웹 에이전트 사용자 특성모델 구축을 위한 비감독 문서 분류※

오재준* · 박영택**

Unsupervised Document Clustering for Constructing User Profile of Web Agent※

Jae-jun Oh* · Young-Tack Park**

요 약

본 연구는 웹 에이전트에 있어서 가장 핵심적인 부분이라 할 수 있는 사용자 특성모델 구축방법을 개선하는데 목적을 두고 있다. 사용자 특성모델을 귀납적 기계학습 방식으로 자동 추출하기 위해서는 사용자가 관심을 가지는 분야별로 문서를 자동 분류하는 작업이 매우 중요하다. 지금까지의 방식은 사람이 관심여부에 따라 문서를 수동적으로 분류해 왔으나, 문서의 양이 기하급수적으로 증가할 경우 처리할 수 있는 문서의 양에는 한계가 있을 수밖에 없다. 또한 수작업 문서분류 방식을 웹 에이전트에 그대로 적용하였을 경우 사용자가 일일이 문서를 분류해야한다는 번거로움으로 인해 웹 에이전트의 효용성이 반감될 것이다. 따라서 본 연구에서는 비감독 문서분류 알고리즘과 그것을 바탕으로 얻어진 문서분류정보를 후처리(Post-Processing)함으로써 보다 간결하고 정확한 문서분류 결과를 얻을 수 있는 구체적인 방법을 제공하고자 한다.

※ 본 연구는 숭실대학교 교내 연구비의 지원을 받았습니다.
* 숭실대학교 컴퓨터학부 대학원
** 숭실대학교 컴퓨터학부 대학원

1. 서 론

1.1. 연구의 배경 및 목적

웹 에이전트의 기본적인 역할은 사용자가 관심을 가지는 정보의 주제를 파악하여 관련된 정보를 수집, 제공하는 것이라 할 수 있다. 웹 에이전트가 이 같은 기본적인 역할을 효과적으로 수행하여 실제 개인의 정보검색에 도움을 주기 위해서는 무엇보다 사용자가 선호하는 정보의 주제를 파악하는 일이라 할 것이다. 즉 사용자 특성모델을 얼마나 효과적으로 구축하느냐에 따라 웹 에이전트의 성능이 좌우된다할 수 있을 것이다. 사용자 특성모델의 구축에 있어서는 두 가지 방법을 고려해 볼 수 있는데, 첫 번째, 사용자가 자신의 관심 여부를 문서별로 일일이 지정하여 특성모델 구축에 이용하는 방법으로 사용자가 직접 관심 여부를 지정하기 때문에 매우 정확한 특성모델을 구축할 수 있는 반면 문서 하나 하나에 대한 분류 작업을 사용자가 직접 수행한다는 치명적인 단점을 가진다. 두 번째, 사용자가 조회한 문서들을 대상으로 URL을 저장하거나, 문서를 별도로 저장 또는 인쇄하는 등, 관심의 가진다고 판단되는 문서들에 대해 분류함수(Category Utility)를 적용하여 사용자가 관심을 가지는분야의 주제를 파악한 후 이를 특성모델 구축에 이용하는 방법이 있다. 후자의 경우에는 전자가 가지는 단점을 해결할 수 있는 반면 어떠한 문서분류 방법을 적용하느냐에 따라 사용자 특성모델의 정확도가 결정되므로 효과적인 문서분류 방법의 적용은 웹 에이전트에 있어서 필수적이라 할 것이다.

근래에는 문서를 분류함에 있어서 단순히 관련이 있는 문서를 분류하는 것에 그치지 않고 문서라는 일련의 입력 정보를 토대로 각 문서가 내포하고 있는 지배적인 주제를 찾아내고, 해당 문서에 대한 개요 및 숨겨진 유사성(Hidden similarity)을 파악, 사용자의 요구에 대한 다양하고 간결한 정보를 제공하는 텍스트 마이닝(Text Mining)에 대한 연구가 활발히 진행중이다 [intelli]. 텍스트 마이닝은 데이터 마이닝(Data Mining)의 한 분야로써 기본적으로 일련의 문서들을 처리 대상으로 하며, 연속된 입력 자료들 내의 키워드의 출현빈도, 형태, 연관성 등을 토대로 많은 양의 정보를 보다 체계적이고 효과적으로 알아볼 수 있도록 가공하는데 목적을 두고 있다[Helena97]. 효과적인 텍스트 마이닝은 웹 에이전트의 사용자 특성모델 구축에 필요한 문서분류정보를 얻는데 활용될 수 있을 것으로 기대된다.

1.2. 연구범위

본 연구에서 제시하고자 하는 사용자 특성모델 구축과정은 다음과 같다. 먼저 사용자 특성모델 구축에 사용될 문서들은 분류되지 않은 상태로 수집되어 일차적으로 트리의 형태로 변환된다. 문서들을 키워드에 의해 트리의 형태로 분류하는 데에는 비감독 개념학습 알고리즘(Unsupervised Conceptual Learning Algorithm)의 일종인 COBWEB을 사용한다. COBWEB 알고리즘의 출력으로 생성되는 분류트리에서는 서로 비슷한 키워드의 분포를 보인 문서들이 인접한 노드로 분류된다. 이처럼 트리의 형태로 분류된 각 문서들은 문서들의 유사한 정도에 따라 인접한 노드에 분류되고 동일한 부모노드를 갖지만 결국 모든 문서들이 하나의 루트노드 아래에 존

재하는 단말노드에 분류되므로 트리 자체로는 어느 문서가 어떤 범주에 속하는지 알 수 없다. 따라서 분류트리로부터 범주정보를 얻기 위해서는 트리를 적당히 분할하는 소그룹화(Subgrouping) 과정과 분할된 소그룹을 각 소그룹의 유사도에 따라 하나의 그룹으로 병합(Merge)하는 소그룹 병합(subgroup merging)을 거쳐야 한다. 본 연구에서는 특성모델 구축에 사용될 분류되지 않은 문서들의 집합을 사용자 특성모델을 구축하기 직전까지의 범주정보로 변환하는 문서분류에 연구의 초점을 맞추고 있다.

1.3. 논문의 구성

2장에서는 관련 연구로 IBM Intelligent Miner for Text에 관하여 간략히 살펴보고, 제 3장에서는 웹 에이전트에서의 사용자 특성모델 구축 과정에 앞서 문서들을 주제별로 분류하여야할 필요성에 대하여 언급하며, 4장에서는 비감독 범주 효용성 알고리즘인 COBWEB의 개요와, 문서분류에의 적용에 대한 방법론에 대하여 설명하고, 5장에서는 COBWEB을 이용하여 얻어진 분류트리를 후처리(Post-Processing)해야할 필요성에 대해 설명하고, 6장에서는 문서분류모듈의 구현을, 7장에서는 본 논문에서 제시하고있는 분류트리의 후처리를 통하여 얻어진 문서분류정보와 이를 토대로 추출된 사용자 특성모델의 예를 보이고, 끝으로 8장에서는 결론 및 향후 연구방향에 대하여 기술한다.

2. 관련 연구

데이터 마이닝(Data Mining)의 한 분야라 할 수 있는 텍스트 마이닝(Text Mining)은 처리하여야할 문서의 양이 많아짐에 따라 다양한 분야에서 그 필요성이 대두되고 있다. 텍스트 마이닝의 목적은 수집된 문서들을 대상으로 분류작업 및 문서가 내포하고 있는 정보들을 체계화하여 사용자의 요구에 따라 가공된 정보를 제공하는 데 있다. 텍스트 마이닝에서 기본적으로 주목하고 있는 기법은 수집된 문서들을 대상으로 키워드 추출, 동시 출현 단어파악 (Co-occurring term), 동의어, 파생어, 반의어 인식, 다국어 해석, 문서간 연관성 파악 등의 다양한 처리과정을 거쳐 사용자의 요구(Query)에 맞는 형태로 문서처리 결과를 제공하는 것이다[Intelli]. 흔히 사용되는 간단한 문서비교 방식이 문서 내에서 찾아낸 키워드의 빈도에 의존하고 있는 반면 텍스트 마이닝에서 비중을 두고 있는 부분은 키워드의 출현 빈도 외에 단어가 문장 내에서 가지는 문맥적 의미와 언어적 특성을 분류 및 정보가공에 활용하는 데 있다고 볼 수 있다.

2.1. IBM Intelligent Miner for Text

Intelligent Miner는 IBM에서 연구중인 텍스트 마이닝을 이용한 툴킷(toolkit)으로써, 비정형적인 문서들을 기업체, 관공서, 또는 개인이 필요로 하는 정보로 변환시켜 이용할 수 있도록 해주는 유틸리티 소프트웨어이다.

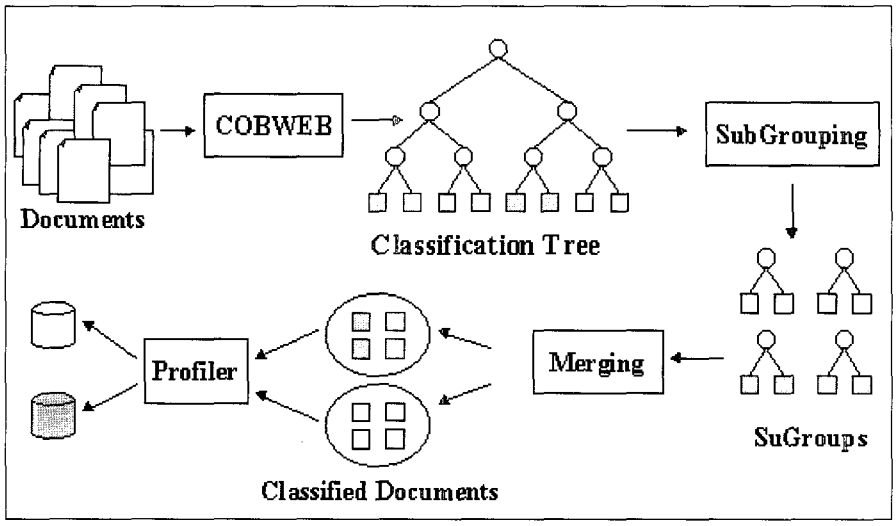
Intelligent Miner는 문서로부터 키워드를 추출하고, 문서들을 주제별로 분류한 다음 분류된 문서들의 그룹에서 지배적인 주제를 찾아내 사용자로 하여금 필요에 따라 원하는 문서를 찾고, 찾아낸 문서로부터 다양한 정보를 얻어낼

수 있도록 해 준다. 이와 같은 절차를 자세히 살펴보면 다음과 같다. 먼저, 문서로부터 키워드를 추출해 내는 특성 추출(Feature Extraction) 단계에서는 사람, 조직, 장소 등의 이름 및 동일 인명에 대한 다른 단어간의 링크 구축, 다양한 방면의 전문용어, 약어, 숫자, 날짜 처리가 포함되어 보다 융통성 있는 키워드 추출을 가능하게 한다. 다음으로, 추출된 키워드에 따라 문서들을 분류하는 단계에서는 문서집합이 갖는 개요, 숨겨진 유사성 파악(hidden similarity), 예외 인식 등으로 문서집합에 대한 정보조회를 용이하게 한다.

Intelligent Miner는 Basic Retrieval System과 Text Mining Enhancement 두 부분으로 구성되어 있는데 Basic Retrieval System에서는 16개의 언어를 지원하며, 동의어, 파생어, 반의어 등 세련된 언어학적 처리기능을 제공하고 하나의 검색 엔진으로 여러 개의 영역을 검색할 수 있도록 해 준다. Text Mining Enhancement에서는 키워드 추출과 동시에 키워드에 대한 색인을 생성하여 사용자 조회 시 사용될 수 있도록 하며, 사전에 사용자가 서로 관련된 문서와 그렇지 않은 문서들을 지정하도록 하여 정확도를 향상시킨다. 이처럼 Intelligent Miner를 비롯한 텍스트 마이닝 분야에서 문서를 대상으로 분류 및 정보가공에 사용되는 기법들은 웹 에이전트의 특성모델 구축을 위한 문서분류에도 응용될 수 있을 것이다.

3. 웹 에이전트 시스템과 사용자 특성모델

웹 에이전트의 기본적인 역할은 정보의 바다라고 일컬어지는 웹 상에서 사용자가 원하는 정보를 쉽게 수집하고 조회할 수 있도록 보조해주는 역할이라 할 수 있다. 웹 에이전트는 기본적으로 사용자가 선호하는 분야에 관한 정보를 저장하고 있는 사용자 특성모델을 구축하여 이를 바탕으로 사용자의 웹 항해를 보조한다. 물론 사용자 특정모델은 사용자의 웹 항해에 따라 지속적으로 갱신된다. 이러한 특성으로 인해 웹 에이전트라는 말 앞에 적응형(Adaptive)라는 말을 붙이기도 한다. 사용자 특성모델 구축에 있어서 우선 웹 에이전트는 사용자가 관심을 가진다고 판단되는 HTML문서들을 수집한다. 예를 들면 사용자가 특정 URL을 별도로 저장(Book marking)하거나 해당 HTML 페이지를 인쇄. 또는 파일로 저장할 경우 이러한 문서들은 웹 에이전트에 의해 별도로 저장되어 사용자 특성모델 구축에 이용된다. 사용자 특성모델 구축 과정을 살펴보면 우선 HTML문서에 대해 HTML Tag와 불용어, 접미어(Suffix) 제거작업이 수행된다. 이렇게 추출된 일련의 단어들은 각각의 단어의 문서 내 발생빈도(Term Frequency)와 문서별 발생빈도(Document Frequency)에 따라 키워드 벡터(Keyword Vector)로 압축되어 구체적인 분류 알고리즘에 적용된다. 본 연구에서는 분류 알고리즘으로 COBWEB 알고리즘을 사용하였다. 각각의 문서별 키워드 벡터들을 토대로 COBWEB은 최종적으로 분류트리를 출력하는데, 서로 비슷한 키워드 분포를 보이는 문서들은 하나의 부모 노드 아래에 배치된다. 본 연구에서 주목하고 있는 부분은 HTML 문서로부터 키워드 벡터를 추출하거나 COBWEB 알고리즘을 통해 분류트리를 생성해 내는 부분이 아니고 생성된 분류트리로부터 최종적인 분류정보를 얻어내는 데에 있다. 분류트리는 일반적인 트리와 마찬가지로



〔그림 3-1〕 문서분류 절차

노드와 링크로 구성된 하나의 트리에 지나지 않는다. 이러한 분류트리는 그대로 사용자 특성모델 구축에 사용될 수 없고 반드시 적절한 형태로 분할되어야 하는데 본 연구에서는 분류트리의 분할방법과 분할된 분류트리를 재차 병합하여 최종적인 분류정보를 사용자 특성모델 구축 프로시저에 전달하는 구체적인 방법을 보이고 실험결과를 제시하고자 한다.

그림 3-1은 사용자가 선호(Preference)를 제시한 HTML 문서들을 대상으로 사용자 특성모델을 구축하는 과정으로써 먼저 수집된 문서들은 COBWEB 알고리즘에 의해 각각의 문서들이 포함하고 있는 키워드 토대로 분류트리로 만들어진다. 이렇게 생성된 분류트리는 사용자 특성모델 구축에 바로 이용될 수 없으므로 분할 및 병합 과정을 거쳐 최종 분류정보로 변환되어 사용자 특성모델 구축을 위한 프로파일 생성기(Profiler)에 전달된다.

4. 비감독 개념학습

본 연구에서는 사용자 특성모델을 구축하기 위한 문서분류 방식으로 COBWEB 알고리즘을 이용하였다. 본 장에서는 COBWEB 알고리즘과 문서로부터 추출된 키워드 벡터들을 대상으로 COBWEB 알고리즘을 적용하여 중간분류결과를 얻기 위한 세부사항에 관해 기술한다.

4.1. COBWEB

COBWEB은 점진적인 개념 형성(Incremental Concept Formation)에 기초한 학습 알고리즘으로서, 범주 정보가 주어지지 않은 예제들을 대상으로 각 예제그룹의 내용에 따라 트리의 형태로 분류정보를 도출해내는 알고리즘이다[Fisher96]. 점진적 개념 형성 모델은 분류 대상이 되는 예제집합에 새로운 예제가 추가되더라도 모든 예제들을 다시 처리하지 않고 추가된 예제들만 처리

대상에 포함된다는 장점을 갖기 때문에 웹 에이전트의 특성모델 구축에 쉽게 적용될 수 있다.

COBWEB의 입력 자료는 속성-값의 쌍으로 이루어진 하나 이상의 특성치로 주어진다. 여기서 속성은 모든 입력 자료에 공통적으로 등장하는 것이어야 한다. COBWEB에서 새로운 입력 예제를 분류트리에 추가하기 위해서는 트리의 최상위 노드부터 예제를 대상으로 분류함수(Category Utility)를 적용하여 예제를 배정할 하위노드를 결정하여 예제가 단말노드에 배정될 때까지 이 같은 과정을 반복하여 적용하게 된다. 분류함수의 기본 개념은 새로운 예제가 들어왔을 때 현 단계에서 예제를 각각의 가능한 분류경로(자식노드)로 분류하였을 때의 유사도 값에서 분류하지 않았을 때의 유사도 값을 뺀 결과 값이 가장 큰 분류경로를 선택하고, 이 같은 분류절차를 예제가 단말 노드에 이를 때까지 반복한다는 것이다.

$$\frac{\sum_{k=1}^{K} P(C_k) \sum_{i=1}^{I} \sum_{j=1}^{J} P(A_i = V_{ij} \mid C_k)^2 - \sum_{i=1}^{I} \sum_{j=1}^{J} P(A_i = V_{ij})^2}{K}$$

위 수식에서 P(Ck)란 분류 k(자식노드)의 전체(전체노드)에 대한 비율이고, P(Ai = Vij | Ck)는 주어진 분류에 대하여 개체의 속성이 특정 값을 가지는 확률을 나타내며, I는 속성의 개수, J는 학습개체의 개수를 나타낸다. 이러한 분류함수는 COBWEB에서 하나의 학습예제가 입력됨에 따라 분류트리의 루트노드 단계에서부터 예제가 단말노드에 이를 때까지 반복 적용되며 예제는 분류함수의 값에 따라 새로운 노드로 분류되거나 기존의 노드로 분류될 수 있고 기존의 노드가 병합되거나 분할될 수 있다.

본 연구에서는 비교연산 자체의 효율을 높이기 위해 위의 식을 다음과 같은 수식으로 변경하여 사용한다.

$$\frac{\sum_{k=1}^{K} P(C_k) \sum_{i=1}^{I} \frac{1}{\sigma_{ik}} - \sum_{i=1}^{I} \frac{1}{\sigma_{ip}}}{K}$$

4.2. COBWEB의 연산자(Operator)

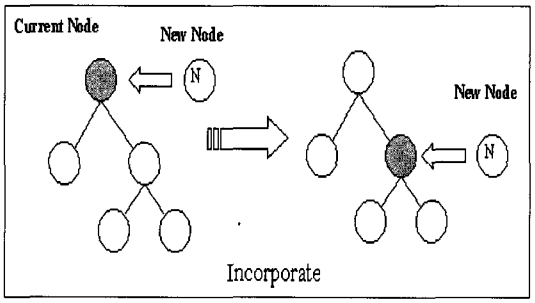
COBWEB은 위에서 설명한 분류함수를 바탕으로 측정된 평가수치를 적용하여 입력된 각 예제를 이전까지 구축된 분류트리에 추가시키면서, 그 내용에 따라서 트리의 형태를 변경시키는 작업까지 수행하게 된다. 이러한 작업을 COBWEB에서는 대표적인 4가지 유형으로 분류하여 각 경우에 가장 최적인 범주 측정수치를 결과로 적절한 분류연산을 적용하게 된다. 분류연산의 종류는 Incorporate, Create-new-disjunct, Merge, 그리고 Split이 있으며, Incorporate의 경우, 새로 입력되는 예제를 기존의 하위 범주에 포함시키는 것이며, Create-new-disjunct는 새로 입력된 예제가 기존 분류 범주와 유사성이 적을 때 새로운 하위 범주를 생성하는 것이며, Merge와 Split은 새로 입력되는 예제의 내용에 따라 각각 기존의 범주계층 구조를 병합하거나 분할하여 분류트리의 형태를 변경하는 작업이다.

현재 주목하고 있는 노드(Current Node)에 새로운 노드(New Instance)가 들어왔을 경우 입력된 노드를 새로운 노드로 분류(Create New Disjunct)하는 경우와 입력된 노드를 현재 주목하고 있는 노드의 자식노드 가운데 하나로 분류하는 경우의 분류함수 값을 구한다. 현재 주목하고 있는 노드의 자식노드에 입력된 노드를 분

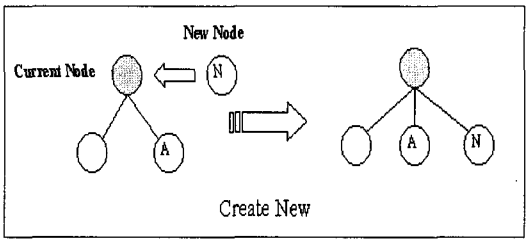
류한다고 가정했을 때 가장 큰 분류함수 값을 B, 두 번째로 큰 분류함수 값을 S, 새로운 노드로 분류하였을 때의 분류함수 값을 C라 하였을 때 C값이 B값보다 크면 입력된 노드를 새로운 노드로 분류(Create New Disjunct)하고 B값이 C값보다 큰 경우에는 Merge와 Split의 경우를 고려하여야 한다.

Best Node와 Second Best Node를 병합(Merge)한 노드에 입력된 노드를 분류하였을 때의 분류함수 값을 M이라 하고 Best Node의 Child노드들을 부모노드에서 분리하여 부모노드를 제거하고 차상위 부모모드에 추가한 후 추가된 노드들 가운데 Best Node에 입력된 노드를 분류하였을 때의 분류함수 값을 P라 하면 B, M, P 세 값들 가운데 가장 큰 값에 해당되는 분류 함수를 선택하여 트리의 형태를 변경하고 입력된 노드를 해당 노드로 분류한 후 입력된 노드가 단말노드에 분류될 때까지 이상의 분류작업을 반복한다.

그림 4-1에서는 새로 입력된 노드 N이 현재 주목하고 있는 Current Node의 오른쪽 자식 노드로 분류되는 것을 보이고 있는데 N을 오른쪽 자식노드로 분류하였을 때의 분류함수 값 B가 C, M, P보다 큰 경우에 해당된다.

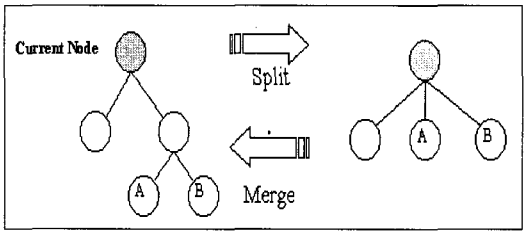


〔그림 4-1〕 Incorporate(하위노드로 분류)



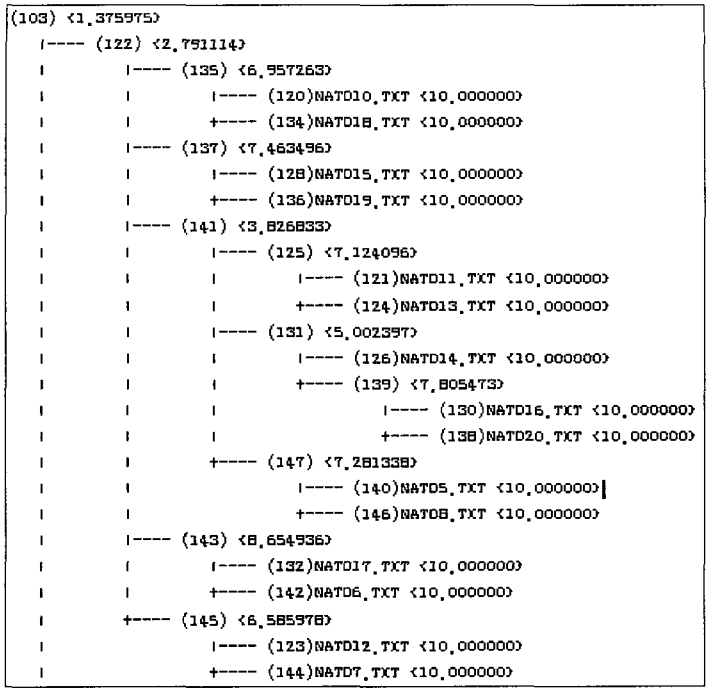
〔그림 4-2〕 Create New (새로운 노드로 분류)

그림 4-2는 새로 입력된 노드 N이 현재 주목하고 있는 노드의 자식노드 가운데 하나로 분류되는 것이 아니라 현재 주목하고 있는 노드의 새로운 자식노드로 분류되는 것을 보인 것인데 분류함수의 C값이 B값 보다 큰 경우에 해당된다.



〔그림 4-3〕 Merge, Split (병합, 분할)

그림 4-3은 현재 주목하고 있는 노드를 기준으로 병합과 분할을 나타내고 있는데 병합에서는 현재 주목하고 있는 노드의 자식노드 A, B(A, B는 각각 분류함수 값이 가장 큰 Best Node와 Second Best Node)를 새로 추가한 하나의 부모노드 아래로 묶는다. 이 때의 분류함수 값은 B가 C보다 크고 M이 P, B보다 큰 경우에 해당된다. 분할에서는 현재 주목하고 있는 노드의 오른쪽 자식노드(Best Node)의 자식노드를 부모노드(Best Node)로부터 분할하여 현재 주목하고 있는 노드의 자식노드로 연결하는 것인데 이 때의 분류함수 값은 B가 C보다 크고 P가 M, B보다 큰 경우에 해당된다. 이상의 4가지 연산자에 의하



〔그림 4-4〕 분류트리

여 COBWEB은 입력자료의 속성(Attribute)에 따라 분류트리를 생성한다. 그림 4-4는 COBWEB 알고리즘에 의해 생성된 분류트리의 예를 보인 것이다.

5. 분류트리의 후처리

5.1. 분류트리 후처리의 필요성

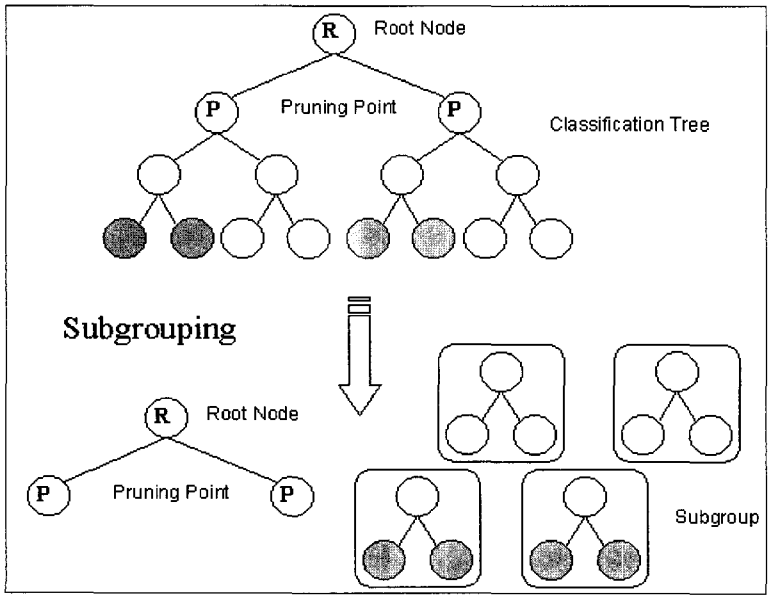
입력 문서들을 대상으로 COBWEB 알고리즘을 수행하여 얻어지는 분류트리는 예제 문서들이 갖는 내용(키워드)에 따라 다양한 형태를 가지며 그것 자체로는 문서들의 유사성에 따른 하나의 트리에 지나지 않으므로 사용자 특성모델 구축에 적용할 수 없다. 따라서 분류트리를 사용자 특성모델 구축에 이용하기 위해서는 분류트리를 해석하여 변환하는 과정이 필요하다. 본 연구에서는 분류트리를 이용하여 문서들을 주제별로 분류한 후 이를 사용자 특성모델 구축에 이용하는 것이 목적이므로 분류트리의 후처리를 통해 트리를 구성하고 있는 문서들을 주제별로 분류하는 데 초점을 맞추고자 한다.

입력된 예제 문서들은 기본적으로 문서들 간의 유사성에 따라 트리의 형태로 분류되었으므로 동일한 부모노드를 갖는 자식노드들에 대응되는 문서들은 항상 그러하지는 않지만 일반적으로 비슷한 주제를 갖고 있는 것으로 간주할 수 있다. 따라서 분류트리를 구성하는 여러 개의 하위트리를 하나의 소그룹으로 인식할 수 있다. 물론 하나의 트리 내에서 하위트리를 분류해 내는 방법은 트리가 가지는 노드의 개수에

비례하여 매우 다양하게 존재한다. 따라서 분류트리로부터 문서들을 각각의 주제별 그룹으로 분류해 내기 위해서는 분류트리를 몇 개의 하위트리로 분류하고, 어떤 노드를 하위트리의 최상위노드로 하느냐를 결정해야 한다. 여기서는 분류트리를 여러 개의 하위트리로 분할하는 과정을 소그룹 분할(Subgrouping)이라 부르기로 한다. 분류트리로부터의 소그룹 분할에 있어서 하나의 소그룹은 서로 다른 주제를 갖는 문서들이 가능한 한 포함되지 않도록 분할되어야하므로 내부에 포함된 단말 노드의 개수는 충분히 작은 수준으로 유지되어야하며, 소그룹의 개수가 많아질수록 그룹간 유사도 비교연산이 복잡해지므로 연산의 복잡성을 악화시키지 않도록 충분히 큰 수준으로 유지되어야 한다. 본 연구에서 제시하고 있는 분류트리의 후처리 과정은 소그룹 분할과 소그룹 병합으로 나뉘는데 소그룹 분할과정에서 발생한 내부잡음(Internal Noise)는 병합과정을 거친다 하더라도 여전히 내부잡음으로 작용하게 되므로 병합연산의 복잡도보다는 내부잡음을 최소화하는 방안을 먼저 고려하여야 한다.

5.2. 소그룹 분할방법 I

소그룹 분할이란 COBWEB 알고리즘의 결과로 얻어진 분류트리를 몇 개의 노드로 구성된 하나의 단위로 떼어내는 것을 말한다. 트리에서 하위트리를 분리해 낼 때에는 하나의 노드를 기준으로 해당 노드의 상위연결을 절단하거나 하위연결을 절단하는 방법이 있다. 하나의 노드를 기준으로 상위연결을 해제했을 경우 원래의 트리와 새로 분리된 하나의 트리를 얻을 수 있다. 반면 노드의 하위연결들을 분할했을 경우 노드의 자식 노드 수만큼의 하위트리가 생겨난다. 분할방법 1에서는 특정 노드를 기준으로 해당 노드의 하위연결들을 분할하는 방법을 취하였다.

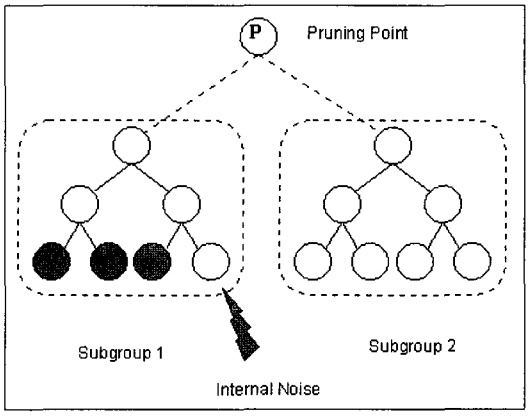


〔그림 5-1〕 분류트리의 분할 (Subgrouping)

구체적인 분할 방법과 기준은 다음과 같다.

- 루트 노드부터 소그룹 분할의 기준이 되는 노드를 탐색한다.
- 소그룹 분할의 기준이 되는 노드를 찾았으면 노드의 하위연결을 절단하여 노드가 가지는 각 자식 노드들을 루트로 하는 새로운 서브트리들이 생기는데 이러한 각각의 서브트리들을 하나의 소그룹으로 인식한다.
- 하나의 노드에 대한 분할이 완료되었으면 해당 노드가 가지는 형제 노드로 이동한다.
- 현재의 노드가 분할 기준에 적합하지 않으면 해당 노드의 첫 번째 자식노드로 이동한다.
- 분할 기준은 현재의 노드에서 분할하였을 때 생성되는 소그룹들 각각에 속한 단말노드가 N값보다 큰가를 살펴보고 크다면 분할할 수 있는데 이러한 경우에 루트노드는 항상 분할 가능하므로 현재 주목하고 있는 노드에서 분할 가능하더라도 바로 분할에 들어가지 않고 자식노드로 이동하여 자식노드에서는 분할할 수 없는 경우에 한하여 분할조건이 성립하는 것으로 한다.

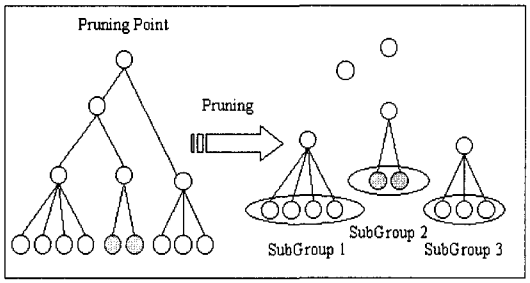
분할방법 1에서는 분할 기준에 사용된 단말노드의 최소 개수인 N값을 2로 설정하였다. N값을 설정하는 이유는 소그룹의 개수가 현저하게 증가하는 것을 막기 위한 것인데 소그룹의 개수가 증가하면 소그룹 병합과정에서의 각 소그룹간 유사도 비교연산의 복잡도가 증가한다. 이러한 연산상의 부하를 줄이기 위해서는 소그룹의 개수가 필요이상으로 증가하는 것을 방지할 필요가 있다. 따라서 N값은 소그룹의 개수가 현저히 증가하지 않도록 충분히 큰 값이 선택되어야 한다.



〔그림 5-2〕 내부잡음 (Internal Noise)

이상의 기준과 절차에 의하여 그림 5-1의 분류트리에 대해 N값을 2로 설정하고 분할했을 때 먼저 루트노드 P에서 분할을 한다면 루트노드 P가 갖는 자식노드가 2개이므로 2개의 소그룹이 생기고 각 소그룹이 갖는 단말노드는 4개씩이므로 소그룹원소의 하한 설정 값인 2보다 크므로 분할할 수 있다. 그런데 마지막 조건에서 자식노드로 이동하여 분할했을 때 더 이상 분할할 수 없으면 현재 노드에서 분할한다고 하였으므로 루트노드에서 분할 가능하다고 하여 바로 분할해서는 안 된다. 루트노드의 자식노드인 P로 이동하여 분할조건을 검색해 보면 역시 소그룹 원소의 개수가 각각 2로 N과 같다. 따라서 노드 P에서도 분할 가능하므로 루트노드는 분할 대상에서 제외한다. 같은 방식으로 노드 P의 자식노드로 이동하여 분할 조건을 검색해 보면 노드 P의 자식노드에서 분할을 했을 경우 단말노드가 하나의 소그룹으로 인식되므로 소그룹

원소의 개수가 2보다 작게되어 노드 P의 자식노드에서는 분할할 수 없다. 따라서 노드 P 에서는 더 이상 자식노드로 진행하여 분할할 수 없으므로 노드 P가 분할 기준이 되는 것이다.



〔그림 5-3〕 잡음이 포함되지 않은 분할

또 한가지 고려하여야할 사항은 N값을 너무 크게 잡으면 분류트리상의 분할기준노드가 단말노드로부터 멀어지고 루트노드에 가까워지기 때문에 소그룹의 개수는 줄어드는 반면 분할된 각 소그룹이 내부잡음(Internal Noise)포함할 가능성이 높아진다. 이러한 내부잡음을 포함하지 않도록 소그룹을 분할하기 위해서는 N값을 내부잡음이 생기지 않을 정도의 충분히 작은 값으로 설정하여야 한다.

그림 5-2는 분할 기준노드를 잘못 선정하여 내부잡음이 포함된 것을 보이고 있고 그림 5-3은 적절한 분할 기준노드 선정으로 내부잡음을 포함하지 않은 분할을 나타내고 있다.

다음은 소그룹 분할 알고리즘을 의사코드(Pseudo Code)로 기술한 것이다. 알고리즘은 두개의 서브루틴으로 나눌 수 있는데 분할 조건을 검사하는 PruningCondition부분과 루트노드에서 출발하여 전체 트리를 대상으로 분할 조건을 검사하는 SubGrouping이 그것이다.

```
Function BOOL PruningCondition( Node n )
Begin
    Node next;

    next = n.child;
    while( next != NULL )
    Begin
        if( next.TotalTerminal <= 2 )
            return( TRUE );
    End.

    return( FALSE );
End.

Procedure SubGrouping( Node root )
Begin
    Node n;

    n = root;
    while( n != NULL )
    Begin
        if( PruningCondition( n ) )
        Begin
            Pruning( n );
            n = n.sibling;
        End.
        else
            n = n.child;
    End.
End.
```

의사코드에 관해 부연설명을 붙이자면, 먼저 SubGrouping 루틴은 루트 노드에서 시작하여 전체 트리를 대상으로 분할 조건(Pruning Condition)을

검사한다. 만일 분할 조건이 만족되면 현재 노드의 하위연결을 제거한다. 이 때 현재의 노드가 가지고 있던 자식 노드의 개수 즉 하위연결의 개수만큼 소그룹이 생겨난다. 현재의 노드를 대상으로 분할이 완료되었으면 이웃한 형제 노드로 이동하여 같은 과정을 반복한다. 만일 분할 조건이 만족되지 않으면 무조건 자식노드로 내려가 분할 조건을 반복 검사한다. 여기서 분할 조건은 현재의 노드가 가지는 인접한 자식 노드들이 가지는 단말 노드의 개수가 N값(2)보다 작은지를 판단하는 것이므로 현재의 노드가 분할 조건에 맞지 않아 자식 노드로 포인터를 이동해 감에 따라 트리의 깊이가 유한하다면 반드시 조건에 맞는 노드를 찾을 수 있다. COBWEB 알고리즘에 의해 생성된 분류트리에 있어서 모든 입력 예제, 즉 문서들은 단말 노드에 위치하게 되므로 이상의 과정을 반복 적용하여 더 이상 형제노드가 존재하지 않는 시점에서는 분류트리 내의 모든 단말 노드가 소그룹으로 분할이 완료된 상태에 있게 된다. 따라서 이상의 알고리즘을 통해 분류트리로부터 적당한 개수와 크기의 소그룹들을 분할 해 낼 수 있게 된다.

5.3. 소그룹 분할방법 II

소그룹 분할 방법에 있어서 입력된 문서들의 개수가 서로 균등하고 각 문서들이 갖는 주제가 확연히 구분되는(예를 들어 자바와 물리학) 경우에는 N값을 높게 설정하여도 COBWEB 알고리즘의 결과로 얻어지는 분류트리 상에서 이미 각 주제의 문서들이 하나의 부모노드 아래에 몰리는 양상을 보이지만 입력된 문서들이 갖는 주제에 있어서 서로 유사한 부분이 존재하거나 하나의 주제가 다른 주제를 포함하고 있는 경우에는 분류트리 내의 문서들이 많은 부분 복잡하게 섞이는 양상을 띤다. 예를 들면 자바와 애플릿 관련 문서의 경우 애플릿은 자바라는 주제에 속한 하부주제이므로 자바와 애플릿간의 구분은 매우 어렵다. 마찬가지로 원자력과 핵무기의 경우에는 Nuclear, Plutonium과 같은 키워드가 동시에 등장하므로 역시 정확한 분류가 쉽지 않다. 이처럼 문서들의 비정형성을 고려할 때 어떠한 경우에도 만족스런 결과를 얻을 수 있는 최적의 분류트리 구축은 본 논문의 범위를 벗어나므로 일반적인 경우에 만족스런 결과를 얻을 수 있는 분류트리의 분할 방식으로는 해당 분류트리를 최대한 잘게 분할하는 것이다.

분류트리를 최대한 잘게 분할하는 방법으로는 루트노드부터 단말노드에 이르기까지 각 노드를 중위 순회하는 과정에서 모든 내부노드를 대상으로 해당 내부노드가 갖는 직계 단말노드들을 하나의 소그룹으로 분할하는 방법을 생각해 볼 수 있다. 이처럼 노드를 최대한 잘게 분할하였을 경우에는 소그룹의 개수가 분할방법 I에 비하여 현저하게 증가하므로 병합연산의 회수가 급격히 증가한다. 분할방법 2에서는 앞서 언급한 바와 같이 키워드가 서로 중복되는 주제 또는 하나의 주제가 다른 주제를 포함하고 있는 경우에 내부잡음을 줄이기 위해 분류트리를 최대한 잘게 분할하는 방법을 사용하고 있다.

5.4. 소그룹간 유사도 측정 및 병합

소그룹분할 알고리즘에 의해 분류트리로부터 얻어진 여러 개의 소그룹들은 각각에 포함된 단말 노드가 가리키는 문서들이 가지는 키워드들을 토대로 유사도 비교연산에 의해 몇 개의 그

룹으로 최종 분류되어 사용자 특성모델 구축에 이용된다. 유사도 비교 연산은 각 소그룹이 가지는 대표 키워드 벡터를 이용하는데, 소그룹 Gs1, Gs2가 각각 v1 = {K1, K2... Kn}, v2 = { L1, L2... Lm }의 키워드 벡터를 갖는다면 다음 공식에 의해 유사도를 비교할 수 있다.

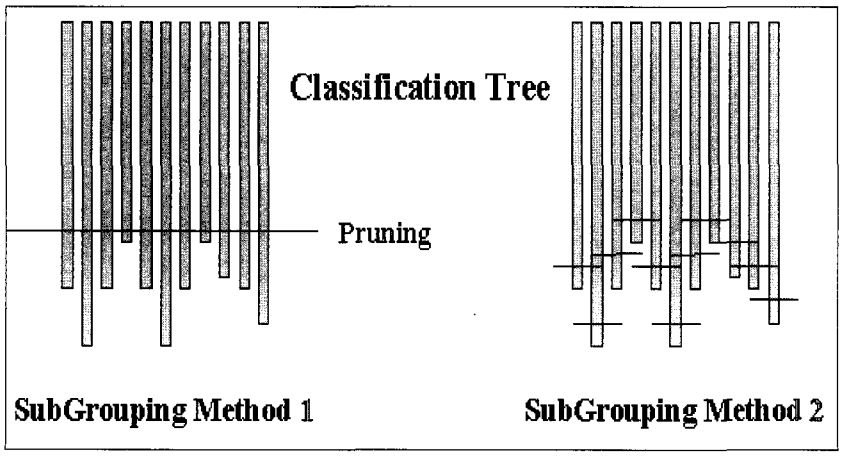
$$\sum_{i=1}^{n}(K_i * L_i)$$

이 때 유사도 비교연산에 사용된 소그룹의 대표 벡터는 소그룹에 포함된 문서들이 가지는 키워드의 출현빈도(Term Frequency)값에 대한 정규화(Cosine normalization)처리 후 얻어지는 값이 된다. 대표 벡터 내의 원소 Ki를 정규화 한 후의 값 Ki'는 다음 공식에 의해 구할 수 있다.

$$K'_i = \frac{K_i}{\sqrt{\sum_{i=0}^{n} K_i^2}}$$

소그룹 간 유사도 비교에 있어서 소그룹의 개수가 n이라 할 때 비교 연산은 n*(n-1)/2 번 수행되어야 한다. 비교 연산 결과 값을 정렬한 후 일정 값 이상을 갖는 소그룹들을 대상으로 큰 값을 가지는 소그룹들을 우선적으로 병합해 나가면 된다. 본 연구에서는 유사도 값이 특정치 이상인 소그룹들을 병합하는 것으로 하였는데 경험적으로 0.3이상인 유사도 값을 갖는 소그룹들을 병합하였을 때 가장 만족스런 결과를 얻을 수 있었다. 구체적인 소그룹 병합 방법은 다음과 같다.

우선 모든 소그룹들을 대상으로 각각의 다른 소그룹들에 대한 유사도 비교 연산을 수행하여 소그룹간 유사도를 구한다. 이 때 값이 0.3 이상인 소그룹들 간에는 Link를 설정한다. 최종적으로 Link가 존재하는 모든 소그룹들은 하나의 그룹으로 병합 처리한다. 만일 소그룹 A,B,C가 존재한다고 가정하고 유사도 비교연산을 SimVal(x, y)라 하자 SimVal(A, B)가 0.35, SimVal(B, C)가 0.28, SimVal(A, C)가 0.4라 한다면 소그룹 B, C는 유사도 값이 0.28로 기준치인 0.3이하임에도 불구하고 소그룹 A와의 유사도 값이 각각 임계값 이상이므로 A에 의해 A, B, C는 하나의 그룹으로 병합된다.



〔그림 5-4〕 분류트리 분할방식

이상의 방식대로 병합을 수행하였을 때 발생할 수 있는 문제는 유사도 링크(Link)가 여러 개 존재하면 할수록 하나로 묶인 그룹 내에 외부잡음 (External Noise)이 포함될 가능성이 커진다. 외부잡음이란 소그룹 병합과정에 있어서 서로 이질적인 소그룹이 하나의 그룹으로 병합되어 소그룹 자체가 잡음으로 작용하는 경우를 말한다. 예를 들어 유사도 링크가 5개 존재하는데 링크를 사이에 둔 두 개의 인접한 소그룹 간에는 임계값 이상의 유사도를 인정할 수 있으나 링크의 처음과 끝에 해당되는 소그룹간에는 임계값에 훨씬 못 미치는 유사도가 존재할 경우 잘못된 분류정보를 얻게 된다. 따라서 이 같은 문제점을 해결하기 위해서는 임계값 이상의 링크를 무조건 하나의 그룹으로 병합해서는 안 된다.

또 한 가지 문제점은 임계값 설정에 있다. 병합에 사용되는 임계값은 너무 낮게 설정하였을 경우에는 이상에서 언급한 바와 마찬가지로 외부잡음이 발생하게 되고 임계값을 너무 높게 설정하였을 때에는 서로 동일한 분야의 주제에 해당하는 소그룹임에도 불구하고 하나의 그룹으로 병합되지 않는 단편화(Fragmentation)가 발생한다.

5.5. 소그룹 병합 임계값

분류트리의 병합에 있어서 무엇보다도 외부잡음은 문서분류 결과에 가장 치명적으로 작용하는 부분이므로 최우선적으로 고려되어야 한다. 본 연구에서는 외부잡음을 최소화하기 위하여 임계값을 충분히 높인 후 그룹의 단편화를 제거할 수 있는 방법을 사용하였다.

소그룹 병합에 있어서 임계값을 0.1등으로 너무 낮게 설정하면 서로 다른 소그룹이 하나의 그룹으로 병합되는 문제가 있고, 임계값을 0.5등으로 너무 높게 설정하면 병합에서 누락되는 문서들이 많아진다. 제 7장 실험의 <표 7-1>과 <표 7-3>을 비교해 보면 임계값을 0.3으로 설정하였을 때 가장 만족스런 결과를 얻을 수 있음을 알 수 있다.

소그룹 병합과정에 있어서 먼저 임계값을 0.3으로 설정하여 각 소그룹들을 대상으로 유사도 비교연산을 수행하여 1차 병합을 수행한다. 경험적으로 보았을 때 임계값을 0.3으로 설정하면 외부잡음을 낮출 수 있었다. 1차 병합에서는 상당히 높은 유사도(내부적으로 키워드의 분포가 매우 유사한) 소그룹들이 하나의 그룹으로 병합된다. 그러나 문서에 따라 동일한 주제를 갖는 문서라 하더라도 해당 주제가 갖는 특성상 유사한 키워드가 많지 않거나 주제 자체의 범위가 범용성을 많이 띨수록 하나의 그룹으로 병합되어야할 많은 문서들이 1차 병합에서 임계값을 넘지 못하고 병합에서 제외된다. 이렇게 병합에서 제외된 소그룹의 비율이 높지 않을 경우에는 해당 소그룹을 병합에서 무시해버릴 수 있지만 높은 비율의 소그룹들이 1차 병합에서 제외되었을 때에는 제외된 모든 소그룹들을 무시할 수 없다. 따라서 1차 병합에서 제외된 소그룹이라 하더라도 해당 소그룹이 어떠한 소그룹과 가장 유사한 값을 갖는지에 따라 1차 병합 후 별도 처리를 수행해 주어야 한다. 각 소그룹은 유사도 비교연산 과정에서 자신과 가장 높은 유사도를 보인 소그룹과 그때의 유사도 값을 별도로 저장해 둔다. 가장 높은 유사도 값이 임계값 이상일 때에는 유사도 링크에 따라 정상적으로 병합되고 임계값 이하일 때에는 유사도 링크를 생

성하지 않고 유사도 값이 임계값의 일정비율 이상(예를 들면 2/3)일 때 자신과 가장 유사한 소그룹 혹은 가장 유사한 소그룹이 속한 그룹에 병합된다. 이 같은 방법을 사용함으로 해서 다중 링크 상에서 발생될 수 있는 외부잡음을 줄일 수 있고 필요 이상으로 임계값을 낮추지 않을 수 있다.

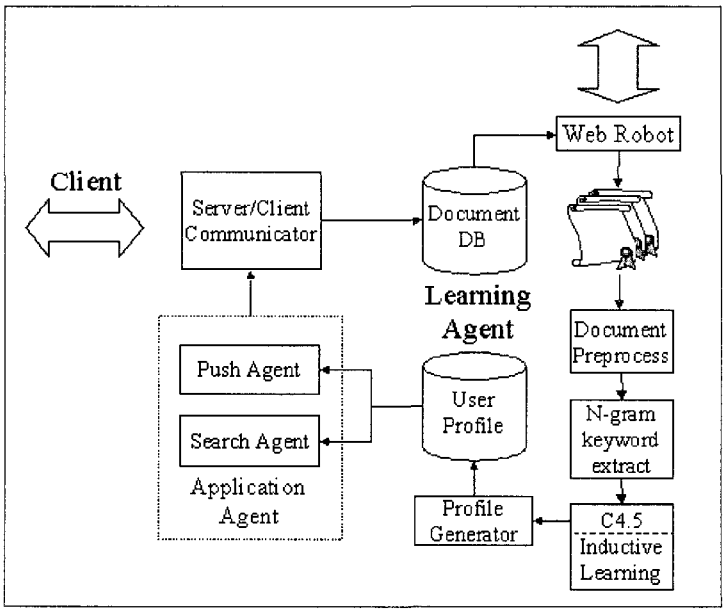
그림 5-4에서는 분할 노드를 기준으로 하위 연결을 일괄 해제하는 분할방법 1과 모든 내부 노드가 갖는 직계 단말노드를 하나의 소그룹으로 인식하는 분할방법 2를 나타내고 있다.

분할방법 1에서는 분류트리가 3개의 소그룹으로 분할되는 반면 분할방법 2에서는 11개의 소그룹으로 분할된다.

6. 문서분류 모듈의 구현

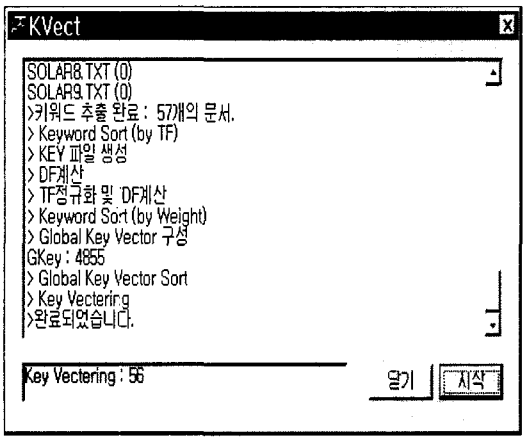
본 연구에서 구현하고 있는 문서분류 모듈은 독립된 프로그램 형태로 웹 에이전트에 연결된다. 웹 에이전트 시스템은 예약된 학습시간이 되면 이전 학습시간부터 현재까지 첨삭된 사용자들 선호(Preference)를 토대로 웹 로봇으로 하여금 지정된 URL의 문서들을 디스크로 전송해 온 후 해당 문서들을 주제별로 분류하기 위하여 문서분류 모듈을 실행(Launch)한다.

웹 에이전트 시스템을 이용하는 사용자들의 관심분야는 서로 다르기 때문에 웹 에이전트는 사용자마다 별로의 특성모델을 유지하며 각 사용자에 대해 그에 따른 서비스를 제공한다. 문서분류 모듈은 각 사용자별로 수집된 HTML문서들을 대상으로 분류작업을 수행하게 되는데 이 때 처리하여야할 사용자 정보는 웹 에이전트로부터 전달받는다.



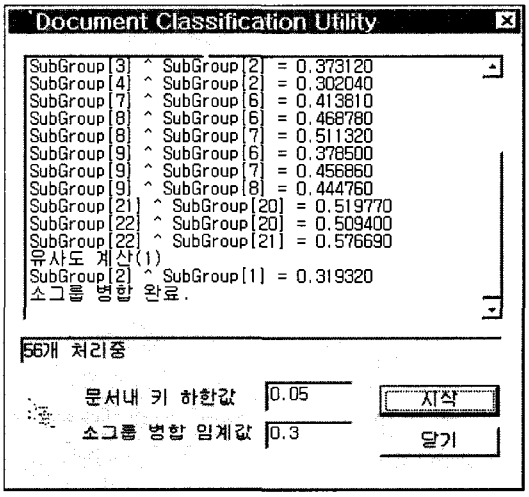
〔그림 6-1〕 웹 에이전트 시스템

그림 6-1은 웹 에이전트 시스템과 문서분류 모듈간의 연결관계를 나타낸 것이다. 먼저 Client로부터 전달된 선호(Preference)정보는 웹 에이전트 시스템의 Document DB에 저장된다. Web Robot은 Document DB에 저장된 URL로부터 사용자가 관심이 있다고 지정한 HTML문서를 수집한 후 문서분류 모듈을 호출한다. 문서분류 모듈에서는 웹 에이전트 시스템으로부터 처리하여야할 사용자 리스트를 넘겨받아 웹 로봇이 수집해 온 문서들을 주제별로 분류한 후 종료한다. 웹 에이전트의 특성모델 생성기에서는 문서분류 모듈에 의해 사용자별로 분류된 카테고리 정보를 토대로 각 사용자의 특성모델(User Profile)을 구축하여 Push Agent로 하여금 비슷한 주제의 정보를 사용자에게 메일로 전달하도록 할 수 있다.



〔그림 6-2〕 키워드 추출

문서분류 모듈은 사용자가 관심을 가지는 HTML문서로부터 의미 있는 키워드를 추출하여 COBWEB 알고리즘에 사용할 키워드 벡터를 생성하는 부분과 이러한 키워드 벡터를 바탕으로 COBWEB 알고리즘을 통해 분류트리를 얻어내고 얻어진 분류트리를 후처리하여 최종 분류정보를 웹 에이전트의 특성모델 추출기(Profiler)에 전달하는 두 부분으로 나뉘어지는데 키워드 추출에 사용된 방법은 정보검색 분야에서 일반적으로 사용하는 불용어 제거, 복수 및 과거형처리, 조사제거 등의 과정을 거친다.



〔그림 6-3〕 COBWEB 및 분류트리 후처리

문서분류 모듈의 구현 및 실험 환경은 Pentium II 300MHz의 CPU와 128M바이트의 메모리를 탑재한 IBM 호환기종의 Windows 98 운영체제 하에서 수행되었으며 구현에 사용된 컴파일러는 마이크로 소프트의 Visual C++5.0을 이용하였다.

다음은 웹 에이전트 시스템의 호출에 의해 키워드 추출 모듈과 COBWEB 및 후처리 모듈이 실행되어 문서분류 작업이 진행된 것을 보인 것이다. 키워드 추출 모듈이 독립적으로 실행되었을 때에는 [시작]버튼을 눌러 키워드 추출 작업을 실행할 수 있고 -b 옵션을 주어 웹 에이전트에서 실행되면 [시작]버튼을 누르지 않고 실행되는 즉시 전달된 사용자 리스트에 따라 키워드

추출작업을 수행한다. 키워드 추출이 끝나면 키워드 추출모듈은 후처리 모듈을 -b옵션을 주어 실행시키고 자신은 종료한다. -b 옵션에 의해 실행된 후처리 모듈은 실행된후 즉시 COBWEB 및 후처리 작업을 실행한다.

문서 내 키 하한 값은 다른 문서들에 비해 추출된 키워드 수가 현저히 작은(5%에 미치지 못하는) 문서들은 COBWEB의 입력으로 주어질 키워드 벡터의 대부분 필드를 0으로 채우므로 전체적 성능향상에 부정적인 영향을 미친다. 따라서 키워드의 개수가 다른 문서들에 비해 현저히 작은 문서들은 분류트리의 생성과정에서 제외된다. 본 연구에서는 그림 6-3에서처럼 문서내 키 하한 값을 5%로 하고 소그룹 병합의 임계값을 0.3으로 설정하였다.

7. 실 험

<표 7-1>은 이상에서 설명된 분류트리의 생성과 분할 및 병합과정에 대한 실험 및 실험결과로 생성된 문서그룹들을 대상으로 사용자 특성모델을 추출한 결과이다. 실험은 임계값을 0.3으로 설정하였으며 모두 10명의 사용자를 대상으로 실시하였으며 각 사용자 당 3-4개의 Category를 입력하였고 각 Category 당 10-20개의 문서를 입력하였다. User 0의 경우를 예로 들면 입력 문서로는 BITMAP 16개, 애견 10개, 폴로 17개로 모두 43개의 문서들이 입력되었고 분류 결과로는 그룹 1이 BITMAP 16개와 폴로 2개, 그룹 2가 애견 8개, 그룹 3이 폴로 6개로 각각 분류되었다. 그룹 1에 대한 정확도는 16/18로 0.88이 나왔고 잡음비율은 2/18로 0.12가 나왔다. 누락비율은 그룹 1의 대표문서인 비트맵이 하나도 누락되지 않고 모두 분류되었으므로 0.0이다. 이 같은 방식으로 USER 0부터 USER 9까지의 분류정보에 대한 평균을 구해보면 정확도 72.2%, 잡음비율 4.2%, 누락비율 20.2%의 결과를 얻을 수 있었다.

<표 7-3>과 <표 7-4>는 각각 임계값을 0.1과 0.5로 설정하였을 때 0.3과 비교하여 상대적으로 부적절한 결과가 나왔음을 보인 것이다.

〈표 7-1〉 분류결과 (병합 임계값 0.3)

User/ 문서개수	입력 문서	그룹	대표 문서	정확도 %	잡음 비율 %	누락 비율 %
User 0 (33)	BITMAP (16) 애견 (10) 폴로 (17)	BITMAP (16) 폴로 (2)	BITMAP	88	12	0
		애견 (8)	애견	80	0	20
		폴로 (6)	폴로	35	0	52
User 1 (70)	China (16) Sports (20) AMD-CPU (21) Solar (13)	AMD (21) Solar (1) Sports (1)	AMD CPU	91	9	0
		Sports (19)	Sports	95	0	5
		Solar (11)	Solar	84	0	16
		China (6)	China	37	0	73
User 2 (52)	Missile (18) NATO (20) Nuclear (14)	NATO (19)	NATO	95	0	5
		Nuclear (10)	Nuclear	71	0	28
		NATO (1) Missile (12)	Missile	66	7	33
User 3 (29)	Bitmap (14) Compress (6) Marx (9)	Compress (3) Marx (2)	Compress	50	40	50
		Bitmap (14)	Bitmap	100	0	0
		Marx (5)	Marx	55	0	45
User 4 (23)	Blood (4) 공자 (9) 유아 (10)	유아 (6)	유아	60	0	40
		Blood (4) 유아 (2)	Blood	66	34	0
		공자 (5)	공자	55	0	45
User 5 (69)	Golf (24) 물리학 (21) Agent (24)	Golf (24)	Golf	100	0	0
		물리 (4) Agent (19)	Agent	79	21	16
		물리 (7) Agent (1)	물리	33	12	47
User 6 (31)	Java (11) Swim (11) Venture (10)	테러 (14)	테러	100	0	0
		Java (11)	Java	100	0	0
		PCS (8)	PCS	100	0	0
User 7 (33)	PCS (8) 보험 (11) 테러 (14)	테러 (14)	테러	100	0	0
		보험 (9)	보험	81	0	11
		PCS (8)	PCS	100	0	0
User 8 (80)	California (30) 병렬처리 (30) Sports (20)	병렬처리 (22)	병렬처리	73	0	0
		Sports (20)	Sports	100	0	0
		California (28)	California	93	0	0
		*** **병렬처리 (8) 단편화 발생 *****				
User 9 (28)	Caesar (11) Motor (8) 패러글라이딩 (9)	Motor (4)	Motor	50	0	50
		Caesar (9)	Caesar	81	0	19
		패러글라이딩 (4)	패러글라이딩	44	0	56
평균				72.2	4.2	20.2

〔표 7-2〕 사용자 특성모델

User/ 문서개수	입력 문서	그룹	대표 문서	Profile
User 0 (43)	BITMAP (16) 애견 (10) 폴로 (17)	BITMAP (16) 폴로 (2)	BITMAP	dib , bitmap , image
		애견 (8)	애견	애견
		폴로 (6)	폴로	
User 1 (70)	China (16) Sports (20) AMD-CPU (21) Solar (13)	AMD (21) Solar (1) Sports (1)	AMD CPU	cpu , k6 , ii , mmx
		Sports (19)	Sports	bull , jordan , final
		Solar (11)	Solar	solar , battery ,
		China (6)	China	
User 2 (52)	Missile (18) NATO (20) Nuclear (14)	NATO (19)	NATO	
		Nuclear (10)	Nuclear	nuclear physics ,
		NATO (1) Missile (12)	Missile	
User 3 (29)	Bitmap (14) Compress (6) Marx (9)	Compress (3) Marx (2)	Compress	data compression
		Bitmap (14)	Bitmap	bitmap , color ,
		Marx (5)	Marx	
User 4 (23)	Blood (4) 공자 (9) 유아 (10)	유아 (6)	유아	유아 , 연구
		Blood (4) 유아 (2)	Blood	혈액 , 헌혈 , 수혈
		공자 (5)	공자	공자 , 제나라
User 5 (69)	Golf (24) 물리학 (21) Agent (24)	Golf (24)	Golf	hole , win , golf
		물리 (4) Agent (19)	Agent	information , user ,
		물리 (7) Agent (1)	물리	physics , information , theoretical
User 6 (31)	Java (11) Swim (11) Venture (10)	Swim (10)	Swim	물 , 동작 , 팔 , 무릎
		Java (11)	Java	url , java , http
		Venture (7)	Venture	의원
User 7 (33)	PCS (8) 보험 (11) 테러 (14)	테러 (14)	테러	
		보험 (9)	보험	insurance , business
		PCS (8)	PCS	서비스 , sk텔레콤 ,
User 8 (80)	California (30) 병렬처리 (30) Sports (20)	병렬처리 (22)	병렬처리	parallel , program ,
		Sports (20)	Sports	bull , jordan , final
		California (28)	California	california , gold , san
		병렬처리 (8) 단편화 발생		architecture , system
User 9 (28)	Caesar (11) Motor (8) 패러글라이딩 (9)	Motor (4)	Motor	자동차
		Caesar (9)	Caesar	caesar , rome , roman
		패러글라이딩 (4)	패러글라이딩	날개 , 형태 , 비행

〔표 7-3〕 분류결과 (병합 임계값 0.1)

User	그룹	대표문서	정확도 %	잡음비율 %	누락비율 %
0 (43)	애견 (10)	애견	100	0	0
	폴로 (15) BITMAP(16)	BITMAP	51	49	0
1 (70)	China (4)	China	25	0	37
	AMD (21) China(6) Solar(12) Sports(20)	AMD	35	65	0
2 (52)	Missile (18) NATO (20) Nuclear (14)	NATO	38	62	0
3 (29)	BITMAP (14) Marx (8) Compress (6)	BITMAP	48	52	0
4 (23)	유아 (10) Blood (4) 공자 (9)	유아	43	57	0
5 (69)	골프 (24)	골프	100	0	0
	물리 (20) Agent (24)	Agent	54	46	0
6 (31)	Swim (11) Venture (3)	Swim	78	22	0
	Java (11)	Java	100	0	0
	Venture (7)	Venture	70	0	0
7 (33)	보험 (11) 테러 (14)	테러	56	44	0
	PCS (8)	PCS	100	0	0
8 (80)	California (30) 병렬처리 (1)	California	96	4	0
	병렬처리 (28)	병렬처리	93	0	3
	스포츠 (20)	스포츠	100	0	0
9 (28)	패러글라이딩 (9)	패러글라이딩	100	0	0
	Motor (7)	Motor	87	0	13
평균			44	12	1.7

〈표 7-4〉 분류결과 (병합 임계값 0.5)

User	그룹	대표문서	정확도 %	잡음비율 %	누락비율 %
0 (43)	BITMAP(14)	BITMAP	87	0	13
1 (70)	China (6)	China	37	0	63
	Sports (6)	Sports	30	0	
	AMD (10) Solar (1)	AMD	47	9	33
	AMD (4)	AMD	28	0	33
	Solar (4)	Solar	30	0	61
	Sports (4)	Sports	20	0	25
	Sports (5)	Sports	25	0	25
2 (52)	NATO (19)	NATO	95	0	5
	Nuclear (6)	Nuclear	42	0	58
3 (29)	BITMAP (10)	BITMAP	71	0	0
	BITMAP(4)	BITMAP	39	0	0
4 (23)	유아 (5)	유아	50	0	50
5 (69)	골프 (24)	골프	100	0	0
	Agent (7)	Agent	29	0	71
6 (31)	Java (11)	Java	100	0	0
	Swim (4)	Swim	36	0	64
	Venture (5)	Venture	50	0	50
7 (33)	테러 (4)	테러	28	0	50
	PCS (6)	PCS	75	0	25
	보험 (4)	보험	36	0	64
	테러 (3)	테러	21	0	50
8 (80)	Sports (14)	Sports	70	0	10
	California (12)	California	40	0	60
	병렬처리 (4)	병렬처리	13	0	66
	병렬처리 (6)	병렬처리	20	0	66
	Sports (4)	Sports	20	0	10
9 (28)	Motor (4)	Motor	50	0	50
	Caesar (4)	Caesar	36	0	64
평균			43	0	32

8. 결론 및 향후연구

웹 에이전트 시스템은 사용자의 수동적 정보 검색을 자동화하는데 그 의미를 찾을 수 있다. 웹 에이전트 시스템이 사용자의 요구에 부합하는 성능을 내기 위해서는 사용자가 관심을 가지는 문서들에 대한 정확한 분류 및 학습이 밑바탕이 되어야 한다. 사용자가 관심을 갖고 있는 분야가 하나일 때에는 문서들을 주제별로 분류할 필요가 없지만 사용자가 여러 분야에 관심을 가지고 있는 경우에는 사용자 특성모델을 구축하기 전에 문서들을 주제별로 분류하는 작업이 선행되어야 한다. 본 연구에서는 사용자가 선호하는 URL로부터 수집된 문서들을 주제별로 분류하여 웹 에이전트 내의 특성모델 추출기에 전달함으로써 보다 효과적인 특성모델 구축을 위한 방법을 제시하였다.

보다 깊은 연구가 필요한 부분은 후처리 과정에 있어서 분류트리를 분할하는 방법이다. 본 연구에서는 분류트리의 분할에 있어서 2가지 방법을 제시하고 있는데, 첫 번째 방법은 소그룹 원소의 하한 값을 설정하고 더 이상 분할할 수 없는 수준까지 트리를 순회한 후 분할조건에 해당되는 노드를 발견하면 해당 노드의 하위연결을 일괄적으로 해제하여 소그룹들을 생성하는 방법인데 문서들이 갖는 주제가 확연히 구분되고 공통적으로 등장하는 단어가 적을 수 록 생성되는 소그룹의 수를 줄이면서 효과적으로 분할할 수 있는 방법이다. 분할방법 1을 사용하였을 때에는 생성되는 소그룹의 개수가 분할방법 2에 비해 상대적으로 작기 때문에 병합에 필요한 유사도 비교연산의 부하를 줄일 수 있다. 그러나 문서들이 갖는 주제에 있어서 하나의 주제가 다른 주제에 포함되거나 서로 공통적으로 등장하는 키워드가 많을수록 분류트리의 내의 문서들이 복잡하게 섞이는 양상을 보이기 때문에 분할방법 1을 사용하였을 때에는 내부잡음의 가능성이 매우 높아진다. 분할과정에서 발생한 내부잡음은 병합연산을 수행하여도 여전히 잡음으로 작용하고 심지어는 병합연산에 있어서도 외부잡음을 유발하는 심각한 부작용을 낳을 수 있다. 따라서 이처럼 문서들의 주제가 유사하거나 공통적인 키워드가 많은 경우에는 분할방법 2에서 제시한 것처럼 분류트리를 최대한 잘게 분할함으로써 내부잡음을 최소화할 수 있다. 분할방법 2는 분류트리를 최대한 잘게 분할함으로써 분할방법 1에 비해 상대적으로 많은 수의 소그룹들을 생성하여 병합과정에 필요한 유사도 비교연산의 복잡도를 악화시킨다. 다만 유사도 비교연산의 복잡도 증가는 연산과정의 시간이 더 많이 소요된다는 점을 제외하면 소그룹들이 필요이상으로 잘게 분할된다고 하여도 병합과정에서 유사한 소그룹들은 다시 하나의 그룹으로 병합될 것이므로 큰 문제가 되지는 않는다. 본 연구에서 제시하고 있는 분류트리의 분할방법 1, 2는 각기 장단점을 가지고 있다. 이상에서 언급한 분할된 소그룹의 내부잡음을 낮추고 필요이상으로 소그룹들을 과도하게 분할하지 않는 적절한 방법이 연구되어야할 것이다.

본 연구에서는 단순히 문서가 포함하는 키워드의 출현빈도(TFIDF)만을 토대로 문서간 유사도 비교연산을 수행하였다. 키워드의 출현빈도에 전적으로 의존한 문서분류가 가지는 최대의 단점은 각각의 주제가 서로 판이하게 구분되는 (예를 들면 음악과, 컴파일러) 경우에는 만족스런 결과를 기대할 수 있지만 농구나 축구처럼

스포츠라는 또 다른 범주로 묶이거나 키워드를 공유하는 주제의 처리에 있어서는 더 많은 내부 잡음 및 외부잡음이 발생하게 된다. 따라서 비슷한 주제에 있어서의 내부잡음을 줄이기 위해서는 텍스트 마이닝에서 연구되고 있는 문서 내 숨겨진 유사성 파악, 동시 출현 단어(Co-occurrence term)인식, 관련이 있는 단어들의 그룹은 짧은 간격을 두고 여러 번 반복된다는 어의적 유사성(lexical affinity)처리(예를 들면 팔레스타인, 협상, 이스라엘 같은 단어들은 한 문단 내에 여러 번 등장하는 단어들이다, 이러한 단어들이 등장하면 중동문제에 관련된 문서라는 것을 추측할 수 있다.)등으로 연구 범위를 더욱 확대하여 나아가야 할 것이다.

참 고 문 헌

[Helena97] Helena Ahnon & Oskari Heinonen, "Mining in the Phrasal Frontier", University of Helsinki, Department of Computer Science, 1997

[Intelli] IBM Intelligent Miner for Text, http://www.software.ibm.com

[Fisher86] Fisher, D. H., & Langley, P., *"Methods of conceptual clustering and their relation to numerical taxonomy"*, In W. Gale (Ed.), Artificial intelligence and statistics, Reading MA: Addison Wesley, 1986.

[Fisher96] Doug Fisher, *"Interative Optimization and Simplification of Hierarchical Clusterings"*, AI Access foundation and Morgan Kaufmann Publishers, 1996.

[Gennari89] Gennari, J. H., Langley, P., & Fisher, D. H., *"Models of incremental concept formation"*, Artificial Intelligence, 40, pp.11-61, 1989.

[Gluck85] Gluck, M., & Corter, J., *"Information, uncertainty and the utility of categories"*, Proceedings of the Seventh Annual Conference of the Cognitive Science Society (pp. 283-287), Irvine, CA: Lawrence Erlbaum, 1985.

[Joachims96] Thorsten Joachims, *"A Probabilistic Anylysis of the Rocchio Algorithm with TFIDF for Text Categorization"*, March 1996.