 수 있다.

본 연구에서는 [9]의 상품속성간의 포함관계를 찾는 방법을 변경하였고, [9]의 깊이 우선 분류체계 생성방법을, 사용자가 분류단계를 지정하여 분류체계를 생성할 수 있도록 개선함으로써 트리의 생성시간을 단축시켰다. 따라서 제안한 알고리즘으로 100,000건의 상품에 대한 분류체계를 생성할 경우[9]보다 수행성능이 향상되었다. 이는 상품속성의 포함관계를 연산하는 시간과, 트리를 생성하는 시간이 기존 연구보다 효율적임을 입증한다.

5. 결론 및 향후연구

본 연구에서는 상품과 상품속성을 이용하여 사용자가 지정한 분류단계에서 분류체계를 자동으로 생성함을 보였다. 이는 분류 전문가가 아니더라도 상품에 대한 정보를 알고 있거나 또는 이미 작성된 상품자료를 활용하여, 제안한 알고리즘에 생성하고자 하는 분류단계를 입력하면, 자동으로 계층구조의 분류체계가 생성된다. 사용자는 자동으로 생성된 분류체계를 기반으로 하여, 분류체계 조작 연산자를 이용하여 사용목적에 적합한 분류체계를 생성할 수 있다. 알고리즘 수행시간은 〈그림 13〉에서 보듯이 100,000건의 상품에 대한 분류체계를 생성하는데 수 분만에 처리가 가능하며, 상품건수별로 연산시간이 선형증가모형을 보인다. 따라서 상품건수에 따른 분류체계 생성시간을 예측할 수 있으나, 대용량의 상품데이터를 취급하는 전자상거래업체나 오픈 마켓 등에서도 적용할 수 있는지에 대한 연구가 추가적으로 필요하다.

앞으로의 과제는 속성간의 포함관계를 찾아 트리구조로 표현할 때 발생되는 미 분류 항목의 발생 비율을 줄이고, 미 분류 항목이 효율적으로 분류체계에 반영되도록 알고리즘을 개선해야한다. 또한 검색 성능 향상을 위해 생성된 분류체계가 균형트리를 유지하는 방법에 대한 연구가 필요하다. 향후에는 사용자가 확정한 상품 분류체계를 기반으로 하여 데이터베이스 스키마를 자동으로 생성하는 방법을 적용하고자 한다[13]. 자동으로 데이터베이스 스키마가 생성된 후, 분류체계를 생성하기 위해 사용한 상품들을 분류체계에 맞게 분류하여, 데이터베이스에 저장하면 시스템에 바로 적용할 수 있는 완벽한 분류시스템이 된다.

참 고 문 헌

[1] 김건호, "분류검색 시스템 도입", 경영과 컴퓨터, 2005.

[2] 김동규, "전자카탈로그의 의미기반 모델 연구", 박사학위논문, 서울대학교, 2004.

[3] 김동규, 이상구, 최동훈, " 상품 데이터베이스의 동적 특성을 지원하는 분류 모형", 정보처리학회 논문지, 제12-D권, 제1호, pp. 165-178, 2005.

[4] 김정민, "토픽맵 기반의 철학 온톨로지 구축과 매핑 및 통합 기법", 박사학위논문, 서울대학교, 2007.

[5] 김효래, 장영철, 이창훈, "사용자 의도 트리를 사용한 동적 카테고리 재구성", 한국정보처리학회 논문지B, 제8권, 제6호, pp. 657-668, 2001.

[6] 상품정보, "ghost.mju.ac.kr", 2006.

[7] 장두석, 전종훈, "분류체계 자동생성을 위한 기법 설계", KDBC2006, pp. 297-304, 2006.

[8] 장두석, 전종훈, "상품 속성정보를 이용한 분류체계 자동 생성" 정보처리학회논문지, 제14-D권, 제4호, 게재예정, 정보처리학회, 2007.

[9] 장두석, 전종훈, "효율적인 상품 분류체

체 생성 기법 구현", KDBC2007, pp. 365-371, 2007.

[10] 정한민, 김평, 성원경, "전자상거래 검색 기술 동향", ITFIND, 주간기술동향, 제 1273호, 2006.

[11] Ben Wolin, "Automatic Classification in Product Catalogs", ACM SIGIR, pp. 11-15, 2002.

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

[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] 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.

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

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

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

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

[19] Kwan Yi, "Challenges in automated classification using library classification schemes", world library and information congress, 2006.

[20] Sang-goo Lee, "Design & Implementation of an e-Catalog Management System", DASFAA 2004 Tutorial, 2004.

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

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

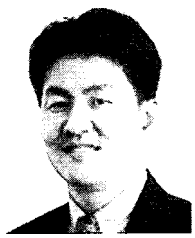
[23] UPC, "Universal Product Code", http://en.wikipedia.org/wiki/Universal_Product_Code.

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

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

저 자 소 개

장두석 (E-mail : dsjang@mju.ac.kr)
1995. 한국방송대학교 전자계산학과 (학사)
2000. 경기대학교산업정보대학원 전자계산학과 (석사)
2005. 명지대학교 대학원 컴퓨터공학과 (박사과정수료)
1994 ~ 현재 (주)이건창호시스템
관심분야 전자상거래, 데이터베이스, CRM



전종훈 (E-mail : jchun@mju.ac.kr)
1986. University of Denver 전산과학과 (학사)
1992. Northwestern University 전산과학과 (석사, 박사)
현재 명지대학교 컴퓨터공학과 교수
2001 ~ 현재 ㈜프람트 대표이사 사장
관심분야 전자상거래, 의료정보, CRM, 디지털 라이브러리