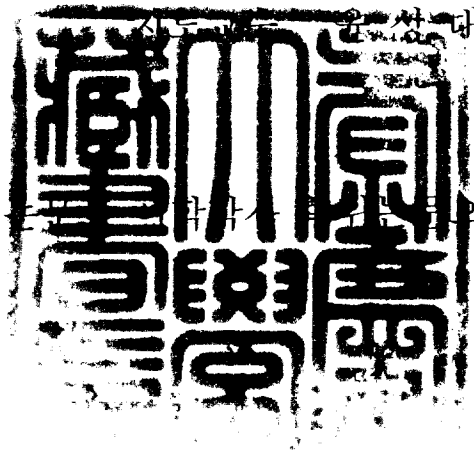
이학박사 학위논문

이행적 폐쇄 트리를 기반으로 한 문서 클러스터링 기법



이 논문을 이학박사 학위논문으로 제출함.

2006년 2월

부경대학교 대학원

전자계산학과

고 석 범

고석범의 이학박사 학위논문을 인준함.

2006년 2월 24일

주 심 공학박사 정 신 일 (인)

위 원 공학박사 여 정 모 (인)

위 원 공학박사 조 우 현 (인)

위 원 이학박사 양 황 규 (인)

위 원 이학박사 윤 성 대 (인)

<차 례>

<그림 차례>

<표 차례>

A Document Clustering Method Based on the Transitive Closure Tree

Seok-Beom Ko

Dept. of Computer Science

Graduate School of Pukyong National University

Abstract

There are various problems that are hard to apply a classification method or a clustering algorithm in a document classification. Because most document classification methods require a post processing or a classification after getting search results from web sites for any keyword.

Among those, two problems are severe. The first problem is the need to categorize the document with the help of the expert. And the second problem is the long processing time the document classification takes.

Therefore we propose a new method of web document

clustering which can decrease the number of times to calculate a document similarity using the Transitive Closure Tree(TCT) and which is able to speed up the processing with a little lose of the precision.

On the other hand, a main property of web resource is to continuously update an information by adding new documents to the web site. Therefore we also propose the web document clustering method that is able to make incrementally a cluster. It is done to insert a new document and to remove a document contained in a cluster.

In simulations, we compare the effectivity of the proposed method with those existing algorithms and present the experimental results.

1. 서론

1990년대 중반 이후부터 웹의 비약적인 성장과 더불어, 웹 문서를 통한 정보의 획득 비율이 높아지고 있다. 대부분의 웹 정보는 인터넷 로봇을 통해 대량의 정보를 보유하고 있는 포털 사이트를 통해서 획득된다. 그러나 검색된 대량의 웹 정보로부터 검색자가 원하는 정보를 획득하는 것은 용이하지 않다[1, 2, 61].

웹을 이용한 정보 수집은 크게 두 가지 특성으로 구분할 수 있다. 첫째는 방대한 검색 결과이다[3]. 즉, 포털사이트에서 제공되는 검색 결과는 검색자가 일일이 살펴보고 분류하기에는 거의 불가능한 대용량이다. 둘째는 중복적인 정보검색이다. 즉, 검색자들은 웹 리소스의 잦은 정보의 갱신을 신뢰함으로써, 주기적으로 동일한 검색어를 사용하여 중복적인 검색을 한다[4]. 첫 번째 특성은 웹 검색 결과를 자동으로 분류하기 위한 웹 문서 분류 알고리즘의 필요성을 의미하며, 두 번째 특성은 이전에 획득한 웹 문서 분류 정보를 이용하는 것이 가능하도록 웹 문서 분류 알고리즘이 점진성(incrementality)을 포함해야 함을 의미한다. 웹 서비스를 제공하는 입장에서도, 홍수처럼 쏟아지는 디지털화된 정보들을 전문가가 일일이 범주를 정하여 분류하는 것은 거의 불가능하다. 그러므로 방대한 양의 문서들을 구조화하여 문서 검색의 효율성을 높일 수 있는 자동화된 문서 분류 기법에 대한 연구가 필요하다.

문서 분류 기법을 통해 관련된 문서들을 그룹화한 것을 클러스

터라고 하며, 이들 클러스터를 생성하는 과정을 클러스터링이라 한다[5]. 효율적인 웹 정보 수집을 위하여 신속하고 정확하게 관련된 문서끼리 분류하는 웹 문서 클러스터링 알고리즘에 대한 연구가 필수적이다[6-8]. 그리고 점진적으로 클러스터를 생성할 수 있는 점진적 문서 클러스터링(incremental document clustering)에 대한 연구도 필요하다. 이것은 문서 클러스터링을 수행하고 나서 차후 새로운 문서의 삽입과 삭제에 대해 미리 저장한 클러스터링 정보를 이용함으로써 중복적인 문서 유사도 계산을 피하기 위한 방법이다[9, 10].

문서 분류 기법에는 크게 문서 범주화(document categorization) 기법과 문서 클러스터링(document clustering) 기법이 있다. 문서 범주화 기법은 전문가에 의해 미리 범주(category)를 정하고, 각 문서들을 임의의 기준 및 계산 결과에 따라서 적절한 범주에 포함시키는 것이다. 반면에 문서 클러스터링 기법은 어떠한 사전적인 범주를 선정하지 않고 단지 문서들간의 유사 정도를 반영하는 문서 유사도를 구하고, 임의의 기준에 따라 자체적으로 클러스터를 형성하는 기법이다[13-15].

문서 범주화 기법은 문서 클러스터링 기법에 비해 정확률(precision) 뿐만 아니라 수행속도도 빠르다. 그러나 문서 범주화 기법은 범주의 선정과 같은 전문가의 개입이 필요하므로, 범주가 다양하고 잦은 갱신이 발생하는 웹 문서 분류에 적용하는 것은 적절하지 않은 문제가 있다[16]. 반면에 문서 클러스터링 기법은 전

문가의 개입과 같은 전제조건을 요구하지 않고 독립적인 분류가 가능하므로 웹 문서 분류에 적절한 기법이다. 그러나 문서 클러스터링 기법은 대체로 문서 범주화 기법보다 정확률 및 수행 속도가 매우 느린 문제가 있다[7, 10].

문서 클러스터링 기법에는 정확률 및 재현율(recall)은 낮으나 수행속도가 빠른 반복적 클러스터링(iterative clustering) 알고리즘과, 역으로 정확률 및 재현율은 높으나 수행 속도가 느린 계층적 클러스터링(hierarchical agglomerative clustering) 알고리즘이 있다[17, 18, 62]. 기존의 문서 클러스터링 기법들은 문서 분류의 질과 수행 속도간의 딜레마(dilemma)가 존재한다. 즉, 문서 분류의 질이 우수한 계층적 클러스터링 알고리즘을 적용하면 과도한 수행 시간 때문에 응답 시간(response time)이 긴 문제가 있고, 수행 시간이 우수한 k-means 알고리즘을 문서 분류에 적용하면 응답 시간은 빠르나 분류의 질이 저조한 문제가 있다.

이러한 문제점을 해결하기위해, 본 논문에서는 웹 문서 분류의 적용에 적절한 문서 클러스터링 기법을 제시한다. 즉, 기존의 분류의 질(quality)을 평가하는 척도인 정확률 및 재현율이 우수한 계층적 클러스터링 알고리즘의 장점을 수용하면서도, 수행 속도를 향상시킬 수 있는 기법을 제안한다. 제안하는 기법의 핵심은 이행적 폐쇄 추론(transitive closure inference)을 도입하여 문서간 유사도 계산 횟수를 대폭적으로 감소시키는 것이다. 그리고 계산 횟수의 감소에 따른 정확률의 손실을 줄이기 위한 다양한 알고리즘

들을 제시한다. 특히, 제안하는 기법에 점진적인 클러스터링이 가능하도록 설계하여, 새로운 문서의 삽입과 삭제에 대하여 신속한 수행이 가능하도록 한다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 제안하는 문서 클러스터링 기법과 관련된 기존의 연구들에 관하여 기술하고, 3장에서는 문서 클러스터링 기법을 제안하기 위해 필요한 요소들과 알고리즘들에 관하여 기술한다. 4장에서는 제안하는 문서 클러스터링 기법의 성능을 평가하기 위하여 기존의 k-means 알고리즘과, 계층적 클러스터링 알고리즘과 여러 가지 척도에 따라 비교 및 분석을 하여 우수성을 입증한다. 끝으로 5장에서는 본 연구에 대한 결론을 기술하고, 향후 과제를 제시한다.

2. 관련 연구

본 장에서는 기존의 문서 분류 기법들에 관하여 기술한다. 특히, 본 논문과 관련된 문서 클러스터링 알고리즘들인 k-means 알고리즘과 계층적 클러스터링 알고리즘에 관하여 자세히 기술한다. 그리고 본 연구에서 유사도 계산 횟수를 줄이기 위하여 핵심적으로 사용되는 이행적 폐쇄 추론에 대하여 기술한다.

2.1 문서 분류 기법의 종류와 특성

방대한 문서 집합으로부터 유사 문서끼리 분류하여 그룹화하는 문서 분류 기법에는 크게 두 가지가 있다.

첫째는 문서 범주화 기법이다. 이 기법은 사전에 전문가에 의해 범주를 구분하여 두었다가, 분류 대상 문서의 요청에 대해 분류기가 가장 적합한 범주에 문서를 포함하는 방식이다. 문서 범주화 기법은 정확성은 높으나, 범주 선정을 위해 지속적인 전문가의 개입이 필요한 문제가 있다[17]. 그러므로 이 기법은 범주가 한정적이고 변화가 적은 도서명 분류와 같은 분야에 적용이 타당하다. 그러나 다양하고 새로운 범주가 자주 발생하는 웹 검색 결과 분류와 같이 후처리(post processing)가 필요한 분류에는 적절하지 않은 기법이다.

두 번째 방식은 문서 클러스터링 기법[13, 15]이다. 문서 클러스

터링 기법은 문서 범주화 기법처럼 사전에 전문가의 개입에 의한 범주 분류를 필요로 하지 않는다. 문서 클러스터링 기법은 각 문서들을 독립적인 개체로 인식하여 각 문서간의 유사도를 측정하여 유사한 문서끼리 그룹화하여 클러스터를 생성하는 것이다. 이 기법의 단점은 낮은 정확률과 방대한 수행시간이다.

문서 범주화 기법과 문서 클러스터링 기법의 특성을 정리하면 <표 1>과 같다[19].

<표 1> 문서 범주화 기법과 문서 클러스터링 기법의 비교

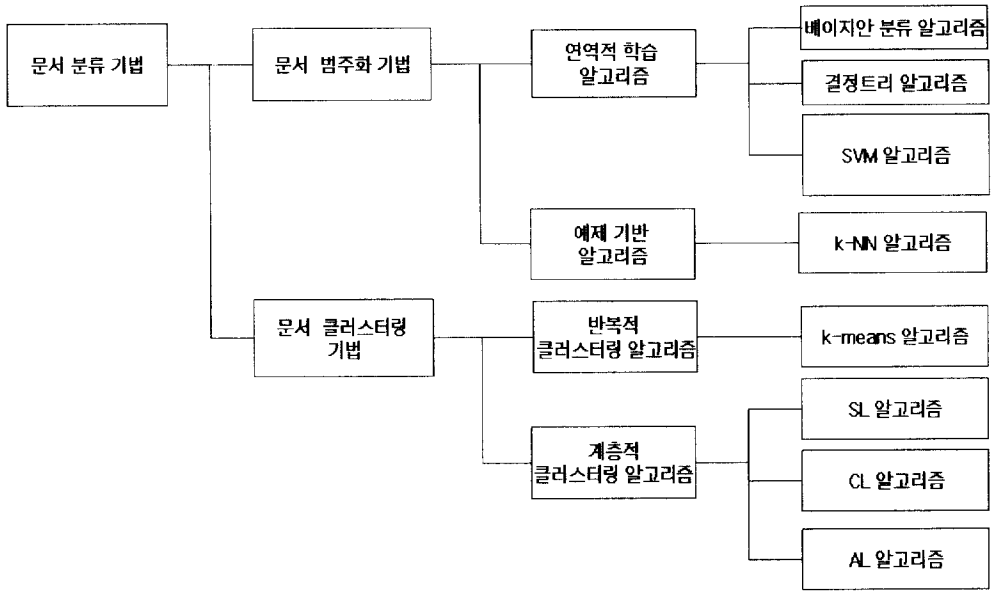
<Table 1> The comparison of the document categorization method with the document clustering method

요소	문서 범주화 기법	문서 클러스터링 기법
전문가 개입 요구	필요	불필요
문서 분류의 질	높다	낮다
수행 속도	고속	저속
정보 갱신의 적응성	난해	용이
유용한 적용 분야 예	도서명 분류	웹 문서 분류

<표 1>과 같이 문서 범주화 기법은 문서 클러스터링 기법에 비하여 문서 분류의 질이 높고, 수행속도도 빠르다. 이 기법은 제한된 범주를 미리 정하여 문서를 할당하며, 알고리즘이 간단하고 계산

횟수도 적기 때문이다. 그러나 문서 범주화 기법은 범주가 다양하고 자주 갱신되는 분야에 대한 적용은 난해하다[19, 20]. 반면에 문서 클러스터링 기법은 자체적으로 문서간의 유사도 계산을 수행하는 특성 때문에 문서 분류의 질도 낮고 수행속도도 느리다는 단점이 있지만, 다양한 범주가 존재하고 잦은 갱신이 일어나는 웹문서 분류에 적절하다[17].

문서 분류 기법에 대한 대표적인 알고리즘들을 (그림 1)에 나타낸다.



(그림 1) 문서 분류 기법을 위한 알고리즘

(Figure 1) Algorithms for the document classification method

2.2 문서 범주화 기법

본 절에서는 문서 범주화 기법으로 활발하게 연구가 진행되고 있는 베이지안 분류 알고리즘(bayesian classifier algorithms)과 k-NN 알고리즘(k-Nearest Neighbor algorithms)에 대하여 기술한다. 두 알고리즘 모두 문서 클러스터링 기법에 비해 정확률이 높고 수행구조도 간단하지만, 대개 베이지안 분류 알고리즘은 k-NN 알고리즘에 비해 정확률은 높지만 수행속도가 느리다는 상호 장단점이 있다.

2.2.1 베이지안 분류 알고리즘

베이지안 분류 알고리즘은 다른 문서 범주화 기법들에 비해 구현이 용이하고 기존의 검색 시스템에 쉽게 적용할 수 있는 장점이 있다[24, 25].

베이지안 분류 알고리즘은 분류하고자 하는 문서를 입력으로 받아 그것이 각 범주에 할당될 확률을 계산하는 방법으로 문서를 분류한다. 문서가 특정 범주에 속하는 확률을 베이즈 이론(Bayes's theorem)을 이용하여 계산한다. 즉, 어떤 문서 d가 주어졌을 때 그 문서가 특정 범주 c에 속한 문서일 확률 $R(c,d)$을 구하는 것으로 그 확률은 베이지언 공식에 따라 식 (2.1)과 같다[64].

$$R(c,d) = \frac{P(d|c)P(c)}{P(d)} \tag{2.1}$$

식 (2.1)을 전확률 공식(total probability formula)에 의해 식 (2.2)로 유도된다[64].

$$P(d) = \sum_{c \in C} P(c)\, P(d|c) \tag{2.2}$$

여기서 C는 범주 c의 집합이며, $P(d)$는 모든 범주에 대하여 같은 값을 가지므로 확률을 계산하는데 고려하지 않아도 된다. 따라서 식 (2.1)의 분자에 위치한 $P(d|c)$와 $P(c)$만 추정하면 문서 d가 범주 c에 할당될 확률을 계산할 수 있다.

베이지안 분류 알고리즘을 이용한 문서분류 시스템은 일반적으로 결정트리 알고리즘 및 k-NN 알고리즘에 비하여 정확률이 상대적으로 높다고 알려져 있다[26].

2.2.2 k-NN 알고리즘

문서 범주화 기법에서 k-NN 알고리즘은 전문가에 의해 미리 분류된 훈련 문서(training document) 집합에서 단어들의 출현 빈도(appearance frequency) 정보를 추출하여 문서 벡터를 형성하고 새로운 문서에 대해 훈련 문서 벡터와의 유사도를 비교하여 k개의 가장 유사한 문서를 정한 후, 이를 이용하여 새로운 문서가 포함될 범주를 예측하는 것이다[27-29, 63]. k-NN 알고리즘은 처리 과정이 간단하여 구현이 용이하다는 장점이 있다[30].

k-NN 알고리즘을 정의하기 위하여, m개의 범주로 구성된 전체 범주를 C라 하고, 각각의 i번째 클러스터를 c_i라 하고, n개의 문서로 구성된 문서집합을 D라 하고, 각각의 문서를 d_i라 하자. 새로운 문서 d_N에 대해 결정되는 k개의 최근접 이웃문서는 d^k라 하면, 범주 c_i에 대하여 d^k의 할당 여부 $c_i(d^k)$는 식 (2.3)과 같다[63].

$$c_i(d^k) = \begin{cases} 1 & \text{if } d^k \in c_i \\ 0 & \text{그외의 경우} \end{cases}, \quad \text{for } i = 1, 2, \cdots, m \tag{2.3}$$

각 범주 c_i에 대하여 새로운 문서 d_N이 속할 정도 w_i를 식 (2.4)와 같이 구한 후, $\max\{w_i\}$인 i 범주에 새로운 문서 d_N을 할당한다.

$$w_i = \sum_{j=1}^{k} c_i(d^{k_j}), \text{ for } i = 1, 2, \cdots, m \quad (2.4)$$

k-NN 알고리즘은 수행구조가 간단하여 신속한 문서 분류를 요구하는 시스템에 자주 적용된다.

2.3 문서 클러스터링 기법

문서 클러스터링 기법은 문서 범주화 기법과는 달리 전문가의 개입을 요구하지 않고 자체적으로 유사한 문서끼리 분류함으로써, 잦은 갱신을 하며 범주가 다양한 웹문서 분류 시스템의 구현에 적절한 기법이다. 문서 클러스터링 기법에는 크게 반복적 클러스터링 알고리즘과 계층적 클러스터링 알고리즘이 있다.

반복적 클러스터링 알고리즘은 비계층적 클러스터링 알고리즘으로서 특정 개수의 클러스터를 미리 정하고, 반복적으로 클러스터의 센트로이드(centroid)를 갱신하면서 관련 있는 문서를 클러스터링 하는 기법이다. 반복적 클러스터링 알고리즘의 대표적인 예는 k-means 알고리즘이 있다[31, 32].

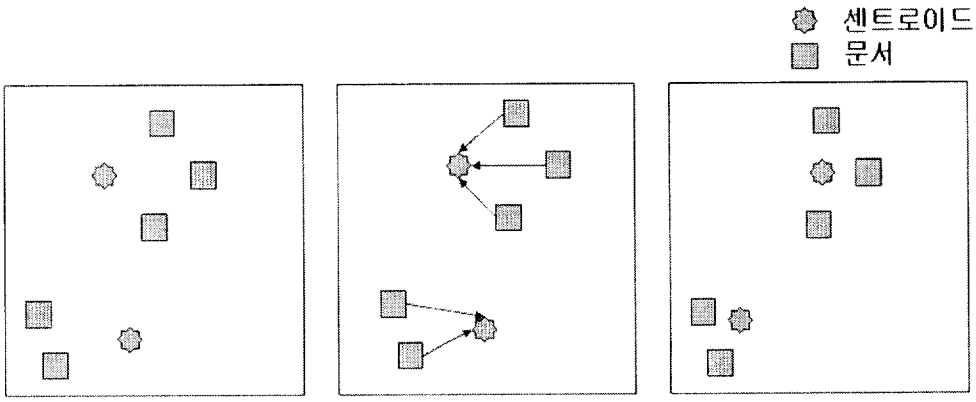
계층적 클러스터링 알고리즘은 하나의 문서를 독립된 클러스터로 간주하여 각 문서간의 유사도를 바탕으로 클러스터들을 합병하는 방법으로서, 클러스터간 유사도 결정 방법에 따라, SL(Single

Link) 알고리즘, CL(Complete Link) 알고리즘, AL(group Average Link) 알고리즘으로 나눈다[17, 62].

대개 계층적 클러스터링 알고리즘은 반복적 클러스터링 알고리즘에 비해 수행속도는 느리지만, 정확률이 높은 특성이 있다.

2.3.1 반복적 클러스터링 알고리즘

반복적 클러스터링 알고리즘의 대표적인 기법은 k-means 알고리즘이다[33-35]. k-means 알고리즘의 주요 생성 과정은 (그림 2)의 (a)와 같이 k개의 초기 센트로이드를 정하고, (그림 2)의 (b)와 같이 각 센트로이드를 중심으로 각 문서와의 거리를 구하여 가까운 센트로이드에 할당하여 클러스터를 생성하고, (그림 2)의 (c)와 같이 생성된 클러스터의 센트로이드를 각각 갱신하여 반복적으로 클러스터링을 수행한다. 그리고 이전의 센트로이드와 갱신된 센트로이드의 차이가 미리 정한 임계치 이하일 때 종료하는 기법이다[65].



(그림 2) k-means 알고리즘의 센트로이드 갱신

(Figure 2) An updating centroid in k-means algorithms

k-means 알고리즘의 가장 큰 장점은 수행 구조가 간단하여 신속한 수행이 가능한 것이다. 초기 센트로이드 개수를 k라하고, 문서의 총수를 n이라 하고, 반복수를 r이라 하면, k-means 알고리즘은 $O(k \cdot n \cdot r)$의 수행 시간이 필요하므로 신속하게 클러스터링 결과를 얻고자 할 때 유용하다. 그러나 계층적 클러스터링 알고리즘에 비해 정확률 및 재현율이 매우 저조하여, 비교적 높은 질의 분류를 원하는 분야에 적용하는 것은 비효율적이다.

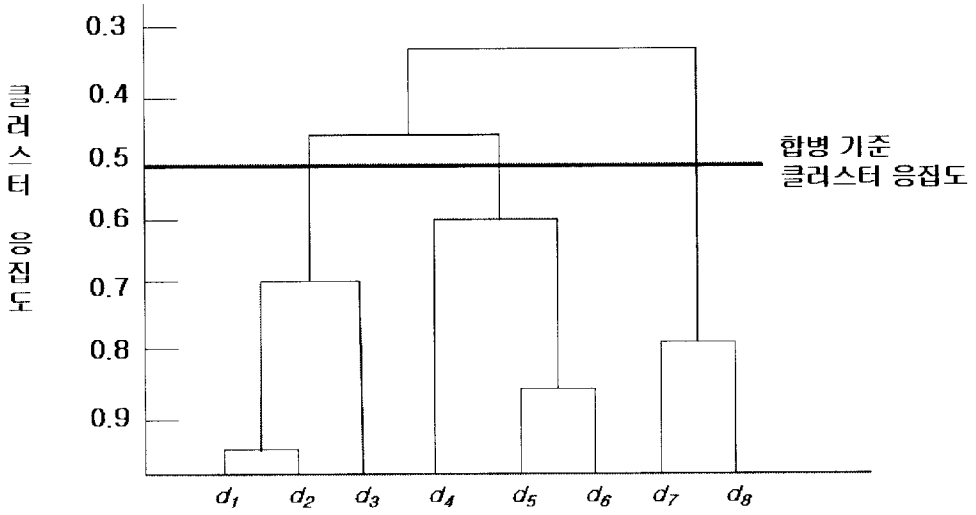
2.3.2 계층적 클러스터링 알고리즘

계층적 클러스터링 알고리즘은 초기에 각 문서들을 하나의 독립된 클러스터로 간주하여, 각 클러스터간의 유사도를 계산하여 점차적으로 클러스터를 합병한다. 그리고 클러스터들은 얼마나 관련있는 문서끼리 합병되었는가에 대한 측정 단위로서 각각의 클러스터 응집도(cluster coherence)를 갖는다. 계층적 클러스터링 알고리즘은 하나로 합병된 클러스터가 존재할 때까지, 클러스터들 중에서 문서간의 임의의 유사도를 만족하고, 하나의 클러스터로 결합했을 때 클러스터 응집도가 가장 높은 두 클러스터를 찾아서 결합하는 것이다.

클러스터링을 하기 위해 주어진 n개의 문서로 구성된 문서 집합을 D라고 하고, 각각의 문서를 d_i라고 하면, 각 단계마다 D의 파티션들을 생성한다. 초기에는 하나의 문서로만 구성된 클러스터 집합인 $\{(d_1), (d_2), \cdots, (d_n)\}$의 파티션으로 클러스터링이 시작되고, 마지막에는 전체 문서 집합인 D로 클러스터링이 종료된다. 이때 클러스터가 생성될 수 있는 병합 기준 클러스터 응집도를 정하여, 임의의 단계에서 클러스터링을 종료할 수 있다.

계층적 클러스터링 알고리즘의 예를 (그림 3)에 나타내었다. (그림 3)에서 계층적 클러스터링 알고리즘은 각 문서간의 유사도를 비교하여 클러스터를 형성하고 나서, 선정되는 병합 기준 클러스터 응집도에 따라서 최종 클러스터를 결정하게 된다. (그림 3)에서

는 병합 기준 클러스터 응집도를 0.5로 정하여, {(d_1, d_2, d_3), (d_4, d_5, d_6), (d_7, d_8)} 3개의 클러스터가 생성된다.



(그림 3) 계층적 클러스터링 알고리즘의 예

(Figure 3) An example of the hierarchical agglomerative clustering algorithm

계층적 클러스터링 알고리즘은 클러스터간의 유사도를 결정하는 방법에 따라 SL 알고리즘, CL 알고리즘, AL 알고리즘으로 구분한다[32].

1) SL 알고리즘

SL 알고리즘에서 클러스터간의 유사도 $SL(c_i,c_j)$는 식 (2.5)와 같

이 클러스터에 포함되어 있는 문서들간의 유사도 중에서 최대 유사도를 클러스터의 유사도로 결정한다[36].

$$SL(C_i, C_j) = \max_{d_i \in C_i, d_j \in C_j} \{\cos(d_i, d_j)\} \tag{2.5}$$

여기서 C_i는 i번째 클러스터이고, d_i는 i번째 문서이며, $\cos(d_i,d_j)$는 문서 d_i와 d_j간의 코사인 유사도이다. $\cos(d_i, d_j)$는 식 (2.6)과 같이 구한다[36, 37].

$$\cos(d_i, d_j) = \frac{d_i \cdot d_j}{|d_i||d_j|} \tag{2.6}$$

여기서 $d_i \cdot d_j$는 단어 수 t에 대한 내적으로서 식 (2.7)과 같이 구한다.

$$d_i \cdot d_j = \sum_{k=1}^{t} d_{ik} \cdot d_{jk} \tag{2.7}$$

그리고 노름(norm) $|d_i|$는 식 (2.8)을 통하여 구한다.

$$|d_i| = \sqrt{d_i \cdot d_i} \tag{2.8}$$

SL 알고리즘은 계층적 클러스터링 알고리즘 중에서 계산 요구량이 가장 적기 때문에 수행속도는 빠르나, 클러스터의 질을 판별하는 정확률 및 재현율은 낮은 단점이 있다. 이것은 SL 알고리즘이 두 클러스터의 소속 문서간의 유사도 중 최대 유사도를 두 클러스터의 유사도로 간주하므로, 두 클러스터가 합병될 확률이 높기 때문이다. 대개 SL 알고리즘의 계산 복잡도는 문서수 n에 대하여 $O(n^2)$ 이상이다.

2) CL 알고리즘

CL 알고리즘에서 클러스터간의 유사도 $CL(c_i,c_j)$는 식 (2.9)와 같이 클러스터에 포함되어 있는 문서들간의 유사도 중에서 최소 유사도를 클러스터의 유사도로 결정한다[36].

$$CL(C_i, C_j) = \min_{d_i \in C_i, d_j \in C_j} \{\cos(d_i, d_j)\} \tag{2.9}$$

CL 알고리즘은 계층적 클러스터링 알고리즘 중 계산량이 가장 많아서 수행속도가 가장 느리지만, 정확률 및 재현율은 가장 우수하다. 이것은 두 클러스터가 합병될 확률이 SL 알고리즘보다 작기 때문에 서로 유사하지 않는 문서가 많이 발생하여 계산량이 더욱 늘어나기 때문이다. 대개 CL 알고리즘의 계산 복잡도는 $O(n^3)$ 이상이다.

3) AL 알고리즘

AL 알고리즘에서 클러스터간의 유사도 $AL(c_i,c_j)$는 식 (2.10)과 같이 클러스터에 포함되어 있는 문서들간의 유사도의 평균을 클러스터의 유사도로 결정한다[36].

$$AL(C_i, C_j) = \frac{1}{n_i n_j} \cdot \sum_{d_i \in C_i,\, d_j \in C_j} \cos(d_i, d_j) \tag{2.10}$$

여기서 n_i는 클러스터 i에 포함된 문서들의 총수이다. AL 알고리즘은 SL 알고리즘과 CL 알고리즘간의 수행속도와 정확률의 딜레마를 절충한 기법이다. 대개 AL 알고리즘의 계산 복잡도는 $O(n^2 \log n)$ 이상이다.

앞에서 언급한 계층적 클러스터링 알고리즘 중에서 SL 알고리즘, AL 알고리즘, CL 알고리즘에 대한 특성 비교를 <표 2>에 정리한다. <표 2>의 시간복잡도에서 n은 문서의 총수이고, k는 k-means 알고리즘에서 초기 클러스터 개수이고, q는 k-means 알고리즘에서 반복수이다. <표 2>와 같이 k-means 알고리즘은 $O(nkq)$의 계산 복잡도로서, 계층적 클러스터링 알고리즘에 비해 수행속도가 현저하게 빠르지만 분류된 문서의 질은 낮다[10]. 계층적 클러스터링 알고리즘에서 AL 알고리즘은 계층적 클러스터링 알고리즘들 중에서 중간적인 성능을 갖는다. 즉, 정확률 및 재현율

면에서는 CL 알고리즘보다는 저조하고 SL 알고리즘보다는 좋으나, 수행 속도에서는 CL 알고리즘보다는 우수하고 SL 알고리즘보다는 저조하다.

<표 2> 문서 클러스터링 기법간의 특성 비교

<Table 2> A property comparison among document clustering methods

기준 / 알고리즘	시간 복잡도	장점	단점
k-means 알고리즘	$O(nkq)$	가장 빠른 수행속도	가장 낮은 질
SL 알고리즘	$O(n^2)$	빠른 수행속도	낮은 질
CL 알고리즘	$O(n^3)$	가장 높은 질	가장 느린 수행속도
AL 알고리즘	$O(n^2\log n)$	CL 알고리즘보다 빠른 수행속도, SL 알고리즘보다 높은 질	CL 알고리즘보다 낮은 질, SL 알고리즘보다 빠른 수행속도

계층적 클러스터링 알고리즘은 미리 정한 클러스터의 개수까지 합병되거나 임의의 클러스터 응집도 이상이 될 때까지 수행한다. 계층적 클러스터링 알고리즘은 클러스터에 속하는 모든 문서간의 유사도를 계산해야 하므로, 반복적 클러스터링 알고리즘에 비해 문서 분류의 질은 우수하나 과도한 수행시간이 걸리는 단점이 있다.

2.4 이행적 폐쇄

이행적 폐쇄는 추론 이론으로서 중복 레코드 제거 기법에서 두 레코드간의 신속한 비교를 위해 적용이 되었다[38, 39]. 이행적 폐쇄 이론의 핵심은 유사도 계산을 통해 레코드 a와 레코드 b가 유사하고, 레코드 b와 레코드 c가 유사하다면, 레코드 a와 c간의 유사도를 계산하지 않고도 레코드 a와 c는 유사하다고 간주하는 것이다. 즉, 데이터베이스에서 중복된 레코드(duplicated record)를 찾아 제거하는 문제에서 레코드간의 유사성을 신속하게 판단하기 위하여 이행적 폐쇄 이론이 적용되었다[38].

그러나 이행적 폐쇄에 따라 유사도를 계산하지 않고 유사성을 추론한다는 것은 그만큼 정확률이 손실된다는 것을 의미한다. 참고문헌 [38]에서는 두 레코드간의 유사성을 판단하기 위한 임계값 뿐만 아니라 CF(Certainty Factor)를 두어 정확률의 손실을 줄이는 기법을 제안하였다.

3. IDC-TCT 기법

본 장에서는 이행적 폐쇄 추론을 이진트리에 접목하여 문서 집합을 구성하는 TCT(Transitive Closure Tree)를 정의하고, 신속한 문서 클러스터링을 수행하기 위한 클러스터 생성 알고리즘(cluster creation algorithm)과 유사도 비교 알고리즘(similarity comparison algorithm)을 정의한다. 그리고 문서 클러스터링 결과를 효율적으로 활용할 수 있는 점진적 문서 클러스터링을 수행하기 위한 알고리즘들을 정의한다. 본 논문에서 제안하는 TCT를 기반으로 한 점진적 문서 클러스터링 기법을 IDC-TCT(Incremental Document Clustering based on the Transitive Closure Tree)라 한다.

3.1 TCT 정의

문서 클러스터링을 수행하면서 대부분의 시간을 소비하는 것은 문서간의 유사도 계산을 위한 것이다. 그러므로 신속한 클러스터링 수행을 위한 관건은 유사도 계산 횟수를 대폭적으로 줄이는 것이며, 이를 위해 본 논문에서는 이행적 폐쇄 추론을 적용한다.

그러나 문서 클러스터링에서 이행적 폐쇄 추론의 적용은 부분적인 문서 유사도 계산을 하기 때문에 클러스터의 질을 감소시키므로 결국 전체적인 분류의 질을 저하시킨다. 이것은 클러스터의 규

모가 커질수록 소속 문서들간의 유사도의 변질이 더욱 커지는 것을 의미한다. 이러한 유사도의 변질을 측정하기 위하여, 클러스터의 규모가 커질 때마다 클러스터 응집도를 계산하여 임의의 클러스터 응집도 이상을 만족하는지 검사한다.

근본적으로, 클러스터의 질적 저하를 최소화하기 위해서는 클러스터들이 항상 일정한 클러스터 응집도를 유지하기 위한 여러 가지 조정이 필요하다. 문서 클러스터링을 수행하면서 수반되는 정확률의 손실을 줄이기 위한 여러 가지 조정에 관해서는 3.3절과 3.4절에서 기술한다.

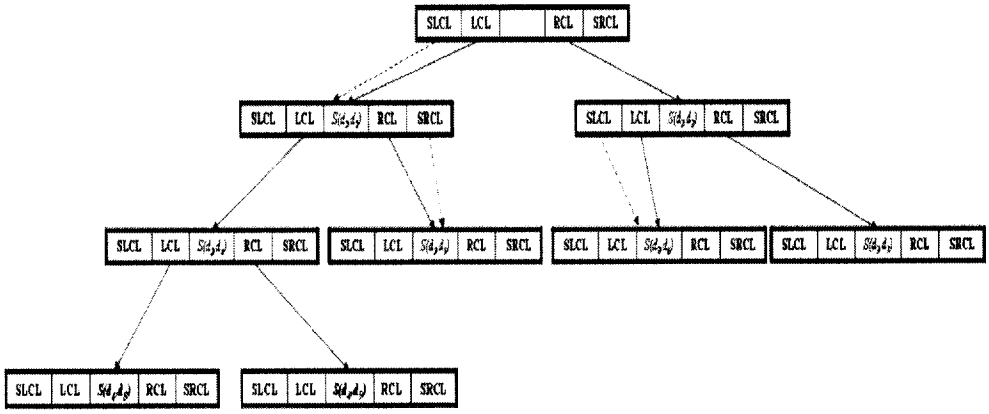
이행적 폐쇄 추론을 이진트리에 적용한 TCT는 노드의 집합 V와 간선의 집합 E로 구성된다. V에서의 각 노드 v_i는 각각의 문서 d_i를 나타내고, E에서의 두 노드간의 링크 e_{ij}는 문서 d_i와 d_j간의 문서 유사도 $S(d_i,d_j)$라고 하면, TCT는 정의 3.1과 같다.

정의 3.1: 이행적 폐쇄 트리 TCT

$$TCT = \{V, E\}$$
$$V = \{v_1, v_2, \cdots, v_n\}$$
$$E = \{e_{12}, e_{13}, e_{24}, e_{25}, \cdots, e_{\{i\}\{2i\}}, e_{\{i\}\{2i+1\}}\}$$

TCT를 저장하기 위한 자료구조는 (그림 4)와 같이 구성한다. (그림 4)와 같이 TCT는 다중 링크 형태로 구성한다. 실선으로 표기된 LCL(Left Child Link)는 왼쪽 자식 노드를 가리키고,

RCL(Right Child Link)는 오른쪽 자식 노드를 가리킨다. 그리고 점선으로 표기된 SLCL(Similar Left Child Link)는 유사한 왼쪽 자식 노드를 가리키고, SRCL(Similar Right Child Link)는 유사한 오른쪽 자식 노드를 가리킨다. 그러므로 SLCL과 SRCL로 연결된 문서 노드들은 서로 유사한 문서 노드의 집합인 클러스터를 형성함을 의미한다.



(그림 4) TCT의 자료구조

(Figure 4) The data structure for TCT

3.2 문서 클러스터링 요소

본 절에서는 IDC-TCT에 필요한 요소 및 수식들에 대하여 기술한다. IDC-TCT를 기술하기 위해 사용되는 요소들을 기술하고, 원

문서로부터 키워드를 추출하는 방법에 대하여 기술한다. 그리고 두 문서가 얼마나 유사한지를 산출하기위한 문서 유사도 수식과 클러스터가 얼마나 유사한 문서들로 구성되었는지를 측정하기 위한 클러스터 응집도 수식을 정의한다.

3.2.1 파라메터 정의

IDC-TCT를 기술하기 위해 사용되는 파라메터들을 (그림 5)에 기술한다.

- D: 문서집합으로 $D=\{d_1, d_2, \cdots, d_n\}$
- d_i: 문서집합 D에서의 i번째 문서
- t_j: 임의의 문서에서의 j번째 단어
- C_i: i번째 클러스터, $C_i \subset D$
- $minS$: 최소 문서 유사도, $0 \leq minS \leq 1$
- $minC$: 최소 클러스터 응집도, $0 \leq minC \leq 1$
- $minMer$: 최소 합병도, $0 \leq minMer \leq 1$
- $minCarSim$: 최소 카테고리 유사도, $0 \leq minCarSim \leq 1$

(그림 5) IDC-TCT를 정의하기 위한 파라메터

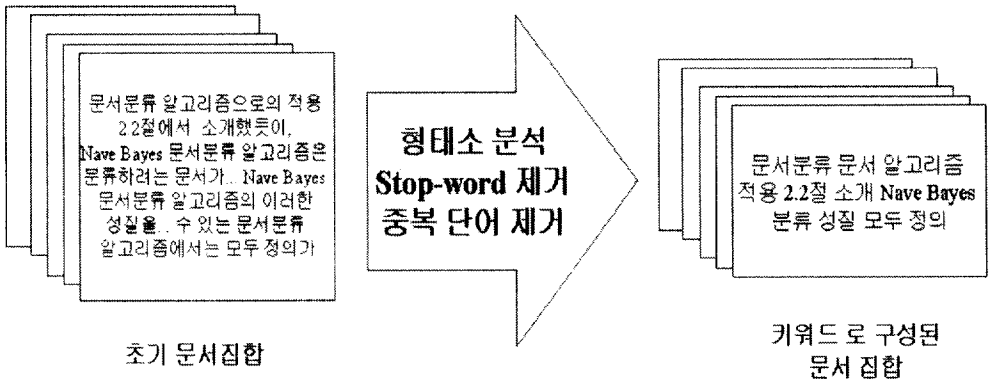
(Figure 5) The parameter to define the IDC-TCT

(그림 5)에서 *minS*는 임의의 두 문서가 유사하다고 판정하기 위한 기준으로 사용되고, *minC*는 클러스터의 생성 여부를 결정하기 위한 기준으로 사용되며, *minMer*는 임의의 두 클러스터의 합병 여부를 결정하기 위한 기준으로 사용되며, *minCarSim*은 통합된 클터스터에 임의의 새로운 문서가 포함되기 위한 기준으로 사용된다.

3.2.2 키워드 추출

각 문서는 문장들로 구성되어 있다. 이 문장들은 형태소 분석을 통해 단어 집합으로 분류하고, 이 단어 집합에서 문서를 특징짓는 단어인 키워드를 추출해야 한다. TCT의 각 문서 노드들은 이러한 키워드의 집합으로 구성된다.

문장으로 이루어진 원 문서로부터 단어 집합을 구성하기 위해서는 형태소 분석 및 불용어 처리과정을 수행하는 형태소 분석기가 필요하다. 이를 위해 본 논문에서는 국민대에서 개발한 형태소 분석기 HAM[66]을 이용한다. HAM은 불필요한 단어 집합인 불용어 목록 사전에 따라 문서에 포함된 불용어를 제거하고, 단어의 다양한 활용형을 하나의 어근으로 처리하는 스테밍 알고리즘을 수행한다. (그림 6)은 HAM을 이용하여 문장으로 이루어진 문서로부터 키워드들로 구성된 문서로 변환하는 예를 나타낸다.



(그림 6) 문서로부터 키워드 추출

(Figure 6) An extraction of keyword from a document

HAM을 수행하여 획득된 단어 집합으로 부터 키워드를 선정하기 위하여 각 단어의 가중치를 구한다. 본 논문에서는 키워드를 선정하기 위하여 보편적으로 널리 사용되는 TFIDF(Term Frequency Inversed Document Frequency) 함수를 적용하여 단어 가중치를 구한다[40-43]. 즉, 문서 d_i에서 단어 t_j의 가중치 w_{ij}는 식 (3.1)과 같다.

$$w_{ij} = TF_{ij} \cdot \ln \frac{n}{IDF_j} \tag{3.1}$$

여기서 TF_{ij}는 문서 d_i에서 단어 t_j의 출현 빈도수이고, IDF_j는 전체 문서 수 n에 대하여 단어 t_j의 출현빈도수이다. 그러므로 임의

의 키워드가 문서에 포함되기 위해서는 해당 문서 내부에서는 빈번하게 출현하고, 다른 문서에서는 가능한 적게 출현해야 한다.

결국 각각의 키워드로 구성된 문서집합들은 TCT로 구성되어 IDC-TCT의 수행을 위한 입력 자료로 사용된다.

3.2.3 문서 유사도

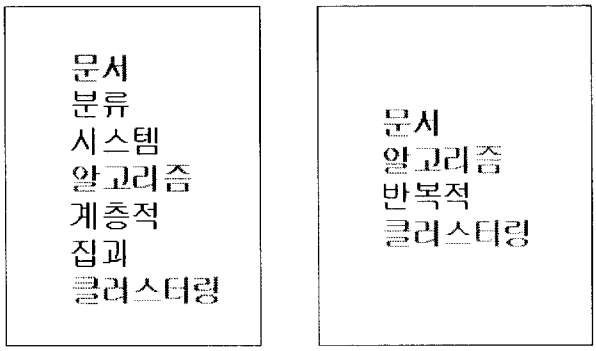
문서 유사도는 두 문서간의 유사 정도를 *0*에서 *1*사이의 값으로 나타낸 것이다. 문서 유사도는 클러스터링을 수행하기 위해 사용되는 기본 함수이다. 그러므로 클러스터링을 위해 소요되는 대부분의 수행시간은 문서 유사도를 계산하기 위한 시간이다. 신속한 클러스터링을 수행하기 위한 핵심은 문서 유사도 횟수를 줄이는 것이다.

문서 d_i와 d_j간의 문서 유사도 $S(d_i, d_j)$를 측정하기 위한 수식은 식 (3.2)와 같다.

$$S(d_i, d_j) = \frac{|d_i \cap d_j|}{\min(|d_i|, |d_j|)} \tag{3.2}$$

여기서 $|d_i|$는 문서 i에 포함된 유일한 단어의 총 개수이고, $|d_i \cap d_j|$는 문서 i와 j에 공통으로 포함된 유일한 단어의 총 개수이다. 분

자는 문서 d_i와 d_j간의 유사 정도를 나타내고, 분모는 문서에 포함된 키워드의 개수가 작은 문서를 기준으로 유사도를 측정함으로써, 키워드가 많은 문서와 키워드가 적은 문서와의 유사도 비교에서 정규화(normalization) 효과를 부여한다. 예를 들어 (그림 7)과 같은 문서 d_i와 d_j가 존재한다면, 두 문서간의 유사도 $S(d_i, d_j)$는 식 (3.2)에 따라 *0.4*가 된다.



(a) 문서 d_i **(b) 문서 d_j**

(그림 7) 문서 예제

(Figure 7) The example of document

식 (3.2)는 계산 시간이 많이 소요되는 코사인 유사도와 같은 기존의 벡터 기반의 유사도 계산보다 간단하여 빠른 계산이 가능하도록 정의하여, 수행속도를 강조하는 본 논문과 부합한다.

3.2.4 클러스터 응집도

클러스터 응집도는 임의의 클러스터에서 소속 문서들이 얼마나 유사한 문서들로 군집되어 있는지를 측정하기 위한 것이다.

$|C_i|$를 클러스터 i에 포함된 문서의 총 개수라고 하면, 클러스터 응집도 $M(C_i)$를 클러스터에 소속된 문서들간의 문서 유사도 평균으로 식 (3.3)과 같이 정의한다. 식 (3.3)을 통해 알 수 있듯이 클러스터 응집도가 높다는 것은 해당 클러스터에 소속된 문서들간의 유사도가 높다는 것을 나타내며 클러스터의 질이 우수함을 의미한다.

$$M(C_i) = \frac{\sum_{d_i \in C_i, d_j \subset C_i - \{d_i\}} S(d_i, d_j)}{|C_i|} \tag{3.3}$$

클러스터 응집도는 문서나 클러스터를 합병할 때, $minC$ 이상을 만족하는지 항상 검사한다. 이것은 클러스터가 합병되어 증대될 때마다 유사문서의 집합인 클러스터로서의 자격을 유지하는지 검사하기 위함이다.

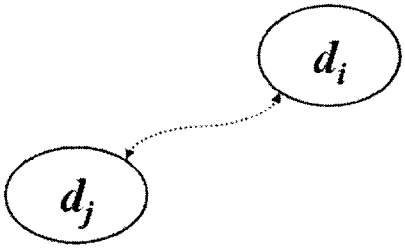
3.3 클러스터 생성 알고리즘

유사한 문서들의 집합인 클러스터를 생성하는 과정은 독립된 문서나 클러스터들을 합병하는 것이다. 제안하는 IDC-TCT는 합병 여부를 결정하는 경우에 따라서 문서와 문서 합병, 문서와 클러스터 합병, 클러스터와 클러스터 합병의 세 가지로 구분하여 각각의 프로시저를 정의한다.

각 프로시저들은 합병 요소들 간에 공평하게 비교될 수 있도록 설계된다. 예를 들어 (그림 10)과 같이 임의의 문서와 클러스터간의 합병 여부를 결정하기 위해서, 링크로 연결된 두 문서간의 유사도를 전체 유사도로 간주하는 것은 불평등하므로 이에 대한 조정이 필요하다.

3.3.1 문서와 문서 합병

(그림 8)과 같이 두 문서간의 합병 여부를 결정하기 위해서 두 문서간의 유사도를 전체 유사도로 간주하더라도 무방하며, 이것은 평등한 비교이다. 식 (3.2)에서 정의한 문서 유사도 식을 통해 두 문서간의 유사도를 구하여 합병 여부를 결정할 수 있다. 즉, 두 문서간의 유사도가 미리 정한 임의의 $minS$ 이상이 되면, 두 문서는 유사하다고 판단하여 합병함으로써 클러스터를 생성한다.



(그림 8) 두 문서간 비교

(Figure 8) The comparison between two documents

문서와 문서 합병을 수행하기 위한 procDAD는 (그림 9)와 같다. procDAD 프로시저에서 두 문서간의 유사도가 *minS* 이상이면 합병을 수행하고, 그렇지 않으면 -1을 리턴하여 합병이 불가함을 알린다.

```
Procedure procDAD(d_i,d_j)
/*      d_i: the compared document
        d_j: the compared opponent document */
begin
  similarity ← S(d_i, d_j);
  if (similarity ≥ minS ) then
    call mergeDTD(d_i,d_j);
  else return -1;
  endif
end
```

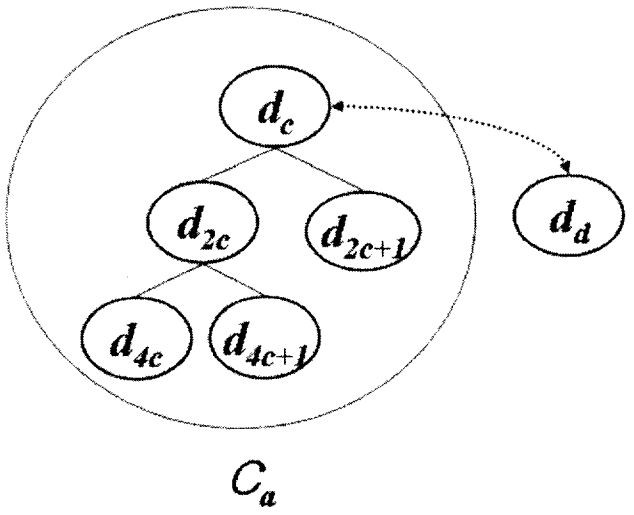
(그림 9) 문서와 문서 합병을 위한 프로시저

(Figure 9) The procedure to merge two documents

3.3.2 문서와 클러스터 합병

문서와 클러스터 합병은 (그림 10)과 같이 임의의 문서와 두 개 이상의 유사 문서로 합병된 클러스터와의 비교이다. 문서와 문서 합병과는 다르게 불평등한 비교이다.

즉, (그림 10)에서 유사 여부를 판단하기 위하여 procDAD처럼 단순히 $S(d_c, d_d)$를 계산하여 $minS$ 이상을 만족하는지 검사하여 합병 여부를 결정하는 것은 불합리하다.



(그림 10) 문서와 클러스터간의 비교

(Figure 10) The comparison between a document and a cluster

이러한 문제점을 고려한 procDAC 프로시저는 (그림 11)과 같다.

Procedure procDAC(d_d, C_a)

/*

d_d: the compared document

C_a: the compared cluster

d_c: the compared document with d_d in the cluster C_a */

begin

$cCoherence \leftarrow M(C_a)$;

$SA_{dc} \leftarrow (S(d_d, d_c) + cCoherence) / 2$;

$mCoherence \leftarrow M(C_a \cup d_d)$;

if (($SA_{dc} \geq minS$) and ($mCoherence \geq minC$)) **then**

call mergeDTC(d_d, C_a);

else return -1;

endif

end

(그림 11) 문서와 클러스터 합병을 위한 프로시저

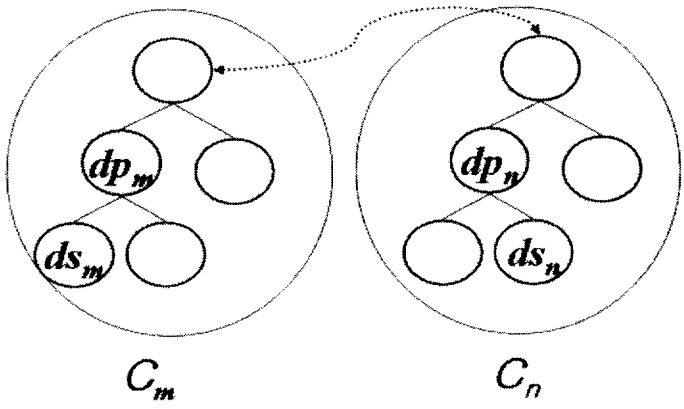
(Figure 11) The procedure to merge a document and a cluster

procDAC 프로시저에서의 주요 고려사항은 문서와 클러스터간의 합병 여부를 결정할 때, 문서 d_d와 클러스터 C_a의 문서 d_c와의 문서 유사도만을 계산하여 합병 여부를 결정하는 것이 아니라, 클러

스터 C_a에 속하는 소속 문서들의 유사도를 고려하는 것이다. 즉, 두 요소간의 유사도는 $S(d_c, d_d)$와 클러스터 C_a에 소속되는 모든 문서들의 평균 유사도이다. 결국 두 요소간의 유사도가 *minS* 이상을 만족하고, 합병 후에 생성되는 클러스터의 응집도가 *minC* 이상이 되어야만 합병이 수행된다.

3.3.3 클러스터와 클러스터 합병

클러스터와 클러스터 합병은 (그림 12)와 같이 두 개 이상의 유사 문서로 구성된 클러스터간의 합병이다. 이것은 동일 개수의 소속 문서를 포함하는 클러스터간의 합병인 경우에는 평등한 합병이 되지만, 소속 문서의 개수가 다르면 불평등한 합병이 된다.



(그림 12) 두 클러스터간 비교

(Figure 12) The comparison between two clusters

클러스터와 클러스터 합병을 수행하기 위한 procCAC는 (그림 13)과 같다.

Procedure procCAC(C_n, C_m)
/*
C_n: the compared cluster
C_m: the compared opponent cluster */
begin
$nCoh \leftarrow M(C_n)$;
$mCoh \leftarrow M(C_m)$;
find a $dp_m - ds_m$ link which approximates to *nCoh*;
find a $dp_n - ds_n$ link which approximates to *mCoh*;
$D_F \leftarrow \{dp_m, ds_m, dp_n, ds_n\}$;
for all elements in D_F **do**
$dSim \leftarrow S(d_i, d_j)$;
endfor
$SC \leftarrow dSim / 6$;
$oCoh \leftarrow M(C_n \cup C_m)$;
if (($SC \geq minMer$) and ($oCoh \geq minC$)) **then**
call mergeCTC(C_n, C_m);
else returen -1;
endif
end

(그림 13) 클러스터와 클러스터 합병을 위한 프로시저

(Figure 13) The procedure to merge two clusters

두 클러스터간의 합병 여부를 결정하기 위해 procCAC 프로시저에서는, (그림 12)에서 두 클러스터의 평균 유사도와 가장 근사하는 각각의 유사 링크 $dp_m - ds_m$, $dp_n - ds_n$을 찾아서 근사 링크 집합(approximate link set) D_F를 구성한다.

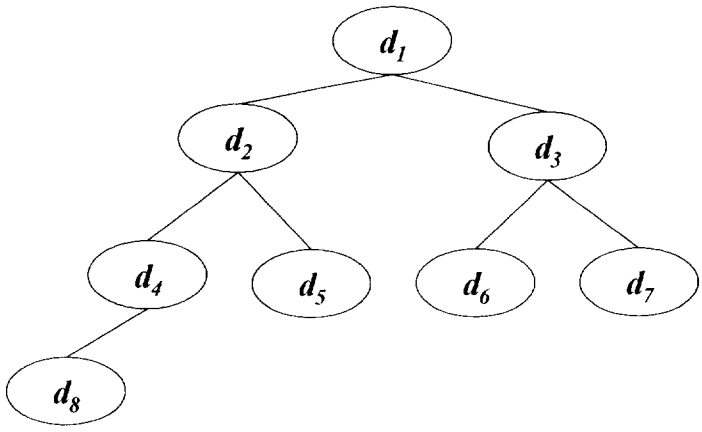
다음으로 두 클러스터간 유사도 SC를 D_F에 포함되는 6개의 유사 링크의 평균으로 구하고, 이것이 $minMer$ 이상을 만족하면서, 두 클러스터를 합병하여 생성되는 클러스터가 $minC$보다 클 때 합병이 수행된다. 여기서 D_F를 구성하여 $minMer$ 이상을 만족하는지 검사하는 이유는 적은 개수의 문서로 구성된 클러스터가 다수의 문서로 구성된 클러스터에 불평등하게 합병되는 것을 막기 위함이다.

3.4 유사도 비교 알고리즘

유사도 비교 알고리즘은 3.3절에서 정의한 클러스터 생성 알고리즘을 이용하여 TCT로 구성된 문서집합에서 이행적 폐쇄 추론에 따라 합병을 수행하기 위한 것이다. 유사도 비교 알고리즘은 합병을 수행하는 단계에 따라서 IC(Initial Comparison) 단계, SC(Sibling Comparison) 단계, DC(Descendant Comparison) 단계, AC(Ancestor Comparison) 단계로 구분된다.

3.4.1 IC 단계

IC 단계는 TCT를 구성하여 초기에 수행된다. (그림 14)와 같이 TCT의 부모-자식 페어(pair)들 간에 유사도를 계산하여 *minS*에 따라 합병 여부를 결정한다.



(그림 14) IC 단계의 예

(Figure 14) The example of the IC step

IC 단계를 수행하기 위한 procIC 프로시저는 (그림 15)와 같다. (그림 15)와 같이 TCT를 BFS(Breath First Search)를 통해 방문하는 노드 d_i와 부모 노드 d_j의 합병 여부를 판단하기 위하여, 클러스터 생성 패턴에 따라 합당한 프로시저를 호출하여 클러스터의 생성 여부를 결정한다. procIC 프로시저에서 parents(d_i)는 d_i의 부모 노드를 구하는 함수이고, comparisonPattern(d_i,d_j)는 d_i와 d_j의

합병 패턴을 구하는 함수이며, clusterInclude(d_j)는 d_j와 유사링크로 연결된 문서 노드들의 집합인 클러스터를 구하는 함수이다.

```
Procedure procIC(TCT)
/*
        TCT: transitive closure tree
        d_i: the visiting current node
        d_j: the d_i's parent node   */
begin
  while each d_i in TCT, i is an order of BFS do
    d_j ← parents(d_i);
    if comparisonPattern(d_i,d_j) is the document and document
    then
      call procDAD(d_i, d_j);
    else
      C_a ← clusterInclude(d_j);
      call procDAC(d_i, C_a);
    endif
  endwhile
end
```

(그림 15) IC 단계를 수행하는 프로시저

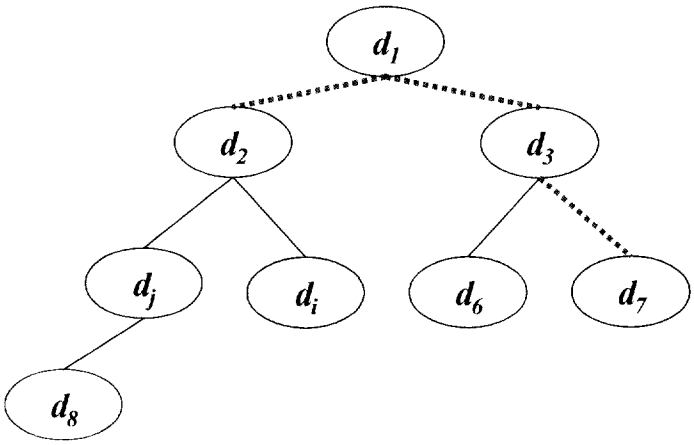
(Figure 15) The procedure to execute the IC step

IC 단계에서 발생하는 합병 패턴은 문서와 문서 합병이거나, 문서와 클러스터 합병이다. 그러므로 비교하는 요소간의 합병 패턴이 문서와 문서 합병이면 3.3.1절에서 정의한 procDAD 프로시저

를 호출한다. 그렇지 않으면 procDAC 프로시저를 호출하여 합병 여부를 결정한다. 결국 BFS를 통해 다음 노드를 방문하면서 모든 노드를 방문할 때까지 반복 수행한다.

3.4.2 SC 단계

SC 단계는 TCT 레벨별로 유사하지 않는 형제 노드(sibling node)간의 합병 여부를 결정하기 위해 수행된다.



(그림 16) SC 단계의 예

(Figure 16) The example of the SC step

(그림 16)에서는 현재 방문하는 노드가 d_2이며, 그 자식 노드(child node) d_i와 d_j는 이전에 수행된 IC 단계에서 유사 하지 않다

고 결정되었으므로, 형제 노드인 d_i와 d_j는 합병 여부를 검사해야 한다. 그러나 형제 노드인 d_6와 d_7은 이미 IC 단계에서 d_3와 d_7이 유사하다고 판단되어 합병 되었고, d_3와 d_6는 유사하지 않다고 판단되었으므로, 이행적 폐쇄 추론에 따라 d_6와 d_7은 유사도를 계산하지 않더라도 유사하지 않다고 판정한다. 결국 d_6와 d_7은 합병 여부를 검사할 필요가 없다. 이제 d_i와 d_j의 합병 여부를 결정하기 위하여 적절한 클러스터 생성 알고리즘을 호출한다. (그림 16)과 같은 경우에는 d_i와 d_j가 독립된 문서이므로 procDAD 프로시저를 호출한다.

SC 단계를 수행하는 procSC 프로시저는 (그림 17)과 같다. (그림 17)의 procSC 프로시저에서 rSibling(d_i) 함수는 문서 노드 d_i의 오른쪽 형제 노드를 구하는 함수이다. SC 단계를 수행하면 각 트리 레벨을 기준으로 클러스터들이 생성되거나, 어떤 클러스터에도 소속되지 않는 기아노드(starvation node)가 남게 된다. 기아노드는 일정 개수가 모이면 기아노드끼리 클러스터링을 수행한다. 기아노드들을 처리하기 위한 과정은 3.6.2절에서 기술한다.

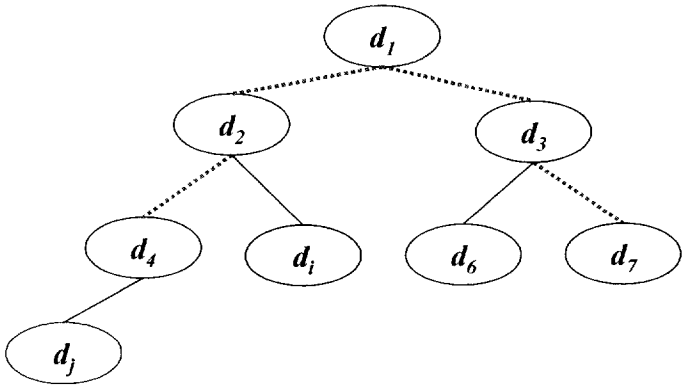
```
Procedure procSC(TCT)
/*      TCT: the transitive closure tree
        d_i: the visiting current node
        d_j: the d_i's right sibling node        */
begin
  while each d_i in TCT, i is an order of BFS do
    d_j ← rSibling(d_i);
    d_k ← parents(d_i);
    if d_i and d_k are not similar then
      if d_j and d_k are not similar then
        case comparisonPattern(d_i,d_j)
          the document and cluster:
            C_a ← clusterInclude(d_j);
            call procDAC(d_i, C_a);
            exit case;
          the cluster and cluster:
            C_a ← clusterInclude(d_i);
            C_b ← clusterInclude(d_j);
            call procCAC( C_a, C_b);
            exit case;
          default:
            call procDAD(d_i, d_j);
            exit case;
        endcase
      endif
    endif
  endwhile
end
```

(그림 17) SC 단계를 수행하는 프로시저

(Figure 17) The procedure to execute the SC step

3.4.3 DC 단계

DC 단계는 후손(하위) 방향으로 유사하지 않는 노드나 클러스터를 찾아 합병 여부를 검사한다. (그림 18)에서 현재 방문하는 노드를 d_i라 하면, 그의 부모 노드 d_2와 유사하지 않지만, d_2와 그의 왼쪽 자식노드 d_4는 유사하며 $\{d_1, d_2, d_3, d_4, d_7\}$으로 클러스터를 형성하고 있다. d_i는 이전에 SC 단계에서 d_4와 이미 비교되어 유사하지 않음을 판정하였으므로, d_4의 하위 자손 방향으로 유사 하지 않는 최초의 노드 d_j와 합병 여부를 결정한다.



(그림 18) DC 단계의 예

(Figure 18) The example of the DC step

DC 단계를 수행하는 procDC 프로시저는 (그림 19)와 같다. (그림 19)의 procDC 프로시저에서 findDescendantNode(d_k) 함수는 문서 노드 d_k와 유사 하지 않은 첫 번째 자손노드를 찾는 함수이다.

```
Procedure procDC(TCT)
/*      TCT: the transitive closure tree
        d_i: the visiting current node
        d_k: the d_i's parent node
        d_j: the first node that is not similar with d_k in descendant
             nodes  */
begin
  while each d_i in TCT, i is an order of BFS do
    d_k ← parents(d_i);
    if d_i and d_k are not similar then
      d_j ← findDescendantNode(d_k);
      case comparisonPattern(d_i,d_j)
        the document and cluster:
          C_a ← clusterInclude(d_j);
          call procDAC(d_i, C_a);
          exit case;
        the cluster and cluster:
          C_a ← clusterInclude(d_i);
          C_b ← clusterInclude(d_j);
          call procCAC(C_a, C_b);
          exit case;
        default:
          call procDAD(d_i, d_j);
          exit case;
      endcase
    endif
  endwhile
end
```

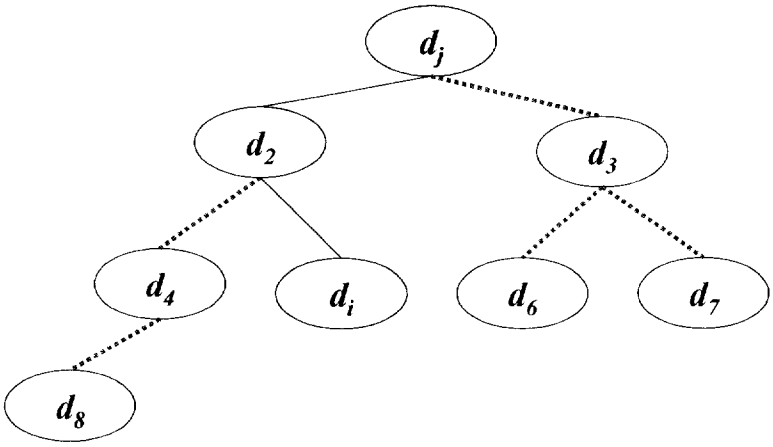
(그림 19) DC 단계를 수행하는 프로시저

(Figure 19) The procedure to execute the DC step

DC 단계의 수행을 통해 BFS로 방문하는 노드들에 대하여, 이전의 수행에서 누락된 하위 방향으로의 합병을 수행한다.

3.4.4 AC 단계

(그림 20)과 같이 AC 단계는 TCT에서 임의의 노드에서 조상(상위) 방향으로 유사하지 않는 노드나 클러스터를 찾아 합병 여부를 검사한다.



(그림 20) AC 단계의 예

(Figure 20) The example of the AC step

DC 단계에서의 수행 과정과 유사하게, AC 단계에서는 (그림 20)의 예와 같이 현재 방문하는 노드 d_i에서 조상(상위) 노드 방향으로 부모노드 d_2와 유사하지 않는 최초의 노드 d_j와 유사 여부를 판별한다. 이때 d_i는 문서노드이지만, d_j는 $\{d_j, d_3, d_6, d_7\}$로 클러스터의 소속 문서이다. 그러므로 클러스터 생성 알고리즘에서 문서와 클러스터의 합병을 결정하는 procDAC 프로시저를 호출하여 수행한다.

AC 단계를 수행하는 procAC 프로시저는 (그림 21)과 같다. (그림 21)에서 procAC 프로시저는 procDC 프로시저와 유사하며, findAncestorNode(d_k)는 문서 노드 d_k를 기준으로 유사하지 않는 최초의 조상 노드를 찾는 함수이다.

```
Procedure procAC(TCT)
/*      TCT: the transitive closure tree
        d_i: the visiting current node
        d_k: the d_i's parent node
        d_j: the first node that is not similar with d_k in ancestor
             nodes */
begin
  while each d_i in TCT, i is an order of BFS do
      d_k ← parents(d_i);
      if d_i and d_k are not similar then
        d_j ← findAncestorNode(d_k);
          case comparisonPattern(d_i,d_j)
            the document and cluster:
              C_a ← clusterInclude(d_j);
              call procDAC(d_i, C_a);
              exit case;
            the cluster and cluster:
              C_a ← clusterInclude(d_i);
              C_b ← clusterInclude(d_j);
              call procCAC( C_a, C_b);
              exit case;
            default:
              call procDAD(d_i, d_j);
              exit case;
          endcase
      endif
  endwhile
end
```

(그림 21) AC 단계를 수행하는 프로시저

(Figure 21) The procedure to execute the AC step

3.5 FDC-TCT 프로시저

FDC-TCT(Fast Document Clustering based on the TCT)는 3.4절에서 제시한 유사도 비교 알고리즘을 이용하여 초기 클러스터링을 수행하기 위한 것이다. FDC-TCT를 수행하는 procFDC-TCT는 (그림 22)와 같다. procFDC-TCT 프로시저는 먼저 초기 클러스터링을 수행하기 위하여 procIC 프로시저를 수행하고 나서 procSC 프로시저를 수행한다. 그리고 루트 노드(root node)를 중심으로 왼쪽 서브 트리(left sub-tree)와 오른쪽 서브트리(right sub-tree)를 구성하여, 각각에 대해 procDC 프로시저를 수행한다. 끝으로 전체 노드에 대해서 procAC 프로시저를 수행하면 최종 클러스터가 결정된다.

```
Procedure procFDC-TCT(TCT)
begin
  ITCT ← call procIC(TCT);
  STCT ← call procSC(ITCT);
  L-TCT ← a left sub-tree of the root node in STCT;
  R-TCT ← a right sub-tree of the root node in STCT;
  DL-TCT ← call procDC(L-TCT);
  DR-TCT ← call procDC(R-TCT);
  D-TCT ←  merge DL-TCT and DR-TCT;
  T-TCT ← call procAC(D-TCT);
end
```

(그림 22) FDC-TCT를 수행하는 프로시저

(Figure 22) The procedure to execute the FDC-TCT

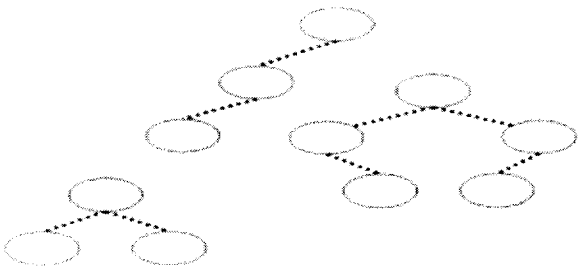
웹 정보는 고정된 것이 아니라 끝임 없이 삽입, 갱신, 삭제가 발생한다. 이러한 변화가 발생 할 때마다 클러스터링을 다시 수행하는 것은 비효율적이다. 다음 절에서는 procFDC-TCT 프로시저를 수행하여 획득한 클러스터링 정보를 체계적으로 저장하여 활용하는 점진적 문서 클러스터링에 관하여 기술한다.

3.6 점진적 문서 클러스터링

점진적 문서 클러스터링을 제공하지 않는 문서 클러스터링 알고리즘은 새로운 문서를 삽입하거나 기존의 클러스터에서 문서를 삭제할 때마다, 새롭게 클러스터링을 수행해야 하는 단점이 있다. 이러한 문제점을 해결하기 위한 대안이 점진적 문서 클러스터링이며 최근에 많은 연구가 진행되고 있다[10, 62].

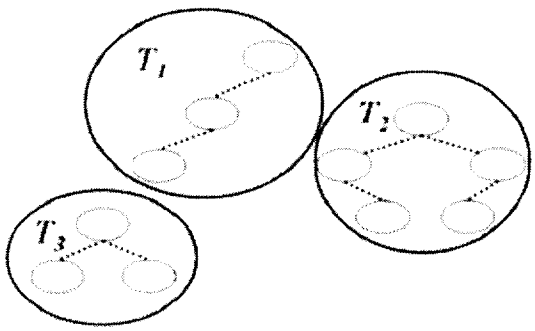
본 논문에서 제안하는 IDC-TCT에서도 점진적 문서 클러스터링이 가능하도록 설계함으로써, 문서의 삽입과 삭제에 대하여 중복적인 문서간 유사도 계산을 하지 않도록 설계한다. 이를 위해 IDC-TCT는 FDC-TCT를 수행하여 생성된 각 클러스터로부터 공통 키워드 테이블(common keyword table) T_i를 구성한다.

(그림 23)과 같이 클러스터링 된 결과가 있으면, 전체 트리에서 포레스트인 각 클러스터들은 하나의 문서로 간주하여 (그림 24)와 같이 공통 테이블 T_i로 구성한다.



(그림 23) 포레스트로 구성된 클러스터 집합

(Figure 23) A set of cluster to be constructed by forest



(그림 24) 각 포레스트별로 공통 키워드 테이블 생성

(Figure 24) A creation of public keyword tables for each forest

클러스터의 총 개수를 m이라 하고, r_i를 i번째 키워드 출현 빈도수라 하고, k_i를 i번째 키워드의 명칭이라 하면, 공통 키워드 테이블 T_i는 정의 3.2와 같다.

정의 3.2: 공통 키워드 테이블 T_i

$$T_i = \{R_i, K_i\}, \text{ for } i = 1, 2, \cdots, m$$
$$R_i = \{r_1, r_2, \cdots, r_n\}$$
$$K_i = \{k_1, k_2, \cdots, k_n\}$$

T_i를 생성하는 procCreateCKT 프로시저는 (그림 25)와 같다. (그림 25)의 procCreateCKT 프로시저에서 integrateDocument(C_i)는 클러스터 C_i에서 포함된 문서들을 하나로 통합하여 공통 문서를 생성하는 함수이다. sortAcending(CD_i,K_i)는 공통 문서 CD_i에서 키워드 집합 K_i를 오름차순으로 정렬하는 함수이다. 끝으로 findFair(count,K_i)는 각 키워드별로 동일한 개수인 count를 계산하여 (R_i,K_i) 페어를 생성하는 함수이다. 결국 procFDC-TCT 프로시저 수행 결과인 TCT로 부터 각 클러스터별로 공통 문서로 통합하고, 키워드 명으로 정렬하여 (R_i,K_i) 페어로 공통 키워드 테이블을 생성한다.

```
Procedure procCreateCKT(RTCT)
/*      RTCT: the execution result of procFDC-TCT
        C_i: the ith cluster in TCT
        m: the total number of clusters in TCT
        CKT_i: the ith public keyword table  */
begin
 for each C_i, i = 1, 2, ..., m do
     CD_i ← integrateDocument(C_i);
     SD_i ← sortAcending(CD_i, K_i);
  endfor
  for each SD_i, i = 1, 2, ..., m do
    while all K_i do
       CKT_i ← findFair(count, K_i);
    endwhile
  endfor
end
```

(그림 25) 공통 키워드 테이블을 생성하는 프로시저

(Figure 25) The procedure to create the common keyword table

3.6.1 문서 삽입 처리

본 논문에서 제안하는 IDC-TCT에서 문서 삽입의 핵심은 미리 계산된 클러스터에 소속되는 모든 문서를 대표하는 공통 키워드 테이블을 생성하고, 삽입되는 새로운 문서와의 유사도를 계산하여, 클러스터에 포함여부를 결정하는 것이다.

새로운 삽입 문서 d_n에 대하여 점진적 문서 클러스터링을 수행하는 procInsertDocument는 (그림 26)과 같다. 새로운 삽입 문서 d_n이 임의의 클러스터에 포함되기 위해서는, 자신과 공통 키워드 테이블과의 유사도가 *minCarSim* 이상이어야 한다. 그러면 새로운 삽입 문서 d_n과 공통 테이블과의 유사도가 *minCarSim* 이상을 만족하는 클러스터 중에서 유사도가 가장 큰 클러스터에 포함된다.

d_n이 모든 공통 키워드 테이블과의 유사도가 *minCarSim* 이하일 때는 어느 클러스터에도 소속되지 않는 문서로서, 기아 문서 테이블(starvation document table)에 저장된다. 기아 문서 테이블은 일정 개수 이상이 수집되면 기아 문서끼리 유사도를 비교하여 클러스터링을 수행한다.

procInsertDocument 프로시저를 수행함으로써, 삽입 문서수를 z라 하고, 공통 테이블 수를 c라 하고, 삽입 문서와 공통 테이블과의 유사도를 계산하기 위해 소요되는 시간을 t라 하면, 새로운 문서를 포함하여 클러스터링을 수행하기 위한 시간은 단지 $O(z \cdot c \cdot t)$가 필요하다.

```
Procedure procInsertDocument(d_n, CKT)
/*      CKT: a set of common keyword table
        d_n: an inserted document
        m: the total number of CKT  */
begin
  for each CKT_i, i = 1, 2, ..., m do
    s_i ← S(d_n, CKT_i);
  endfor
    maxSim ← 0;
    position ← -1;
  for each CKT_i, i = 1, 2, ..., m do
    if s_i ≥ minCarSim then
      isStarvation ← -1;
      if s_i ≥ maxSim then
        position ← i;
      endif
    else isStarvation ← 1;
    endif
  endfor
  if isStarvation = -1 then
    return position;
  else return -1;
  endif
end
```

(그림 26) 문서 삽입을 수행하는 프로시저

(Figure 26) The procedure to execute a document insertion

3.6.2 문서 삭제 처리

임의의 클러스터로부터 문서를 삭제할 때, 클러스터 응집도의 변화가 발생한다. 따라서 문서 삭제에 따라 클러스터가 *minC*를 유지하는지 검사해야 한다.

문서 삭제를 수행하는 procDeleteDocument는 (그림 27)과 같다.

```
Procedure procDeleteDocument(d_r, RC)
/*      d_r: a removed document
        RC: A cluster which includes d_r
        m: a total number of cluster  */
begin
   UC ← {RC } - {d_r};
   UCCoherence ←  M(UC);
   if UCCoherence ≥ minC then
      RC  ← UC;
   else
      store UC to the starvation document table;
      RC ← -1;
   endif
      return RC;
end
```

(그림 27) 문서 삭제를 수행하는 프로시저

(Figure 27) The procedure to execute a document deletion

procDeleteDocument 프로시저와 같이 삭제될 문서 d_r을 제외한 클러스터 UC의 클러스터 응집도를 구하여 $minC$보다 크면 d_r을 제거하고, $minC$보다 작으면 더 이상 유사 문서의 집합인 클러스터의 기능을 상실하였으므로 삭제되어야 한다. 그리고 소속된 문서들은 모두 기아 문서 테이블에 저장하여 차후 다른 문서들과 클러스터링을 수행한다.

기아 문서 테이블에 소속된 문서들간의 유사도 처리는 기아 문서 처리율인 STR을 선정하여, 전체 문서수에서 기아 문서의 비율이 STR을 초과 하는 경우에 수행한다.

3.7 IDC-TCT 프로시저

점진적 문서 클러스터링을 수행하는 procIDC-TCT 프로시저는 (그림 28)과 같다. (그림 28)처럼 키워드로 추출된 문서집합을 연관 순서(association order)에 따라 문서들을 정렬하고, 3.1절에서 정의한 TCT로 구성하여, procFDC-TCT 프로시저를 통해 클러스터링을 수행한다. 생성된 클러스터 정보를 procCreateCKT 프로시저를 수행하여 공통 키워드 테이블을 구성한 후에, 새로운 문서의 삽입이 발생하면 procInsertDocument 프로시저를 호출하여 처리하고, 삭제되는 문서가 발생하면 procDeleteDocument 프로시저를 호출하여 처리한다.

```
Procedure procIDC-TCT(TCT)
 /*   TCT: the transitive closure tree
      CKT: a set of common keyword table
      z_i: the ith common keyword table in CKT
      RC: a cluster which includes d_r
      d_n: an inserted document
      d_r: a removed document      */
begin
  rTCT ← call procFDC-TCT(TCT);
  CKT ← call procCreateCKT(rTCT);
  if d_n exists in CKT then
    position ← call procInsertDocument(d_n,CKT);
    if position ≠ -1 then
      store d_n to the starvation document table;
    else insert d_n to z_position in CKT;
    endif
  endif
  if d_r exists in CKT then
    RC ← clusterInclude(d_r);
    RC ← call procDeleteDocument(d_r,RC);
      if RC = -1 then
       delete RC from CKT;
      else update RC in CKT;
      endif
  endif
end
```

(그림 28) IDC-TCT를 수행하는 프로시저

(Figure 28) The procedure to execute the IDC-TCT

본 논문에서 제안한 IDC-TCT는 범주 선정과 같은 전문가의 개입을 필요로 하지 않으므로 방대한 웹 문서 결과 분류나 문서 데이터 분류에 적절한 기법이다. 또한 IDC-TCT는 정확률이 저조한 반복적 클러스터링 알고리즘보다 정확률을 향상시키고, 수행속도가 느린 계층적 클러스터링 알고리즘보다 수행 시간을 단축하는 기법이다. 그러므로 제안하는 기법의 적용은 일정 수준 이상의 정확률을 유지하면서 빠른 수행속도를 요구하는 문서 분류 분야에 적절하다.

4. 성능 평가

본 논문에서 제안한 IDC-TCT의 성능을 평가하기 위하여 국문과 영문으로 구성된 웹 문서 집합을 대상으로 실험한다.

먼저 *minC*의 변화에 따라 생성되는 클러스터를 분석한다. 그리고 정확률, 재현율, F-Measure, 초기 클러스터링 수행시간 및 점진적 문서 클러스터링의 수행속도를 측정하여 분석한다. 점진적 문서 클러스터링의 수행속도는 IDC-TCT의 점진적 문서 클러스터링의 성능을 분석하기 위하여 새로운 문서 삽입 개수 증가에 따른 수행속도를 측정하는 것이다.

기존의 알고리즘과 정확률, 재현율 및 F-Measure를 비교 및 평가하기 위하여, 수행속도가 우수한 반복적 클러스터링 알고리즘으로는 k-means 알고리즘, 분류의 질이 우수한 계층적 클러스터링 알고리즘으로는 SL, CL, AL 알고리즘과 비교한다.

4.1 실험 환경

IDC-TCT의 성능을 평가하기 위한 프로그램, 수행 환경, 문서 데이터에 대한 실험 환경을 <표 3>에 요약한다.

<표 3> 실험 환경

<Table 3> The simulation environment

프로그램	키워드 추출: HAM 이용 IDC-TCT: 자작 프로그램(JDK 1.3)
수행 환경	PC: P-IV 2.0GHz, 512M RAM
문서데이터	네이버 문서 집합, Reuters-21578 문서 집합

<표 3>과 같이 키워드 추출은 HAM을 이용한다. 하나의 어근으로 통일하는 스테밍 알고리즘과 불용어 처리를 위한 데이터 사전은 HAM에서 기본적으로 제공하는 것을 이용한다. 그리고 단어 목록 중에서 IDF 수치가 3 미만인 단어들을 제거하여 키워드 집합을 구성한다. 이것은 불용어 처리에서 제외된 단어들을 키워드 집합에서 필터링하기 위한 것이다.

실험에 사용되는 문서는 두 종류의 문서 집합으로, 첫째는 국문으로 된 포털 사이트의 문서 집합으로서 네이버의 디렉토리 검색에서 '소프트웨어' 범주에 소속되는 문서 집합이다. 이 문서 집합은 20개의 범주로 구성되며 각 범주명과 소속 문서수는 <표 4>와 같다.

<표 4> 네이버 문서 집합

<Table 4> Naver document set

일련번호	범주명	소속 문서수
1	FTP	100
2	P2P	60
3	PDA	80
4	WWW	140
5	가정용	80
6	공개자료실	110
7	교육용	80
8	기관, 단체	60
9	단위환산	40
10	디바이스 드라이버	90
11	라이센스	80
12	멀티미디어툴	160
13	문서 뷰어	90
14	문서 편집기	40
15	사무용	150
16	소프트웨어업체	180
17	운영체제	120
18	유틸리티	140
19	전자출판	120
20	제품소개, 리뷰	80
합계		2000

두 번째 실험에 사용된 문서는 영문으로 구성된 Reuters-21578 문서 집합이다[17]. 이 문서 집합은 Reuters newswire에 게재된

기사의 모음으로써, Reuter와 Carnegie 그룹에서 수작업으로 범주를 인텍싱하여 Reuters-22173으로 공개하였고, 이를 David Lewis 등이 중복되는 문서를 정리하여 구성한 것이다. 대개 텍스트 마이닝(text mining) 분야에서 분류의 질을 판단하기 위하여 자주 사용되는 문서 집합이다.

Reuters-21578은 <표 5>와 같이 5개의 범주와 각각의 세부 범주들로 구성되어 있다.

<표 5> Reuters-21578의 범주 구성

<Table 5> The construction of category for the Reuters-21578

범주명	세부 범주수
EXCHANGE	39
ORGS	56
PEOPLE	267
PLACES	175
TOPICS	135

5개의 범주들 중에서 본 논문에서는 경제 분야에 대한 기사로 구성된 TOPICS 범주에서 10개의 세부 범주에 소속되는 문서를 실험에 사용한다.

<표 6>과 같이 각 세부 범주명에 따라 소속 문서수를 구성하여 실험에 사용한다. 원래 Reuter-21578 문서 집합은 각 범주별로 학습 데이터(training data)와 테스트 데이터(test data)를 포함하고 있다. 본 실험에서 각 범주별로 사용된 문서들은 비교적 질적으로 우수한 학습 데이터를 우선으로 선정하여 사용하며 부족한 문서수에 대해서만 테스트 데이터를 보충하여 사용한다.

<표 6> Topic 범주 문서 집합

<Table 6> The document set for Topic category

일련번호	범주명	소속 문서수
1	Earn	290
2	Acq	280
3	Money-fx	265
4	Grain	240
5	Crude	225
6	Trade	190
7	Interest	160
8	Wheat	130
9	Ship	120
10	Corn	100
합계		2000

4.2 문서 라벨링

정확률과 재현율을 효율적으로 계산하기 위해 <표 7>과 같이 문서 라벨링(document labeling)을 구성한다. 즉, 문서 라벨링을 통해 두 문서가 동일 범주에 소속되는지를 신속하게 판단할 수 있을 뿐만 아니라, 문서 클러스터링 후에 원래의 소속 범주를 판단하기 위한 수단으로 사용한다.

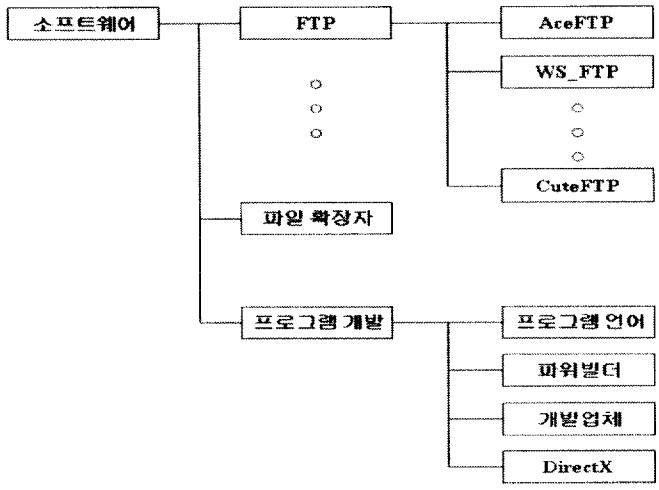
<표 7> 문서 라벨링 구조

<Table 7> The structure for a document labelling

the lowest level number	category_id	document_id

<표 7>에서 'the lowest level number'는 임의의 문서가 최상위 레벨이 1인 범주의 레벨에서 최하위 범주를 의미한다. 즉, 'the lowest label number'는 1에서 최하위 카테고리 수까지의 일련번호이다. 그리고 임의의 문서가 해당 레벨에서 몇 번째 범주에 속하는지 category_id로 표기하고, 해당 범주에서 몇 번째 문서인지 document_id로 표기한다. category_id는 1에서 최대 범주 수까지의 일련번호이고, document_id는 1에서 최대 문서수까지의 일련번호이다.

예를 들어 (그림 29)와 같은 네이버에서의 '소프트웨어' 디렉토리 검색에 대한 범주 집합에서, 'FTP' 범주에 속하는 문서들은 문서 라벨링이 '2 1 document_id'가 되고, 'WS_FTP' 범주에 속하는 문서들은 문서 라벨링이 '3 2 document_id'가 된다.



(그림 29) 네이버에서 '소프트웨어' 범주 트리

(Figure 29) The category tree for the software in the Naver

이러한 문서 라벨링을 통해 클러스터링 수행 후 임의의 두 문서가 동일 카테고리에 속하는지 여부를 간단한 마스크 연산(mask operation)으로 신속하게 판별할 수 있다. 예를 들어, 문서 라벨링이 구성되는 항목이 각각 32 비트로 구성되고, 문서 A와 B의 문서 라벨링을 각각 L_A, L_B라고 하면, (그림 30)과 같은 간단한 마스크 연산으로 동일 범주에 소속되는지 판단한다.

```
if (( L_A and [1111 1111 0000]) = ( L_B and [1111 1111 0000] )) then
        "identical category";
else "different category";
```

(그림 30) 동일 범주를 판단하는 마스크 연산

(Figure 30) The mask operator to decide a identical category

4.3 성능 평가 요소

본 실험에서의 정확률은 클러스터링 수행 후 IDC-TCT가 분류한 문서들에 대하여, 원래의 문서 집합에 소속된 문서와 일치하는 개수에 대한 비율이다. 한편 재현율은 원래의 문서 집합에 소속된 문서들에 대하여, 분류된 문서가 동일 범주에 소속되는 문서와 일치하는 개수에 대한 비율이다.

우선 정확률과 재현율을 구하기 위하여 IDC-TCT가 분류한 클러스터별로 대표 클러스터를 구한다. 대표 클러스터는 분류기가 분류한 클러스터별로 가장 많은 소속 문서를 포함하는 클러스터이다. 대표 클러스터 RC_j를 구하기 위해 L_{d_i}를 문서 d_i의 라벨링이라 하고, n은 문서의 총수라고 하면, 분류기가 분류한 클러스터별로 문서 d_i가 어느 카테고리에 속하는지를 구하는 $L(d_i)$는 (식 4.1)과 같다.

$$LC(d_i) = L_{d_i}\ AND\ [1111\,1111\ 11111111\ 0000000000000000],$$
$$\text{for } i = 1, 2, \cdots, n \tag{4.1}$$

그리고 m을 IDC-TCT에 의해 분류된 클러스터의 총수라고 하면, 분류기가 분류한 클러스터별로 소속 문서가 제일 많은 대표 클러스터 RC_j를 구하는 수식은 식 (4.2)와 같다.

$$RC_j = \max\{|LC(d_i)|\,|i \neq k,\ LC(d_i) = LC(d_k)\},$$
$$\text{for } i, k = 1, 2, \cdots, n,\ j = 1, 2, \cdots, m \tag{4.2}$$

그러면 정확율은 IDC-TCT가 분류한 클러스터의 소속 문서와 네이버 범주의 소속 문서와 일치하는 개수에 대하여 분류된 문서 수와의 비율이므로, CD_j를 네이버 문서 집합에서 j번째 범주에 포함된 문서 집합이라 하면, 정확률 P는 식 (4.3)과 같다.

$$P = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{|\,CD_j \cap RC_j|}{|RC_j|},\ j = 1, 2, \cdots, m \tag{4.3}$$

한편 재현율 R은 식 (4.4)를 통해 구한다.

$$R = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{|CD_j \cap RC_j|}{|\overline{CD_j}|}, \quad j = 1, 2, \cdots, m \tag{4.4}$$

끝으로 식 (4.5)을 통해 F-Measure F를 측정한다.

$$F = \frac{2 \times P \times R}{P + R} \tag{4.3}$$

4.4 실험 결과 및 분석

본 절에서는 IDC-TCT를 수행한 결과를 분석하고 평가한다. 실험 1은 국문 키워드로 구성된 네이버 문서 집합을, 실험 2는 영문 키워드로 구성된 Reuters-21578 문서 집합을 대상으로 실시한다.

4.4.1 네이버 문서 집합

실험 1에 사용된 네이버 문서집합은 각 사이트를 설명하는 문구와 해당 사이트에서 사용되는 단어를 추가하여 구성한다. 그러므로 소속 범주간에 다소 약한 결합으로 이루어져 있다. 따라서 실험 1은 Reuter-21578 문서 집합을 수행하는 실험 2보다 전체적인

성능이 다소 저하되는 특성이 있다.

클러스터 응집도는 소속 문서들간의 유사 정도를 나타내므로, 클러스터 응집도를 관찰하면 클러스터링 알고리즘의 특성을 파악할 수 있다.

1) *minC*의 변화에 따른 클러스터 개수 변화

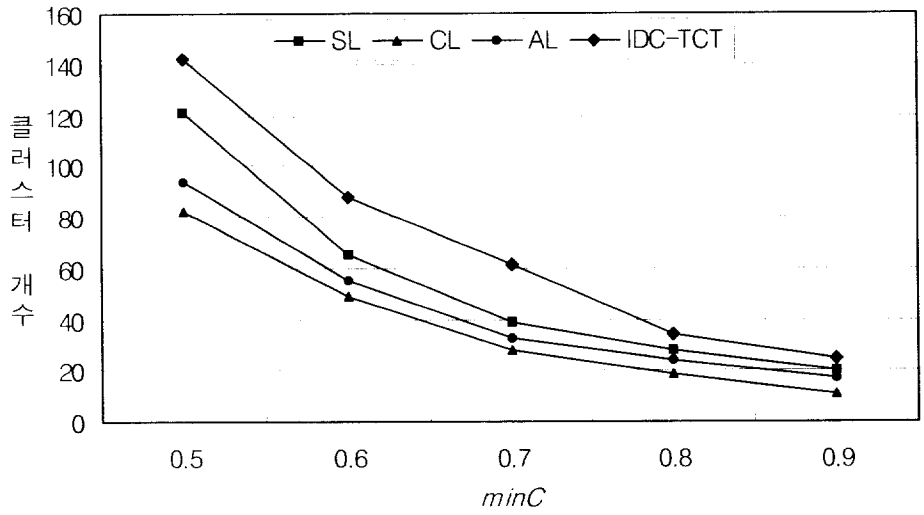
우선 네이버 문서집합에 대하여 IDC-TCT를 수행하면서, *minC*를 0.5, 0.6, 0.7, 0.8, 0.9로 설정하여 클러스터 개수의 변화를 SL 알고리즘, CL 알고리즘. AL 알고리즘과 비교하여 <표 8>과 (그림 31)에 나타낸다. 이후에는 SL 알고리즘, CL 알고리즘, AL 알고리즘을 간단하게 SL, CL, AL로 표기한다.

<표 8> *minC*의 변화에 따른 클러스터 개수 비교

<Table 8> The comparison of cluster number according to a minimum coherence

(단위: 개)

minC / 알고리즘	0.5	0.6	0.7	0.8	0.9
SL	121	65	39	28	20
CL	82	49	28	19	11
AL	94	55	33	24	17
IDC-TCT	142	88	61	34	25



(그림 31) *minC*의 변화에 따른 클러스터 개수 비교

(Figure 31) The comparison of cluster number according to a minimum coherence

(그림 31)과 같이 IDC-TCT나 계층적 클러스터링 알고리즘들은 *minC*가 증가함에 따라 *minS*가 높은 문서들만을 합병하므로 클러스터의 개수가 감소한다. 이것으로 *minC*의 증가에 따라 비교 기법들이 클러스터링을 제대로 수행하는 것을 알 수 있다.

합병되는 클러스터의 개수가 작은 것은 그만큼 합병의 기준이 엄격하고 클러스터의 질이 높다는 것을 나타낸다. 그러므로 CL, AL, SL, IDC-TCT 순으로 엄격한 클러스터링을 수행하지만, 수행시간은 많이 소요된다. (그림 31)을 통해 알 수 있듯이 *minC*가 낮

을 때는 IDC-TCT와 AL에 따라 합병되는 클러스터의 개수의 차가 크지만, *minC*가 높을수록 그 차가 줄어든다.

*minC*가 0.6에서 부터는 클러스터 개수의 감소가 완만한 것을 알 수 있으므로 정확률 및 재현율을 산출하기 위해 *minC*를 0.6으로 설정하는 것이 적절하다. 이것은 *minC*가 너무 높으면 엄격한 클러스터링을 수행하여 클러스터간 합병시 유사한 클러스터간의 합병이더라도 해체될 가능성이 크기 때문이다.

2) *minC*의 변화에 따른 클러스터당 평균 문서수

*minC*의 증가에 따라 클러스터당 평균 문서수가 감소한다. 이는 강한 응집을 이룰 수 있는 자격이 있는 문서가 줄어들기 때문이다. <표 9>와 (그림 32)를 보면 SL, CL, AL이 IDC-TCT보다 대체적으로 클러스터 개수와 소속 문서수가 작아서 비교적 엄격한 클러스터링을 수행함을 알 수 있다.

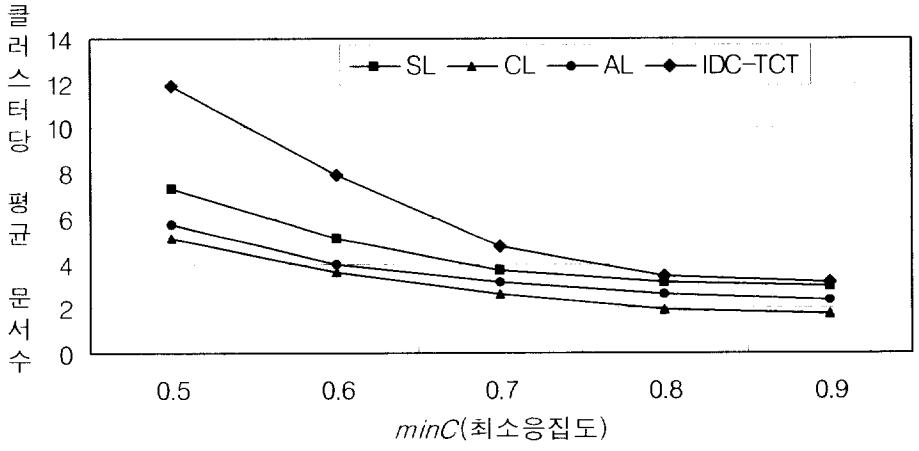
*minC*가 0.5일 때는 IDC-TCT와 SL은 CL과 AL에 비하여 클러스터당 평균 문서수의 편차가 크지만, *minC*가 0.8보다 커지면 거의 차이가 나지 않는 것을 알 수 있다. 그러므로 클러스터링을 수행하면서 *minC*가 0.8 이상을 설정하여 엄격한 클러스터링을 수행할 때에는 수행속도가 많이 걸리는 AL과 CL을 적용하는 것보다 IDC-TCT나 SL을 적용하는 것이 타당하다.

<표 9> *minC*의 변화에 따른 클러스터당 평균 문서수 비교

<Table 9> The comparison of an average number of document per a cluster according to a minimum coherence

(단위: 개)

minC / 알고리즘	0.5	0.6	0.7	0.8	0.9
SL	7.35	5.15	3.74	3.18	2.98
CL	5.13	3.58	2.68	1.97	1.79
AL	5.74	3.97	3.15	2.63	2.37
IDC-TCT	11.85	7.92	4.74	3.41	3.19



(그림 32) *minC*의 변화에 따른 클러스터당 평균 문서수 비교

(Figure 32) The comparison of an average number of document per a cluster according to a minimum coherence

3) *minC*의 변화에 따른 평균 클러스터 응집도

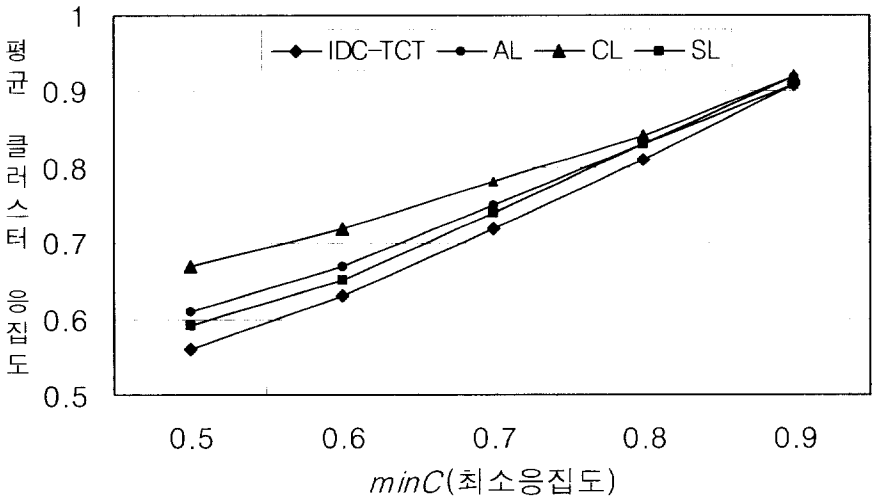
*minC*의 증가에 따라 평균 클러스터 응집도를 <표 10>과 (그림 33)에 나타낸다. (그림 33)를 보면 IDC-TCT보다 SL, CL, AL이 비교적 높은 평균 클러스터 응집도를 나타낸다. 이것은 계층적 클러스터링 알고리즘이 IDC-TCT보다 생성되는 클러스터의 질이 좋다고 할 수 있다. 그러나 IDC-TCT는 *minC*가 커질수록 다른 알고리즘들과의 성능차가 줄어든다.

<표 10> *minC*의 변화에 따른 평균 클러스터 응집도 비교

<Table 10> The comparison of an average for cluster coherence according to a minimum coherence

(단위: 비율)

알고리즘 \ *minC*	0.5	0.6	0.7	0.8	0.9
SL	0.59	0.65	0.74	0.83	0.91
CL	0.67	0.72	0.78	0.84	0.92
AL	0.61	0.67	0.75	0.83	0.92
IDC-TCT	0.56	0.63	0.72	0.81	0.91



(그림 33) *minC*의 변화에 따른 평균 클러스터 응집도 비교

(Figure 33) The comparison of an average cluster coherence according to a minimum coherence

4) 정확률, 재현율 및 F-Measuer

IDC-TCT와 기존의 문서 클러스터링 기법에 대해 정확률, 재현율 및 F-Measure를 측정한다. 계층적 클러스터링 알고리즘인 SL, CL, AL뿐만 아니라 반복적 클러스터링 알고리즘인 k-means 알고리즘과 비교 및 평가한다. (그림 31)에서 클러스터의 개수가 완만하게 감소되는 지점으로 *minC*를 0.6으로 설정하고, 그 외에 *minS*를 *0.6*, *k*를 *20*, *minMer*를 *0.7*로 설정하여 클러스터링을 수행한다.

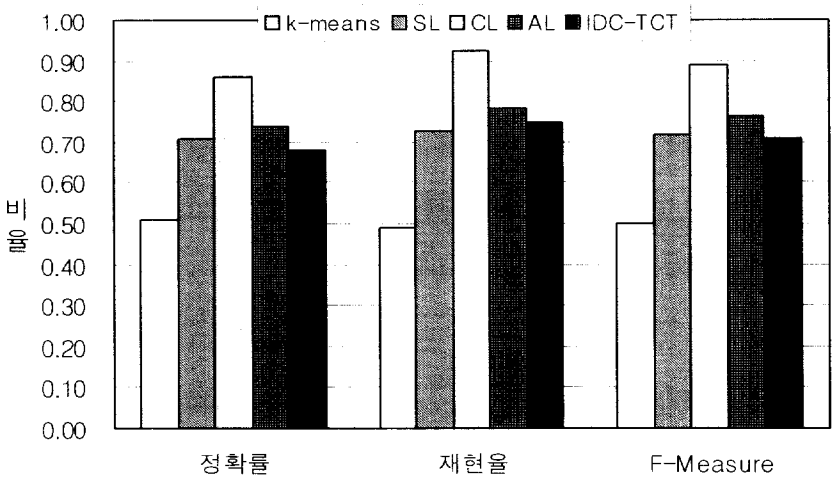
<표 11>, (그림 34)와 같이 과도한 수행시간이 요구되는 CL이 정확률 및 재현율이 가장 우수한 반면에 수행시간이 가장 빠른 k-means 알고리즘은 정확률 및 재현율이 가장 저조하다. 여기서 k-means 알고리즘은 초기 센트로이드 선정을 적절하게 조정하지 않으면 분류의 정확성을 요구하는 분야에는 적용이 어렵다는 것을 알 수 있다.

제안하는 IDC-TCT는 정확률 및 재현율에서 k-means 알고리즘보다 아주 우수하고 SL과는 비슷한 성능을 보인다. 그리고 SL보다 정확률은 저조하지만 재현율에서는 오히려 우수한 특성이 있다. 이것은 이행적 폐쇄 이론에 따라 연계된 유사 문서간의 링크가 강한 결합을 이루는 특성 때문이다.

<표 11> 정확률, 재현율 및 F-Measure 비교

<Table 11> The comparison of a precision, a recall, and a F-Measure

알고리즘 / 평가요소	k-means 알고리즘	SL	CL	AL	IDC-TCT
정확률	0.51	0.71	0.86	0.74	0.68
재현율	0.49	0.73	0.92	0.78	0.75
F-Measure	0.50	0.72	0.89	0.76	0.71



(그림 34) 정확률, 재현율 및 F-Measure 비교

(Figure 34) The comparison of a precision, a recall, and a F-Measure

5) 초기 클러스터링 수행속도 비교

문서 클러스터링에서 소요되는 대부분의 시간은 문서간의 유사도를 비교하는 횟수이다. IDC-TCT의 수행시간은 분류하려는 문서의 총수를 n이라 하고, 생성되는 클러스터 수를 c라고 하면, 대부분의 수행시간이 걸리는 IC 단계에서 $nlogn$번의 유사도 비교 횟수가 필요하다. 그리고 SC, DC, AC단계에서는 생성되는 클러스터 개수만큼 추가적인 비교가 필요하므로 $cnlogn$번의 비교 횟수가 필요하다. 여기서 c의 범위는 $1 \leq c \leq n/2$이 되므로 전체 시간

복잡도는 $O(n^2 \log n)$으로 유도된다. 그러나 검색자가 일일이 분류하기에는 난해 할 정도로 문서수가 충분히 많다고 가정하면, $n \gg c$ 이므로 거의 $nlogn$ 번의 유사도 비교횟수가 필요하다.

IDC-TCT의 수행속도를 평가하기 위해 문서의 개수를 1000, 2000, 3000, 4000, 5000개로 구성하여 k-means 알고리즘, SL, CL, AL과 비교한다. <표 12>와 (그림 35)에서 보는 바와 같이 수행시간에서는 k-means 알고리즘이 가장 우수한 성능을 보이는 반면 CL이 가장 긴 수행시간이 소요된다. CL은 다른 기법에 비하여 정확률은 우수하지만, 수행시간은 다른 알고리즘들에 비하여 너무 저조하여 분류하려는 문서수가 많은 경우에는 적용이 곤란한 것을 알 수 있다.

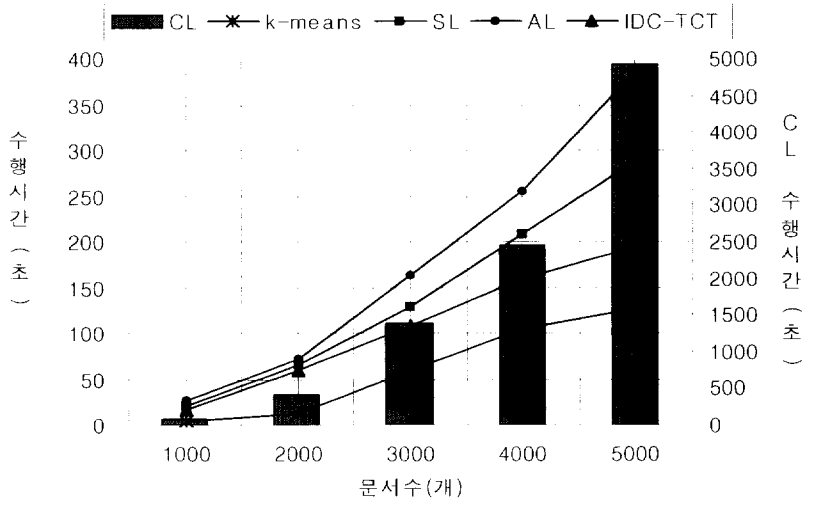
IDC-TCT는 정확률이 저조한 k-means 알고리즘을 제외하면 나머지 알고리즘들 보다 가장 빠른 수행성능을 보인다. IDC-TCT는 유사도 비교 알고리즘 수행 특성상, 초기에 문서노드의 배치에 큰 영향을 받는다. 즉, 유사한 문서끼리 이진트리 관계에 따라 연결되어 있는 문서집합은 초기에 소속 문서가 많은 클러스터를 형성한다. 이러한 경우에, 이행적 폐쇄 추론에 따라 유사도 계산 횟수가 대폭적으로 감소되어, 전체적인 수행시간이 줄어든다.

<표 12> 초기 클러스터링 수행속도 비교

<Table 12> The comparison of a processing time for an initial clustering

(단위: 초)

알고리즘 / 문서수	k-means 알고리즘	SL	CL	AL	IDC-TCT
1000	3.8	19.6	87.6	26.8	17.2
2000	11.4	64.4	417.3	71.6	59.4
3000	58.6	127.9	1385.2	163.2	108.7
4000	104.5	207.3	2439.7	255.3	158.3
5000	126.7	286.8	4932.7	385.9	194.3



(그림 35) 초기 클러스터링 수행속도 비교

(Figure 35) The comparison of a processing time for an initial clustering

6) 점진적 문서 클러스터링의 수행속도 측정

IDC-TCT는 이전에 수행된 클러스터링 정보를 저장하여 차후 새로운 문서의 삽입에 활용함으로써, 중복적인 클러스터링 수행을 피하는 점진적 문서 클러스터링 알고리즘이다. 점진적 문서 클러스터링의 수행속도를 평가하기 위해 제안하는 기법에서 초기 클러스터링을 수행하는 FDC-TCT 수행결과와 점진적 문서 클러스터링을 수행하는 IDC-TCT와 비교 및 평가한다.

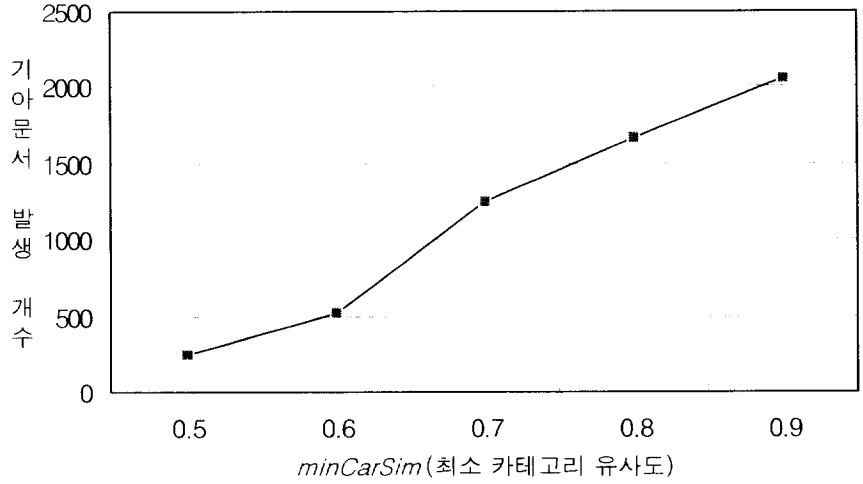
우선 <표 13>, (그림 36)에서는 *minCarSim*을 0.5에서 0.9로 설정하여 발생하는 기아문서의 개수를 측정한다.

<표 13> 최소 카테고리 유사도의 변화에 따른 기아문서 수

<Table 13> The number of starvation document according to a minimum category similarity

(단위: 개)

minCarSim	0.5	0.6	0.7	0.8	0.9
기아문서 발생 개수	251	523	1245	1671	2053



(그림 36) 최소 카테고리 유사도의 변화에 따른 기아문서 수

(Figure 36) The number of starvation document according to a minimum category similarity

(그림 36)은 *minCarSim*의 증가에 따라 새로운 문서와 공통 키워드 테이블과의 유사도인 $S(d_n, T_i)$가 모든 공통 키워드 테이블에 대하여, *minCarSim* 이상을 만족하지 않는 기아문서의 발생 개수를 측정한 것이다. *minCarSim*이 0.7인 지점을 보면 기아문서가 급격하게 발생하므로 *minCarSim*을 0.6으로 설정하여 점진적 문서 클러스터링을 수행하는 것이 타당하다.

IDC-TCT의 점진성을 측정하기 위하여 3000개의 문서를 클러스터링 한 결과를 저장하고, 새로운 문서의 삽입을 1000개씩 추가하여 점진적 문서 클러스터링의 수행 속도를 측정한다. 점진적 문서

클러스터링을 수행하지 않는 FDC-TCT 알고리즘과 점진적 문서 클러스터링을 수행하는 IDC-TCT와의 수행속도를 비교한다.

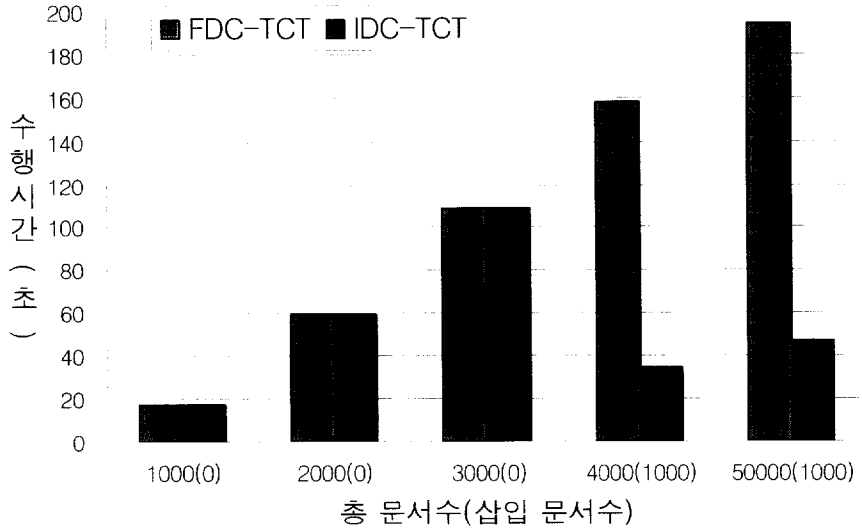
(그림 37)에서 알 수 있듯이 문서수 4000개부터는 IDC-TCT의 점진적 문서 클러스터링의 수행속도가 FDC-TCT에 비해 월등한 성능을 갖는다. 이것은 FDC-TCT는 새로운 문서의 삽입 때마다 다시 클러스터링을 수행해야 하지만, IDC-TCT는 이전의 클러스터링 정보를 이용함으로써 각 클러스터의 공통 키워드 테이블과의 유사도를 계산하는 시간을 필요로 하기 때문이다.

<표 14> IDC-TCT의 점진적 문서 클러스터링 성능 비교

<Table 14> The comparison of incremental document clustering performance for IDC-TCT

(단위: 초)

알고리즘 / 총 문서수(삽입문서수)	IDC-TCT	FDC-TCT
1000(0)	17.2	17.2
2000(0)	59.4	59.4
3000(0)	108.7	108.7
4000(1000)	34.7	158.3
50000(1000)	47.2	194.3



(그림 37) IDC-TCT의 점진적 문서 클러스터링 성능 비교

(Figure 37) The comparison of incremental document clustering performance for IDC-TCT

실험을 통해 본 논문에서 제안한 IDC-TCT는 계층적 클러스터링 알고리즘에 근사하는 정확률 및 재현율을 요구하면서, 빠른 수행 결과를 요구하는 문서 클러스터링 기법의 적용에 적절함을 알 수 있다.

4.4.2 Reuters-21578 문서집합

실험 2 문서인 Reuters-21578 문서집합은 문서 분류를 위해 검증된 데이터로서 영문으로 구성되어 있다. 이 문서 집합은 실험 1에서 사용한 네이버 문서에 비해 동일 범주에 소속되는 문서간에 다수의 공통 키워드로 강한 결합으로 이루어져 있다. 그리고 네이버 문서집합은 20개 범주에 소속되는 2000개의 문서로 구성되어 있지만, Reuters-21578 문서집합은 10개의 범주에 소속되는 2000개의 문서로 구성된다.

1) *minC*의 변화에 따른 클러스터 개수 변화

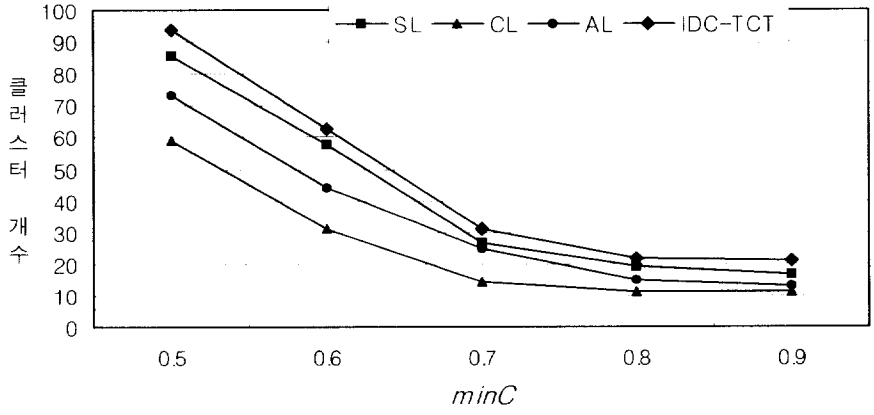
*minC*를 0.5에서 0.9까지의 변경에 따른 클러스터 개수의 변화는 <표 15>, (그림 38)과 같다. 네이버 문서 집합에 비하여 Reuters-21678 문서 집합은 범주의 개수가 10개가 적기 때문에 *minC*가 증가함에 따라 클러스터가 감소되는 특성은 동일하지만, 실험 1에 비하여 클러스터 개수가 급격하게 감소함을 알 수 있다.

<표 15> *minC*의 변화에 따른 클러스터 개수 비교

<Table 15> The comparison of cluster number according to a minimum coherence

(단위: 개)

알고리즘 \ *minC*	0.5	0.6	0.7	0.8	0.9
SL	86	58	27	19	17
CL	59	31	14	11	11
AL	73	44	25	15	13
IDC-TCT	94	63	31	22	21



(그림 38) *minC*의 변화에 따른 클러스터 개수 비교

(Figure 38) The comparison of cluster number according to a minimum coherence

2) *minC*의 변화에 따른 클러스터당 평균 문서수

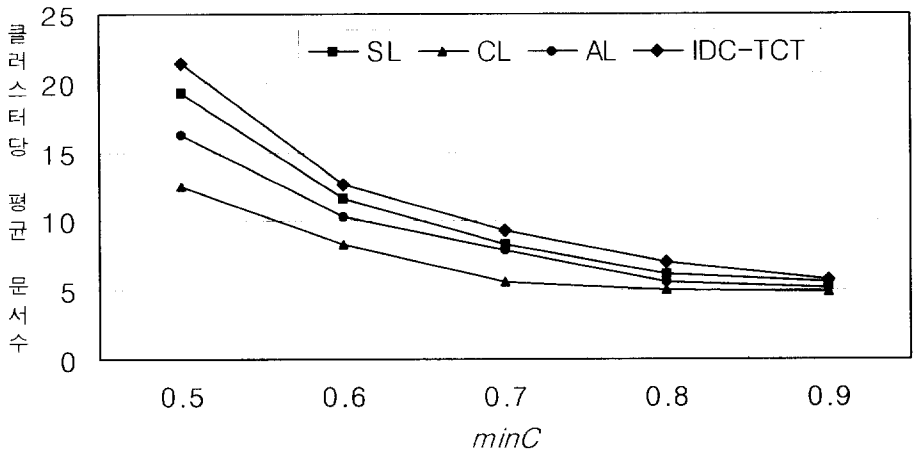
*minC*의 변화에 따른 클러스터에 소속되는 문서의 평균을 구하여 <표 16>과 (그림 39)에 나타낸다. Reuters-21678 문서 집합은 범주당 문서 개수가 많아 *minC*가 증가함에 따라 실험 1보다 평균 문서수가 더 많은 것을 알 수 있다.

<표 16> *minC*의 변화에 따른 클러스터당 평균 문서수 비교

<Table 16> The comparison of an average number of document per a cluster according to a minimum coherence

(단위: 개)

minC / 알고리즘	0.5	0.6	0.7	0.8	0.9
SL	19.3	11.6	8.4	6.2	5.6
CL	12.5	8.4	5.6	5.1	4.9
AL	16.2	10.3	7.9	5.6	5.2
IDC-TCT	21.4	12.7	9.3	7.1	5.8



(그림 39) *minC*의 변화에 따른 클러스터당 평균 문서수 비교

(Figure 39) The comparison of an average number of document per a cluster according to a minimum coherence

3) *minC*의 변화에 따른 평균 클러스터 응집도

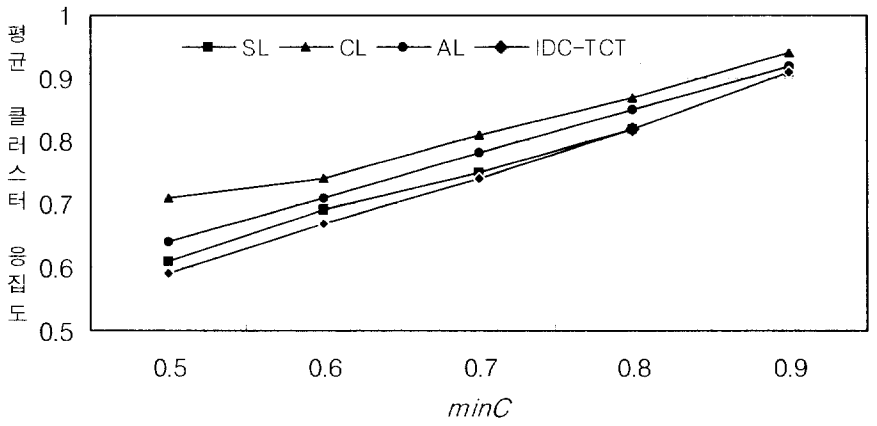
*minC*의 변화에 따른 평균 클러스터 응집도를 <표 17>과 (그림 40)에 나타낸다. 실험 1과 비슷한 패턴을 보이고, *minC*가 0.5인 지점을 보면 SL, CL, AL이 IDC-TCT보다 초기에 생성되는 클러스터의 질이 우수함을 알 수 있다. 그러나 IDC-TCT는 *minC*의 증가에 따라 엄격한 클러스터링의 영향 때문에 다른 알고리즘들과의 평균 클러스터 응집도의 격차가 해소됨을 알 수 있다.

<표 17> *minC*의 변화에 따른 평균 클러스터 응집도 비교

<Table 17> The comparison of an average for cluster coherence according to a minimum coherence

(단위: 비율)

알고리즘 \ *minC*	0.5	0.6	0.7	0.8	0.9
SL	0.61	0.69	0.75	0.82	0.91
CL	0.71	0.74	0.81	0.87	0.94
AL	0.64	0.71	0.78	0.85	0.92
IDC-TCT	0.59	0.67	0.74	0.82	0.91



(그림 40) *minC*의 변화에 따른 평균 클러스터 응집도 비교

(Figure 40) The comparison of an average cluster coherence according to a minimum coherence

4) 정확률, 재현율 및 F-Measure

앞의 (그림 39)를 보면 *minC*가 0.6인 지점에서 클러스터당 평균 문서수가 급격하게 감소하는 것을 알 수 있다. 그러므로 정확률, 재현율, F-Measure 및 클러스터링 수행속도 측정을 위해 *minC*를 *0.6, minS*를 *0.6, k*를 *10, minMer*를 *0.7*로 설정하여 각 알고리즘들에 대하여 클러스터링을 수행한 결과는 <표 18>과 (그림 41)과 같다.

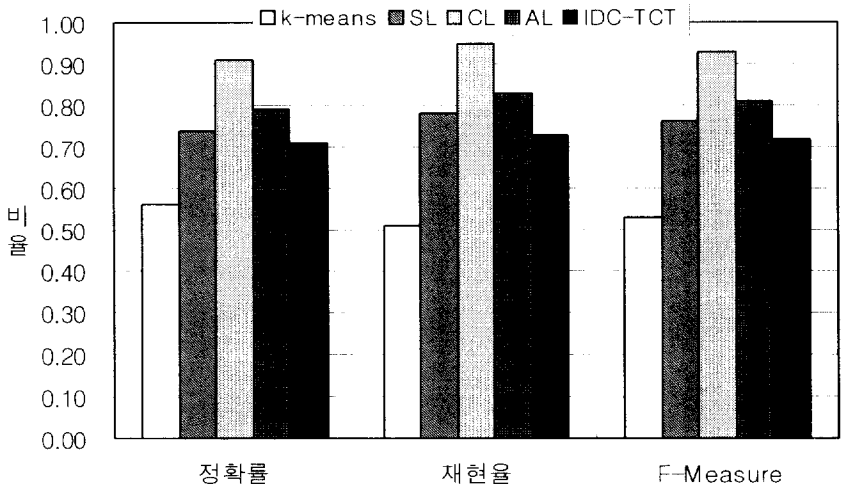
(그림 41)과 같이 네이버 문서 집합에 비하여 Reuters-21578 문서 집합은 문서간의 유사도가 큰 문서들로 구성되었기 때문에, SL, CL, AL에서는 실험 1보다 좋은 성능을 보인다. 그러나 이행적 폐쇄 이론에 따라 모든 문서와의 유사도를 비교하지 않는 IDC-TCT에서는 문서의 질과 상관없이 거의 비슷한 성능을 보이는 특성이 있다.

<표 18> 정확률, 재현율 및 F-Measure 비교

<Table 18> The comparison of a precision, a recall, and a F-Measure

(단위: 비율)

알고리즘 / 평가요소	k-means 알고리즘	SL	CL	AL	IDC-TCT
정확률	0.56	0.74	0.91	0.79	0.71
재현율	0.51	0.78	0.95	0.83	0.73
F-Measure	0.53	0.76	0.93	0.81	0.72



(그림 41) 정확률, 재현율 및 F-Measure 비교

(Figure 41) The comparison of a precision, a recall, and a F-Measure

5) 초기 클러스터링 수행속도 비교

실험 1과 같이 문서수를 1000, 2000, 3000, 4000, 5000개로 구성하여 k-means 알고리즘, SL, CL, AL과 비교하여 수행 속도를 측정한다.

<표 19>와 (그림 42)를 보면 실험 1과 유사하게 다른 알고리즘들은 거의 유사한 수행속도를 보이지만 IDC-TCT는 현저한 수행속도의 향상을 보인다. 이것은 네이버 문서 집합은 20개의 범주로 구성되어 있지만 Routers-21578 문서 집합은 10개의 범주 문서로

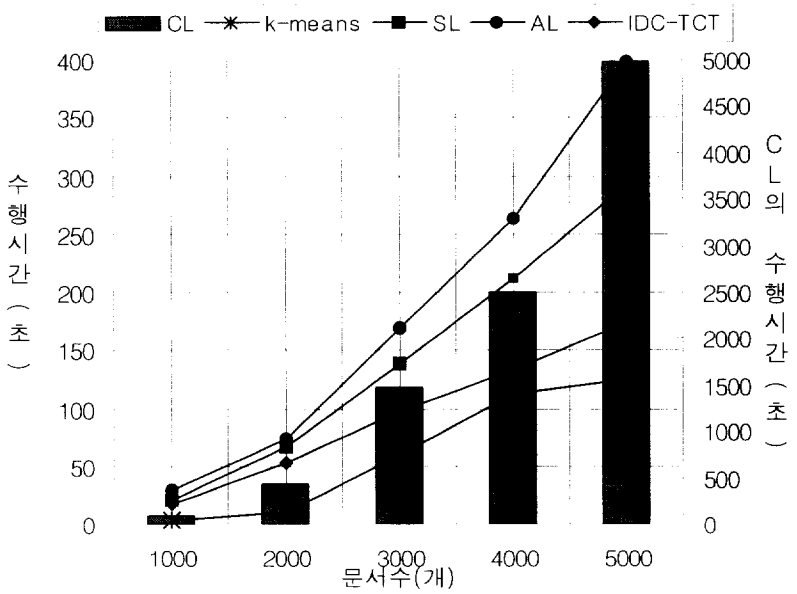
구성되어 있기 때문에, IDC-TCT는 IC 단계에서 소속 문서수가 많은 규모가 큰 클러스터를 형성하여 유사도 비교회수가 감소하였기 때문이다. 그러므로 IDC-TCT는 소속 문서수가 많고 클러스터 개수가 적은 문서 집합에서 우수한 성능을 갖는다.

<표 19> 초기 클러스터링 수행속도 비교

<Table 19> The comparison of a processing time for an initial clustering

(단위: 초)

알고리즘 / 문서수	k-means 알고리즘	SL	CL	AL	IDC-TCT
1000	3.7	20.3	88.4	29.3	16.3
2000	10.9	67.3	424.8	74.3	52.5
3000	59.3	138.2	1473.9	168.5	96.8
4000	112.5	211.8	2498.5	263.4	132.8
5000	125.3	292.1	4975.4	397.8	174.7



(그림 42) 초기 클러스터링 수행속도 비교

(Figure 42) The comparison of a processing time for an initial clustering

5. 결론

기존의 문서 클러스터링 기법에서 반복적 클러스터링 알고리즘과 계층적 클러스터링 알고리즘은 정확률이 우수하거나 수행속도가 우수한 상호 장단점이 있다. 웹 검색 결과 분류는 신속한 결과를 요구하면서 정확률을 보장해야 한다.

따라서 본 논문에서는 웹 검색 결과 분류와 같이 일정 수준의 정확률 및 재현율을 요구하면서 신속한 문서 클러스터링이 가능한 IDC-TCT 기법을 제안하였다. IDC-TCT는 사전 범주 선정과 같은 전문가의 개입을 요구하지 않는 문서 클러스터링 방법으로서 자체적으로 문서간의 유사도를 계산하여 클러스터링을 수행한다. 또한 IDC-TCT는 정확률 및 재현율이 우수한 계층적 클러스터링 알고리즘의 장점을 보장하면서도 저속의 수행 속도라는 단점을 개선하였다.

이것은 이행적 폐쇄 추론을 접목한 TCT를 구성하여 문서간 유사도 계산 횟수를 대폭적으로 감소시킴으로써 가능하다. 이에 상응하는 정확률 및 재현율의 손실을 줄이기 위하여 클러스터 생성 알고리즘 및 유사도 비교 알고리즘을 제안하였다.

그리고 IDC-TCT는 점진적 문서 클러스터링이 가능하도록 설계하였다. 즉, 문서 클러스터링 수행 후 각 클러스터별로 공통 키워드 테이블을 유지함으로써, 이후 문서의 삽입과 삭제에 대하여 문서 유사도를 재계산하지 않는 점진적인 클러스터링이 가능한 기법

이다.

제안한 IDC-TCT의 효율성을 검증하기 위하여 국문과 영문으로 구성된 문서 집합에 대하여 비교 실험을 수행하였다. 실험을 통해 IDC-TCT는 F-Measure는 AL에 비하여 약 9% 감소하였으나, k-means 알고리즘에 대해서는 약 39% 향상되었고, 초기 클러스터링 수행속도에서는 AL보다 약 82% 향상된 것을 알 수 있다.

특히, IDC-TCT의 초기 클러스터링 수행시간은 초기 클러스터의 규모와 문서수의 증가에 따라 더 좋은 결과를 갖는다. 그리고 점진적 문서 클러스터링의 성능 측정을 통해 점진적 문서 클러스터링을 수행하지 않는 FDC-TCT에 비해 속도가 크게 향상되어, 새로운 문서의 삽입에 대하여 신속한 클러스터링이 가능함을 알 수 있었다.

본 논문에서 제안한 IDC-TCT 기법은 웹 검색 결과 분류뿐만 아니라 e-mail, 신문기사 분류와 같이 텍스트기반 문서 분류기법에 활용할 수 있다.

향후 연구과제로는 점진적 문서 클러스터링에서 삭제되는 클러스터의 소속 문서들을 기아 문서 테이블에 전송하기 전에 삭제 결정된 다른 클러스터와 클러스터링을 고려하는 방법과 이에 대한 효율성에 대한 연구가 필요하다.

참 고 문 헌

[1] O. Zamir and O. Etzioni, "Web Document Clustering : A Feasibility Demonstration," *Proceedings of the 19th International ACM SIGIR Conference on Research and Development in Information Retrieval(SIGIR'98)*, pp. 46-54, 1998.

[2] D. S. Modha and W. S. Spangler, "Clustering Hypertext with Applications to Web Searching," *Proceedings of ACM Hypertext Conference*, PP. 147-150, 2000.

[3] A. E. Monge and C. P. Elkan, "An efficient domain-independent algorithm for detecting approximately duplicate database records," *Proceeding of the ACM SIGMOD workshop on research Issues on knowledge discovery and data mining*, pp. 125-130, 1997.

[4] H. Yu, J. Han, and K. C. Chang, "PEBL: Web Page Classification without Negative Examples," *IEEE transactions on knowledge and data engineering*, vol. 16, no. 1, pp. 70-81, Jan. 2004.

[5] R. Douglass, D. Cutting, J. Karger, P. Jao, O. Pedersen and W. John, "Scatter/Gather : A cluster-based Approach to Browsing Large Document Collections," *15th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 318-329, 1992.

[6] R. Fagin, Y. Maarek, I. Ben-Shaul, and D. Pelleg, "Epthemeral document clustering for web applications," *IBM Research Report RJ 10186*, Apr. 2000.

[7] G. Karypis, E. H. Han, V. Kumar, "A Hierarchical Clustering Algorithm Using Dynamic Modelling," *IEEE Computer*, vol. 32, no. 8, pp. 68-75, 1999.

[8] G. Karypis, V. Kumar, "A fast and highly quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 34-46, 1999.

[9] M. Ester, H. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental Clustering for Mining in a Data Warehousing Environment," *Proceedings of the 24th VLDB Conference*, pp. 245-248, 1998.

[10] K. M. Hammouda and M. S. Kamel, "Efficient phrase-based document Indexing for web document clustering," *IEEE transactions on knowledge and data engineering*, vol. 16, no. 10, pp. 1279-1296, Oct. 2004.

[11] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, Issue. 1, pp. 1-47, 2002.

[12] M. L. Antonie and O. R. Zaiane, "Text document categorization by term association," *In proceeding of the second IEEE International Conference on Data Mining(ICDM)*, pp. 19-26, Dec. 2002.

[13] U. Hiroshi, M. Takao and I. Shioya, "Improving Text Categorization By Resolving Semantic Ambiguity," *In Proceeding of the IEEE Pacific Rim Conference on*

Communications Computers and Signal processing (PACRIM), pp. 796-799, 2003.

[14] D. D. Lewis and M. Ringuette, "A comparison of two learning algorithms text categorization," *In 3rd. Annual Symposium on Document Analysis and Information Retrieval*, pp. 81-93, 1994.

[15] Y. Yang, "Expert Network : Effective and efficient learning form human decisions in text categorization and retrieval", *17th ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 13-22, 1994.

[16] F. Sebastiani, "Machine learning in automated text categorization," *ACM Computing Surveys*, vol. 34, no. 1, pp. 1-47, 2002.

[17] J. Kacprzyk, *Text Mining and its Applications*, Springer-Verlag publisher, Berlin, 2002.

[18] E. Rasmussen, *Clustering Algorithms In Information Retrieval*, Prentice-Hall PTR, New Jersey, 1992.

[19] W. Klosgen and J.M. Zytkow, *Handbook of Data Mining and Knowledge Discovery*, Oxford University Press, New York, 2002.

[20] J. C. Song and J. Y. Shen, "A web document clustering algorithm based on concept of neighbor," *Proceedings of the second international conference on machine learning and cybernetics*, pp. 46-50, Nov. 2003.

[21] P. Soucy and G. W. Mineau, "A Simple KNN Algorithm for Text Categorization," *Proceeding of 1st. IEEE international conference on data mining*, vol. 28, pp. 647-648, 2001.

[22] Y. Yang, "An Evaluation of Statistical Approaches to Text Categorization," *Journal of Information Retrieval*, vol. 1 no. 1, pp. 67-88, 1999.

[23] R. E. Schapire and Y. Singer, "Improved Boosting Algorithms Using Confidence-rated Predictions," *Machine Learning*, vol. 37, no. 3, pp. 297-336, 1999.

[24] P. Domingos and M. J. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, no. 2, pp. 103-130, 1997.

[25] P. Domingos and M. J. Pazzani, "Beyond Independence: Conditions for the optimality of the simple bayesian classifier," *In proceedings of the 13th international conference on machine learning*, pp. 105-112, 1996.

[26] S. Dumais, J. Platt, D. Heckerman, M. Sahami, "Inductive Learning Algorithms and Representations for Text Categorization," *Proceedings of the 7th International Conference on ACM-CIKM*, pp. 148-155, 1998.

[27] Y. Bao and N. Ishii, "Combining Multiple K-Nearest Neighbor Classifiers for Text Classification by Reducts," *Proceeding of the fifth International Conference on Discovery Science*, pp. 340-347, 2002.

[28] S. H. Lim, "A comparative Evaluation of Korean Text Categorization Based on k-NN Learning," *Proceeding of the*

International Conference on Artificial Intelligence, pp. 755-758, 2002.

[29] E. H. Han, G. Karypis and V. Kumar, "Text categorization using weight adjusted k-nearest neighbor classification," *Proceeding of the fifth Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pp. 53-65, 1999.

[30] T. Ault and Y. Yang, "kNN Rocchio and Metrics for Information Filtering at TREC-10," *In The 10th Text Retrieval Conference(TREC-10)*, pp. 127-134, 2001.

[31] M. A. Hernandez and S. J. Stolfo, "The merge/purge problem for large databases," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 127-138, May. 1998.

[32] Y. Zhao and G. Karypis, "Web clustering: Evaluation of hierarchical clustering algorithms for document datasets," *Proceedings of the eleventh international conference on Information and knowledge management*, pp. 515-524, Nov. 2002.

[33] R. O. Duda and P. E. Hart, "An algorithm for text categorization with SVM," *In Proceeding of the IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, vol. 1, pp. 47-50, 2002.

[34] P. S. Bradley, U. M. Fayyad, "Refining Initial Points for k-means Clustering," *Proceedings of the 5th International Conference on Machine Learning*, pp. 254-258, 1998.

[35] T. Kanung, "The Analysis of a Simple k-Means Clustering Algorithm," *Proceedings of ACM Symposium on computational Geometry*, pp. 542-546, Jun. 2000.

[36] D. R. Hill, "A Vector Clustering with Committees," *Proceedings of ACM SIGIR Conf. Research and Development in Information Retreval*, pp. 199-206, 2002.

[37] Khaled Alsabti, et al, "An efficient k-means Clustering Algorithm", *IIPS 11th International Parallel Processing Symposium*, pp. 128-131, 1998.

[38] W. L. Low, M. L. Lee, and T. W. Ling, "A knowledge-based approach for duplicate elimination in data cleaning," *Information Systems*, vol. 26, no. 8, pp. 585-606, Dec. 2001.

[39] A. E. Monge and C. P. Elkan, "An efficient domain-independent algorithm for detecting approximately duplicate database records," *Proceedings of the ACM SIGMOD workshop on research Issues on knowledge discovery and data mining*, pp. 125-130, 1997.

[40] L. Zhuang and H. Dai, "A maximal frequent itemset approach for Web document clustering," *Computer and Information Technology 2004(CIT '04)*, pp. 970-977, Sep. 2004.

[41] D. Lin, "An Information-Theoretic Definition of Similarity," *Proceedings of 15th International Conference Machine Learning*, pp. 296-304, 1998.

[42] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing and*

Management, vol. 24, no. 5, pp. 513-523, 1988.

[43] G. Salton, A. Wang, and C. Yang, "A vector space model for information retrieval," *Journal of the American Society for Information Science*, vol. 18, pp. 613-620, 1975.

[44] O. Zamir and O. Etzioni, "Fast and intuitive clustering of web documents," *KDD97*, pp. 287-290, 1997.

[45] Y. Wang, "Use link-based clustering to improve web search results," *Proceedings of 2nd. conference on web information systems engineering*, vol. 1, pp. 115-123, Dec. 2001.

[46] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore, "Partitioning-based clustering for web document categorization," *Decision Support Systems*, vol. 27, no. 3, pp. 329-241, 1999.

[47] R. Kosala and H. Blockeel, "Web Mining Research: A Survey," *ACM SIGKDD Explorations Newsletter*, vol. 2, no. 1, pp. 1-15, 2000.

[48] O. Zamir and O. Etzioni, "Grouper: A Dynamic Clustering Interface to Web Search Results," *computer Networks*, vol. 31, no. 11-16, pp. 1361-1374, 1999.

[49] S. Soderland, "Learning Information Extraction Rules for Semi-Structured and Free Text," *Machine Learning*, vol. 34, no. 1-3, pp. 233-272, 1999.

[50] A. Sthehl, G. Joydeep, and R. Mooney, "Impact of similarity measures on web-page clustering," *Proceedings of the 17th*

National conference on Artificial Intelligence, pp. 30-31, 2000.

[51] O. Zamir and O. Etzioni, "Web document clustering: a feasibility demonstration," *Proceedings of SIGIR '98*, pp. 46-54, 1998.

[52] C. Cortes and V. Vapnik, "Support-vector Networks", *Journal of Machine Learning*, vol. 20, pp 273-297, 1995.

[53] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," *Proceedings of 7th International Conference on Information Retrieval and Knowledge-Management ACM-CIKM-98*, pp. 148-155, 1998.

[54] H. Drucker, D. Wu, and V. Vapnik, "Support vector machines for spam categorization," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1048-1054, 1999.

[55] G. Paab, E. Leopold, M. Larson, J. Kindermann, and S. Eickeler, "SVM Classification Using Sequences of Phonemes and Syllables," *European Conference on Machine Learning(ECML)*, pp. 19-23, 2002.

[56] S. Fuamura and M. Fumihiro, "Automatic Indexing by Stop Word Removal on Scientific and Technical Documents Written in English," *Information Processing Society of Japan*, vol. 28, no. 7, 1987.

[57] G. Salton and M. J. MCGill, *Introduction to Modern Information Retrieval*, McGraw-Hill Computer Science Series, New York, 1983.

[58] W. B. Frakes and R. Baeza-Yates, *Information Retrieval : Data Structures and Algorithms*, Prentice-Hall, Englewood Cliffs, New Jersey, 1992.

[59] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1998.

[60] T. M. Cover and J. A. Thomas, *Elements of information theory*, John Willey & Sons, New York, US, 1991.

[61] 고석범, 윤성대, "FDC-TCT를 이용한 웹 문서 클러스터링 성능 개선 기법," 한국정보처리학회논문지 D, 제 12-D권, 4호, pp. 637-646, 2004. 08.

[62] 강동혁, 주길홍, 이원석, "대용량 문서 데이터베이스를 위한 효율적인 점진적 문서 클러스터링 기법," 한국정보처리학회 논문지 D, 10-D권, 1호, pp. 57-66, 2003. 02.

[63] 방선이, 양제동, 양형정, "k-NN 분류 알고리즘과 객체 기반 시소러스를 이용한 자동 문서 분류," 한국정보과학회논문지: 소프트웨어 및 응용, 31권, 9호, pp. 1204-1217. 2004. 09.

[64] 김제욱, 김한준, 이상구, "베이지언 문서분류시스템을 위한 능동적 학습 기반의 학습문서집합 구성방법," 한국정보과학회논문지: 소프트웨어 및 응용, 29권, 12호, pp. 966-978, 2004. 09.

[65] 이신원, 오형진, 안동언, 정성종, "클러스터 중심 결정 방법을 개선한 K-Means 알고리즘의 구현," 한국정보처리학회 B, 11-B권, 7호, 2004. 12.

[66] 강승식, "HAM : 한국어 분석 모듈," http://nlp.kookmin.ac.kr.

[67] 박우창 외, 데이터 마이닝:개념 및 기법, 자유아카데미, 2003. 09.

[68] 고석범, 윤성대, "데이터준비를 위한 XML 기반의 분산 MDR 검색 시스템 설계," 한국멀티미디어학회논문지, 7권 9호, pp. 1329-1338, 2004. 09.

감사의 글

박사학위 졸업을 앞두고 대학 입학과 함께 14년이라는 시간 속에 고마운 분들과 여러 가지 소중한 기억들이 떠오릅니다. 박사수료 후 국방의 의무를 위한 육군사관학교에서의 생도들과의 생활, 소중한 결혼 등 이벤트도 참으로 많았던 기간이었습니다. 그래서 나에게 박사학위는 새로운 출발을 의미하면서도 더없이 소중하게 느껴지나 봅니다.

소중하고 고마운 분들이 너무 많아서 학위논문 분량보다 많은 감사의 마음을 기술해도 부족하지만, 무엇보다도 항상 부족한 저에게 연구의 방향뿐만 아니라 인생을 살아가는 법과 인간관계의 귀중한 교훈을 주시면서 지속적인 관심, 질책, 칭찬을 아끼지 않으신 윤성대 교수님께 감사와 존경의 마음을 전하고 싶습니다. 분주한 가운데도 연구의 핵심과 논문의 책임을 강조하시며 큰 깨달음을 주신 정신일 교수님께 감사드리고, 저 자신보다도 더욱 세심하고 꼼꼼한 분석을 통해 논문의 완성도를 높여주신 여정모 교수님께 감사드리고, 학문적인 지적뿐만 아니라 올바른 논문의 기술에 대해 깨달음을 주신 조우현 교수님께 감사드리며, 날카로운 지적과 앞으로의 연구에 방향을 제시해 주신 양황규 교수님께 존경과 감사를 드립니다. 박사과정 중에 연구에 기초를 마련해 주신 박만곤 교수님, 정순호 교수님, 박승섭 교수님, 박흥복 교수님, 김창수 교수님, 이경현 교수님, 김영봉 교수님께도 깊은 존경과 감사를 드립니다.

기쁨과 슬픔을 함께 나누며 연구에 매진하던 연구실 가족들에게도 깊은 감사를 드립니다. 먼저 항상 따스한 관심과 조언을 주신 김용덕 교수님께 감사드리고, 연구 조언뿐만 아니라 힘들 때면 항상 따스하게 응원해주신 황순환 박사님, 친형처럼 관심을 아끼지 않았던 박현호 선생님, 친동생보다 더욱 소중한 박상일 선생, 앞으로 연구실을 꾸려나갈 윤종찬 선생님, 임선자 선생님과 모든 연구실 가족들에게도 감사와 함께 앞으로의 발전을 기원합니다.

저의 연구의 기초를 마련해주신 와세다 대학의 Mitsuo Gen 교수님과 아시카가 대학의 Yasuhiro Tsujimura 교수님, Yokota 교수님께도 깊은 감사를 드리고, 학부과정부터 항상 관심과 조언을 아끼지 않으시는 동서대학교의 이재욱 교수님께 깊은 존경과 감사를 드리며, 항상 다정한 말씀으로 용기를 불어 넣어주신 조정복 교수님, 임효택 교수님께도 수줍은 감사를 드립니다. 그리고 저에게 항상 조언과 연구의 중요성을 일깨워주시는 육군사관학교에서 생도 교육에 여념이 없으시는 양교수님, 나교수님, 이교수님, 오교수님, 박교수님께도 존경과 감사와 함께 건승을 기원합니다. 그리고 논문 발표를 여러모로 도와주신 한양대의 문치웅 교수님께 감사드리고, 일본 유학시절부터 용기와 사랑만 주시는 국방부의 이중희 선생님께도 감사의 말씀을 드리고 싶습니다.

육사 생활동안 함께 기쁨과 슬픔을 나누었던 내 사랑하는 동기 은구, 영진, 재범, 재성, 호식, 승훈, 지석, 형준, 그리고 교석, 창훈, 상태에게도 감사와 앞으로의 발전을 기원합니다. 그리고 멀리 떨어져있지만 항상 서로를 아끼며 살피는 죽마고우 무호, 지훈, 래율

에게도 고마움을 전하고, 항상 기쁨과 슬픔을 나누는 봉오, 진숙 부부에게도 감사와 기쁨을 함께합니다.

학위 취득에 누구보다 자랑스러워하고 기뻐하실 먼 세상의 아버지께 미안함과 감사를 드리고, 항상 따스한 마음으로 손자를 아껴주시는 할머니께 감사드리고, 부족한 저를 항상 신뢰하고 묵묵히 바라만 보시면서 한평생을 아들에게 헌신하신 어머니께 사랑과 존경의 마음을 전합니다. 그리고 어린 시절부터 항상 아버지 역할을 다하시며 저를 인도해 주신 형님께 감사를 드리며, 형수님과 귀여운 조카 연지, 나영, 혁빈에게도 사랑한다는 마음을 전합니다. 아버지, 어머니처럼 항상 따스한 사랑을 주시는 장인어른과 장모님께도 감사와 기쁨을 함께합니다.

그리고 제가 연구의 길을 가도록 이끌어 주셨고 바다의 등대처럼 어려운 순간마다 희망으로 이끌어 주시는 고 이종국 교수님께 존경과 감사를 드립니다.

나에게 이토록 사람을 사랑하는 마음을 심어주고 베풀어주는 나의 사랑하는 아내 미란에게 더없는 감사를 드리며 기쁨을 함께하고 싶습니다. 그리고 이제 세상을 만난 지 넉달이 되가는 우리의 보배 유림에게도 탄생의 축하와 함께 깊은 사랑을 전합니다.

2005년 12월 28일

한 해를 마무리하고 새로운 도약을 준비하면서.....