Journal of Korea Multimedia Society Vol. 9, No. 1, January 2006(pp. 1-10)

이행적 폐쇄트리를 기반으로 한 점증적 웹 문서 클러스터링

고석범[†], 윤성대[††]

요 약

기존의 문서 클러스터링 기법에는 k-means와 같이 수행속도가 우수한 기법과, 분류의 정확률이 우수한 계층적 집괴 클러스터링 기법이 있다. 두 기법은 각각 분류의 정확률 저하와 저속의 수행속도로서 상호 단점을 가지며, 새로운 문서를 삽입 할 때마다 문서 유사도를 재계산해야 하는 문제가 있다. 웹 정보의 특성은 잦은 문서의 추가를 통해 정보를 축적하는 것이다. 따라서 본 논문에서는 정확률이 우수한 계층적 집괴 클러스터링 기법을 기반으로 수행속도를 향상 시킬 수 있는 이행적 폐쇄 트리 기법을 제안하고, 또한 새로운 문서의 삽입과 삭제에 우수한 점증적인 클러스터링이 가능한 기법을 제안한다. 제안된 기법의 효율성을 검증하기 위하여 기존의 알고리즘과 정확률, 재현율, F-Measure, 수행속도에 대해 비교 평가 및 분석한다.

An Incremental Web Document Clustering Based on the Transitive Closure Tree

Suc-Bum Ko[†], Sung-Dae Youn[††]

ABSTRACT

In document clustering methods, the k-means algorithm and the Hierarchical Alglomerative Clustering(HAC) are often used. The k-means algorithm has the advantage of a processing time and HAC has also the advantage of a precision of classification. But both methods have mutual drawbacks, a slow processing time and a low quality of classification for the k-means algorithm and the HAC, respectively. Also both methods have the serious problem which is to compute a document similarity whenever new document is inserted into a cluster. A main property of web resource is to accumulate an information by adding new documents frequently. Therefore, we propose a new method of transitive closure tree based on the HAC method which can improve a processing time for a document clustering, and also propose a superior incremental clustering method for an insertion of a new document and a deletion of a document contained in a cluster. The proposed method is compared with those existing algorithms on the basis of a precision, a recall, a F-Measure, and a processing time and we present the experimental results.

Key words: Document Clustering(문서 클러스터링), Incremental Clustering(점증적 클러스터링), Text Mining(텍스트마이닝), Transitive Closure(이행적 폐쇄)

1. 서 론

방대한 웹을 이용한 정보 수집은 크게 두 가지 특성으로 고려 할 수 있다. 첫째는 검색정보의 대용량성이다[1]. 즉, 포털사이트에서 제공되는 검색결과는 일일이 살펴보기에 힘든 대용량이다. 둘째는 정보검

※ 교신저자(Corresponding Author) : 고석범, 주소 : 부산광역시 남구 대연3동 599-1(608-737), 전화 : 051)620-6398, E-mail : sbko73@yahoo.co.kr
접수일 : 2005년 6월 23일, 완료일 : 2005년 8월 16일
[†] 준회원, 부경대학교 대학원 전자계산학과
[††] 정회원, 부경대학교 전자컴퓨터정보통신공학부 교수
(E-mail :sdyoun@pknu.ac.kr)
※ 이 논문은 2005년도 두뇌한국21(BK21) 사업에 의해 지원되었음.

색의 중복성이다. 즉, 검색자들은 웹 리소스의 잦은 정보의 갱신을 신뢰함으로써, 주기적으로 동일 검색어를 사용하여 중복적인 검색을 한다. 첫번째 특성은 웹 문서 클러스터링 알고리즘의 필요성을 의미하며, 두번째 특성은 문서 클러스터링이 점증적 클러스터링(Incremental clustering)이 가능해야 함을 의미한다. 여기에서 점증적 클러스터링이란 문서 클러스터링을 수행하고 나서 차후 새로운 문서의 삽입과 삭제에 대해, 미리 저장한 클러스터링 정보를 이용함으로써 중복적인 문서 유사도(Document similarity) 계산을 피하기 위한 방법이다[1,2]. 대개의 문서 클러스터링 기법은 클러스터링 후 그 정보를 저장하지 않으므로 새로운 문서가 삽입 및 삭제 될 때마다 새롭게 문서 유사도를 비교하여 클러스터링을 수행한다.

문서 클러스터링 알고리즘은 베이지안 분류와 같은 문서 분류 알고리즘에 비해, 초기 범주(Category) 선정과 같은 전문가의 개입을 필요로 하지 않는다. 그래서 웹 검색 결과 분류에는 문서 클러스터링 알고리즘이 적절하며, 최근 많은 연구가 진행되고 있다[2,3]. 문서 클러스터링 알고리즘에서 가장 자주 사용되는 기법은 수행 속도가 우수한 k-means 알고리즘과, 정확률 및 재현율이 우수한 계층적 집괴 클러스터링이다. 그러나 k-means 알고리즘은 정확률이 낮고, 계층적 집괴 클러스터링은 수행속도가 느리다는 상호 단점이 있다[4,5,12].

따라서, 본 논문에서는 기존의 정확률 및 재현율이 우수한 계층적 집괴 클러스터링의 장점을 수용하면서, 수행속도를 향상시키기 위한 이행적 폐쇄 트리(Transitive closure tree) 기반의 문서 클러스터링 알고리즘을 제안한다. 특히 제안하는 알고리즘에 점증적인 기법이 가능하도록 설계하여, 새로운 문서의 삽입과 삭제에 대하여 문서 클러스터링을 위한 중복적인 계산을 수행하지 않는 기법을 제안한다.

이후 2장에서는 본 논문에서 제안하는 문서 클러스터링 알고리즘인 IDC-TCT와 관련된 기존의 연구들에 관하여 기술한다. 3장에서는 IDC-TCT의 정의 및 설계에 관하여 기술한다. 4장에서는 IDC-TCT의 성능을 평가하기 위하여 기존의 k-means 알고리즘과, AL 기법과 여러 가지 척도에 따라 비교 및 분석을 한다. 끝으로 5장에서는 본 연구에 대한 결론 및 향후과제를 제시한다.

2. 관련연구

문서 클러스터링 알고리즘은 사전에 전문가의 범주 구분을 요구하는 문서 범주화(Document categorization) 기법과는 달리, 자체적으로 문서간의 유사도를 비교하여 관련 있는 문서끼리 클러스터를 구성하는 것이다[1]. 그러므로 다양한 범주를 가지며, 잦은 갱신을 하는 웹 문서 및 웹 검색결과 분류를 위해서는 문서 클러스터링 기법의 적용이 타당하다. 따라서, 본 논문은 점증적인 클러스터링이 가능한 문서 클러스터링 기법을 적용하며 이에 관한 관련 연구를 기술한다.

문서 클러스터링 알고리즘으로 가장 선호하는 기법은 수행속도가 우수한 반복적 클러스터링 기법과 클러스터의 질(Quality)이 우수한 계층적 집괴 클러스터링 기법이 있다.

2.1 반복적 클러스터링

계산 복잡도가 작아 수행 속도가 우수한 반복적 클러스터링 기법의 대표적인 기법은 k-means 알고리즘이다. k-means 알고리즘은 k개의 클러스터를 목표로, 각 centroid를 중심으로 각 문서와의 거리를 구하여 클러스터링하고, 생성된 클러스터의 centroid를 각각 갱신하여 다시 반복적으로 클러스터링한다. 이것을 갱신되는 centroid의 변화가 임계치 이하일 때까지 반복하는 기법이다[5].

k-means 알고리즘은 구현이 간단하고 빠른 클러스터링 결과를 얻고자 할 때 유용한 알고리즘이다. 그러나, 계층적 집괴 클러스터링에 비해 클러스터링의 정확률(Precision) 및 재현율(Recall)가 저조한 특성이 있다.

2.2 계층적 집괴 클러스터링

계층적 집괴 클러스터링(HAC)은 초기에 각 문서들을 하나의 독립된 클러스터로 간주하여, 각 클러스터간의 유사도(Similarity)를 계산하여 점차적으로 클러스터를 병합하는 기법이다[4]. k-means 알고리즘에 비해 클러스터의 질이 우수한 계층적 집괴 클러스터링은 클러스터의 유사도를 결정하는 방법에 따라 단일링크, 완전링크, 집단평균링크로 구분한다[8].

(1) 단일 링크(Single Link : SL)

단일링크(SL)에서 클러스터간의 유사도는 다음 수식처럼 클러스터에 포함되어 있는 모든 문서들간의 유사도를 구하고, 이중에서 최대 유사도를 클러스터의 유사도로 정한다.

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

where, C_i : i 번째 클러스터

d_i : i 번째 문서

$\cos(d_i, d_j)$: 문서 d_i와 d_j간의 코사인 유사도

SL은 계산요구량이 작아 수행속도는 빠르나, 클러스터의 질을 판별하는 정확률 및 재현율은 낮은 단점이 있다.

(2) 완전 링크(Complete Link : CL)

완전링크(CL)의 클러스터 유사도는 다음 수식과 같이 클러스터에 포함되어 있는 문서들간의 유사도 중에서 최소 유사도를 클러스터의 유사도로 정한다.

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

CL은 HAC 기법 중 계산량이 가장 커서 수행속도가 가장 느리지만, 정확률 및 응답도는 가장 우수하다.

(3) 집단 평균 링크(group Average Link : AL)

집단 평균 링크(AL)에서 클러스터의 유사도는 다음 수식과 같이 클러스터에 포함되어 있는 문서들간의 유사도의 평균을 클러스터의 유사도로 결정하는 방법이다.

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

where, n_i : 클러스터 i에 포함된 문서들의 총수

이상의 HAC는 분류 후 클러스터의 질은 우수하지만, 클러스터에 속하는 모든 문서간의 유사도를 계산해야하므로 수행시간이 길다는 단점이 있다.

2.3 점증적 클러스터링

웹 문서의 클러스터링은 문서가 방대하기 때문에 그에 따른 문서간 유사도를 계산하기 위한 시간이 많이 든다. 그러므로, 이전에 수행한 클러스터링 결과를 저장하여, 새로운 문서의 삽입과 삭제에 대하여 중복적인 계산을 하지 않고 신속한 클러스터링이 가능한 방법인 점증적 클러스터링 기법이 필요하다[2,7,8]. 이를 위해 클러스터링이 수행된 결과를 체계적으로 저장한 후, 새로운 문서의 삽입과 삭제를 신속하게 처리하는 알고리즘이 필요하다.

2.4 이행적 폐쇄

이행적 폐쇄 이론은 데이터베이스의 두 레코드간의 비교에서 적용이 되었다[9,10]. 이행적 폐쇄 이론의 핵심은 레코드 a와 레코드 b가 유사하고, 레코드 b와 레코드 c가 유사하다면, 레코드 a와 c간의 유사도를 계산하지 않고도 레코드 a와 c는 유사하다고 추론하는 것이다[10].

이행적 폐쇄 이론은 문서 클러스터링 기법에 적용하면 문서 유사도 계산 회수를 대폭적으로 줄일 수 있다. 그러나, 그에 따른 정확률이 손실되는 단점이 있다. 정확률의 희생을 최소화하기 위해서는 유사도 계산 및 문서 비교를 효과적으로 조절 할 수 있는 기법이 필요하다.

3. IDC-TCT 설계

본 논문에서 제안하는 이행적 폐쇄트리 기반 점진적 문서 클러스터링(Incremental Document Clustering based Transitive Closure Tree : IDC-TCT) 기법은 이행적 폐쇄 트리(Transitive Closure Tree: TCT)로 문서를 구성하여 신속한 클러스터링을 수행하고, 그 결과 생성되는 클러스터를 체계적으로 저장하여 새로운 문서의 삽입과 삭제에 대해 신속하게 클러스터를 갱신하는 기법이다.

IDC-TCT의 클러스터 생성과정은 각 문서에 대하여 특징을 나타내는 키워드(Keyword)를 추출하고, 이를 TCT로 구성하며, 각 문서들은 키워드를 기반으로 유사도를 계산하고 유사한 문서끼리 클러스터를 형성한다. 그리고 클러스터링 완료 후 포레스트(Forest) 형태의 각 클러스터 정보를 활용하여, 새로운 문서의 삽입에 대하여, 문서간 유사도를 산출하기 위해 재계산하지 않는다. 그리고 문서의 삭제에 대해서도 기존의 클러스터 정보를 유지 할 수 있도록 조정하는 메커니즘을 갖는다. 전체적인 IDC-TCT의 수행과정은 다음과 같다.

Step 1. 모든 문서에 대하여 대표 키워드 추출

Step 2. 문서 집합을 TCT로 구성

Step 3. TCT 오퍼레이션에 따라 클러스터링 수행
Step 4. 각 클러스터별 공통키워드 추출
Step 5. 점증적 클러스터링 수행

3.1 IDC-TCT 요소 정의

본 논문에서 제안하는 IDC-TCT를 정의하기 위한 요소들을 다음과 같이 정의한다.

- D : 문서집합, $D=\{d_1, d_2, \cdots, d_n\}$
- d_i : 문서집합 D에서의 i번째 문서
- C_i : i 번째 클러스터로서, $C_i \subset D$
- $minS$: 최소 문서 유사도, $0 \leq minS \leq 1$
- $minC$: 최소 응집도, $0 \leq minsC \leq 1$
- $minMer$: 최소 병합도, $0 \leq minMer \leq 1$
- $minCarSim$: 최소 카테고리 유사도로서, $0 \leq minCarSim \leq 1$
- $|d_i|$: 문서 i에 포함된 유일한 키워드의 총 개수
- $|d_i \cap d_j|$: 문서 i와 j에 공통으로 포함된 유일한 키워드의 총 개수
- $|C_i|$: 클러스터 i에 포함된 문서의 총 개수

정의 1. 문서 유사도(Document similarity)
두 문서간의 유사도를 측정하기 위한 수식은 식 (3.1)과 같다.

$$S(d_i, d_j) = \frac{|d_i \cap d_j|}{\min(|d_i|, |d_j|)} \tag{3.1}$$

식 (3.1)은 기존의 계산량이 많은 벡터 기반의 유사도 계산식보다 간단하여 빠른 계산이 가능하다. 그리고 두 문서간의 유사도 계산에서 키워드 개수의 차이가 크더라도 공평하게 유사도를 계산 할 수 있도록 정규화를 반영한다.

정의 2. 클러스터 응집도(Cluster coherence)
클러스터 응집도는 클러스터에 소속되는 문서들 간에 얼마나 유사한 문서들로 구성되어 있는가에 대한 척도이다. 클러스터 응집도는 식 (3.2)와 같이 소속 문서들의 평균 유사도로 계산한다.

$$C(C_i) = \frac{\sum_{d_i \in C_i, d_j \in C_i - \{d_i\}} S(d_i, d_j)}{|C_i|} \tag{3.2}$$

정의 3. 이행적 폐쇄 트리(Transitive Closure Tree : TCT)

문서간의 유사도 계산회수를 대폭적으로 줄이기 위하여, 이행적 폐쇄 추론을 적용한 이행적 폐쇄트리를 다음과 같이 정의한다.

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

각 노드 v_i는 정렬된 문서 집합 D로부터 문서 d_i를 의미한다. 또한, 두 노드간의 링크 e_{ij}는 문서 d_i와 d_j 간의 문서 유사도 $S(d_i, d_j)$를 의미한다.

3.2 문서 키워드 추출

본 절에서는 원 문서(Origin document)로부터 특징을 이루는 키워드를 추출하는 기법에 관하여 기술한다.

임의의 문서로부터 특징을 나타내는 명사 키워드를 추출하기 위해 가장 보편적으로 사용되는 TFIDF(Term Frequency Inversed Document Frequency) 함수를 적용한다[11]. 문서 d_i에 단어 j에 대한 가중치 w_{ij}는 식 (3.3)과 같다.

$$w_{ij} = TF_{ij} \cdot \ln\frac{n}{IDF_j} \tag{3.3}$$

여기에서, TF_{ij}는 문서 d_i에서 단어 j의 출현 빈도수이고, IDF_j는 전체 문서수 n에 대하여 단어 j의 출현빈도수이다. 그러므로 임의의 문서를 대표하는 키워드로 선택되기 위해서는, 임의의 키워드가 해당 문서 내에서는 빈번하게 출현하고, 다른 문서에 대해서는 가능한 적게 출현 할수록 키워드로 추출될 확률이 높아진다. 그리하여 문서의 키워드 집합은 임의의 임계치보다 높은 w_{ij}를 만족하는 단어들로 구성된다.

3.3 문서 병합 패턴

TCT에서 문서를 병합하는 패턴은 (문서 vs. 문서 병합), (문서 vs. 클러스터 병합), (클러스터 vs. 클러스터 병합) 세 가지 경우로 구분할 수 있다. 각 병합 패턴에 따라 유사도를 구하고 클러스터로 병합하는 알고리즘을 기술한다.

(1) 문서 vs. 문서 병합
두 문서간에 유사도를 계산하여 임계치 $minS$에

따라 병합 여부를 결정한다.

```
Procedure DocToDoc
 begin
  S(d_i, d_j) 계산;
  if ( S(d_i, d_j) ≥ minS )
  then mergeDTD( d_i, d_j);
  end
```

(2) 문서 vs. 클러스터 병합

문서와 클러스터간의 병합으로서, 문서와 클러스터에 포함된 모든 문서의 유사도의 평균값으로 병합 여부를 결정한다.

```
Procedure DocToCluster
 d_d: 비교 대상 문서;
 d_c: 클러스터 C_a에 포함되며 문서 d_d와 비교 되는 문서;
begin
 클러스터 응집도 C(C_a) 계산:

        C(C_a) = Σ_{k≠l and k, l∈C_a} S(d_k, d_l) / |C_a| ;

 d_d와 d_c의 문서 유사도 계산:

        SA_dc = (S(d_d, d_c) + C(C_a)) / 2 ;

 문서 d_d를 포함한 클러스터 응집도 계산:

        C(C_a ∪ d_d) = Σ_{d_i, d_j∈C_a} S(d_i, d_j) / (|C_a| + 1) ;

 if ( SA_dc ≥ minS and C(C_a ∪ d_d) ≥ minC )
 then mergeDTC( d_d, C_a);
end
```

(3) 클러스터 vs. 클러스터 병합

두개 이상의 유사 문서로 결합된 클러스터간의 비교로서, 다음의 과정을 통해 클러스터간에 유사도를 구하여 병합 여부를 결정한다.

두 클러스터의 평균 유사도와 가장 근사한 각각의 링크 $dp_m - ds_m$; $dp_n - ds_n$을 찾아 D_F를 구성한다. 그러고 나서 두 클러스터의 연결 유사도 SC를 D_F에 포함되는 6개의 문서간의 유사도의 평균으로 하여

```
Procedure ClusterToCluster
 C_n: 비교 대상 클러스터 n;
 C_m: 비교 대상 클러스터 m;
begin
각각의 클러스터 응집도 C(C_n), C(C_m) 계산;
클러스터 C_m에서 클러스터 응집도 C(C_m)과 가장 근사하는 유사도를 갖는 dp_m − ds_m 링크 검색;
클러스터 C_n에서 클러스터 응집도 C(C_n)과 가장 근사하는 유사도를 갖는 dp_n − ds_n 링크 검색;
D_F={ dp_m, ds_m, dp_n, ds_n} 구성;

        SC = (Σ_{d_i∈D_F, d_j∈D_F−{d_i}} S(d_i, d_j)) / 6   계산;

if ( SC ≥ minMer and C(C_n ∪ C_m) ≥ minC )
then mergeCTC( C_n, C_m);
end
```

최소병합도 *minMer* 보다 크고 병합된 클러스터가 최소 응집도를 만족할 때 두 클러스터는 병합된다. *minMer* 이상을 유지하는지 검사하는 이유는 적은 개수의 문서로 구성된 클러스터가 다수의 문서로 구성된 클러스터에 강제적으로 병합되는 것을 막기 위함이다.

3.4 유사도 비교 알고리즘

IDC-TCT는 Initial Comparison, Sibling Comparison, Descendant Comparison, Ancestor Comparison을 수행하면서 유사한 문서끼리 클러스터링 한다. 각 유사도 비교 알고리즘은 BFS(Breath First Search)에 따라 노드를 방문하며 유사도 계산시 3.3절에서 정의한 문서 비교 패턴에 따라 유사도를 계산한다. 이 알고리즘들의 핵심은 이행적 폐쇄 이론에 따라, 각 Comparison을 수행하면서 이전에 비교되지 않은 문서의 페어(Pair)를 찾아 유사 여부를 조사하는 것이다.

(1) Initial Comparison(IC)

이진트리로 구성된 TCT에서 부모-자식간의 링크에 대하여 문서 유사도를 계산하고 그 결과를 자식의 데이터 필드에 저장한다. 이때 유사도가 *minS*보다 큰 값을 갖는 부모 자식간 페어에 대하여 유사 링크를 설정한다.

(2) Sibling Comparison(SC)

트리 레벨별로 유사하지 않는 형제 노드간의 문서 유사도를 계산한다. SC의 핵심은 BFS로 현재 방문하는 노드 d_c와 부모 노드가 유사 하지 않고, d_c의 형제 노드 d_s와 그 부모 노드가 유사하지 않은 경우에만, d_c와 d_d에 대해 문서 비교 패턴에 따라 유사도를 구한다.

Sibling Comparison을 수행하면 각 레벨을 기준으로 클러스터들이 생성되거나, 어떤 클러스터에도 소속되지 않는 기아노드가 남게 된다.

(3) Descendant Comparison(DC)

현재 방문하는 노드에서 후손(하위) 방향으로 유사하지 않는 노드나 클러스터를 찾아 유사도를 검사하기 위한 비교이다. DC 수행의 핵심은, 현재 방문하는 노드 d_c와 그의 부모가 유사하지 않으면, d_c를 기준으로 부모와 유사하지 않는 최소의 후손 노드 d_d를 찾아, d_c와 d_d에 대해 문서 비교 패턴에 따라 유사도를 구하는 것이다.

(4) Ancestor Comparison(AC)

현재 방문하는 노드에서 조상(상위) 방향으로 유사하지 않는 노드나 클러스터를 찾아 유사도를 검사하기 위한 비교이다. AC의 핵심은 현재 방문하는 노드 d_c와 그의 부모가 유사하지 않으면, d_c를 기준으로 조상 노드들 중에 부모와 유사하지 않는 최초의 조상 노드 d_a를 찾아, d_c와 d_a에 대해 문서 비교 패턴에 따라 유사도를 구하는 것이다.

위에서 기술한 유사도 비교 알고리즘에 따라 클러스터를 생성하기위한 IDC-TCT의 전체적인 수행 알고리즘은 다음과 같다.

```
Procedure TCT_Clustering
begin
 initial comparison 수행;
 sibling comparison 수행;
 루트노드의 왼쪽 자손 트리를 L-TCT로 구성;
 루트노드의 오른쪽 자손 트리를 R-TCT로 구성;
 FORALL TCT의 노드 DO
 L-TCT에 대하여 descendant comparison 수행;
 R-TCT에 대하여 descendant comparison 수행;
 ENDFOR
 FORALL TCT의 노드 DO
 ancestor comparison 수행;
 ENDFOR
end
```

3.4 점증적 클러스터링

IDC-TCT는 점증적 클러스터링이 가능한 알고리즘으로서, 문서의 삽입과 삭제에 대하여 문서간 유사도 계산을 중복적으로 하지 않도록 설계한다. 이를 위해 IDC-TCT는 3.3절에서 기술한 방법으로 생성된 각 클러스터로부터 공통 키워드 테이블 T_i를 구성한다. 즉, i번째 클러스터의 공통키워드 테이블을 다음과 같이 정의한다.

정의 3. 공통키워드 테이블 T_i

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

여기에서 m은 클러스터의 총 개수이고, r_i는 i번째 키워드 출현 빈도이고, k_i는 i번째 키워드의 명칭이다.

공통 키워드 테이블의 생성과정은 다음과 같다.

Step 1. 클러스터내의 모든 문서의 키워드들을 하나의 공통 문서로 병합

Step 2. 공통문서에서 키워드명을 오름차순으로 정렬

Step 3. 동일한 키워드명 k_i에 대하여 출현 빈도 r_i를 계산하여, (r_i, k_i) 순서쌍을 구함

Step 4. 공통문서의 모든 키워드에 대하여 Step 3을 반복

(1) 문서 삽입

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

새로운 삽입 문서 d_n에 대하여 점증적 클러스터링을 수행하는 절차는 다음과 같다.

Step 1. 모든 공통 테이블과의 유사도를 다음 수식에 따라 구함

$$S(d_n, T_i) = \frac{|T_i \cap d_n|}{\min(|T_i|, |d_n|)}, \text{ for } i = 1, 2, \cdots, m$$

Step 2. d_n이 소속될 클러스터 i를 다음의 조건에 따라 선정

$\max\{S(d_n, T_i)\}$, for $i = 1, 2, \cdots, m$

단, $S(d_n, T_i) \geq minCarSim$

Step 3. 모든 클러스터 i에 대하여, $S(d_n, T_i) \leq minCarSim$이면, 기아 문서 테이블에 저장

기아 문서 테이블(Starvation document table)은 어느 클러스터에도 소속되지 않는 문서의 집합이다. 그러나 기아 문서 테이블에 일정개수 이상이 수집되면 기아 문서끼리 유사도를 비교하여 클러스터링을 수행한다.

(2) 문서 삭제

임의의 클러스터로부터 문서를 삭제 할 때, 클러스터 응집도의 변화가 발생한다. 따라서 문서 삭제에 따라 *minCoh*를 유지하는지 검사해야 한다. 문서 삭제를 위한 단계를 다음과 같이 정의한다.

Step 1. 삭제 문서 d_r을 제외한 문서 응집도 C_i를 계산한다.

$$C(C_i) = \frac{\sum_{d_i \in C_i - d_r,\ d_j \in C_i - \{d_i\}} S(d_i, d_j)}{|C_i|}$$

Step 2. $C(C_i) \geq minCoh$이면, 문서 d_r삭제

그렇지 않으면, C_i의 모든 문서를 기아 문서 테이블에 저장하고 클러스터 C_i를 제거한다.

기아 문서 집합에 소속된 문서들간의 유사도 처리는 기아 문서 처리율인 *STR*을 선정하여, 전체 문서 수에서 기아 문서의 비율이 *STR*을 초과 하는 경우에 수행한다.

4. 성능 평가 및 분석

본 연구에서 제안한 IDC-TCT의 성능을 평가하기 위하여 정확률, 재현율, F-Measure, 초기 클러스터링 수행시간을 측정한다. 그리고 IDC-TCT의 점증적 클러스터링의 성능을 분석하기 위하여 새로운 문서의 삽입 개수 증가에 따른 수행속도를 측정한다. 기존의 알고리즘과 평가하기 위하여, 수행속도가 우수한 반복적 클러스터링 기법으로는 k-means 알고리즘, 클러스터의 질이 우수한 HAC 기법으로는 AL 기법과 비교한다.

실험 문서는 네이버에서 '소프트웨어' 범주로 분류된 문서 3796개 중에서 선별하여 사용한다. 이 문서는 56개의 범주로 구성되며, 1056개의 문서를 포함한다. 56개의 범주를 하나의 클러스터로 간주하여 실험결과를 분석한다. 문서에서 불용어 처리 및 키워드 추출을 위해 HAM[13]을 이용한다. 이상의 실험 환경은 표 4.1과 같다.

표 4.1 실험 환경

프로그램	IDC-TCT: 자작 프로그램 사용 언어: JDK 1.3	
수행 환경	PC: P-Ⅳ 2.0GHz, 512M RAM	
문서 데이터	네이버 디렉토리 문서 (분류명: '소프트웨어')	1056개 문서, 56개 범주
parameter	*minS = 0.7, minC=0.6,* *minMer=0.4, minCarSim=0.6, k=56*	

IDC-TCT를 수행하여 생성된 클러스터들의 정확률 P를 산출하기 위하여 식 (3.4)를 정의한다.

$$P = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{|OC_j \cap RC_j|}{|\overline{OC_j}|},\ j = 1, 2, \cdots, m \qquad (3.4)$$

where, OC_j: 분류기가 분류한 j 번째 클러스터에 포함된 문서 집합
RC_j: 분류기가 분류한 j번째 클러스터에서 가장 많은 문서를 포함하는 대표 클러스터

식 (3.4)와 같이 정확률은 분류기(IDC-TCT)가 분류한 클러스터의 소속문서와 네이버 범주의 소속문서와 일치하는 개수에 대하여, 분류된 문서와의 비율을 통해 구한다.

또한 재현율 R을 구하기 위해 식 (3.5)를 정의한다.

$$R = \frac{1}{m} \cdot \sum_{j=1}^{m} \frac{|CD_j \cap RC_j|}{|\overline{CD_j}|},\ j = 1, 2, \cdots, m \qquad (3.5)$$

where, CD_j: 네이버 j 범주에 포함된 문서 집합

식·(3.5)와 같이 재현율은 네이버 범주에 소속되는 문서와 분류기가 분류한 클러스터의 소속문서의 일치하는 문서에 대하여 네이버 범주에 소속된 문서의 개수의 비율로 구한다.

따라서 다음 수식을 통해 정확률과 재현율의 성능을 반영하는 F-Measure를 구할 수 있다.

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

네이버의 56개의 범주에 포함되는 1056개의 문서에 대하여, k-means 알고리즘, AL 기법, IDC- TCT를 통해 클러스터링을 수행하여 정확률, 재현율, F-Measure를 측정한 결과는 그림 1과 같다. 그림 1을 통해 IDC-TCT는 k-means 알고리즘에 비하여 약 140%의 정확률이 향상되었고, 재현율은 약 274%가 향상 되어, 결과적으로 약 203%의 F-Measure가 향상되었음을 알 수 있다. 특히 정확률에 비해 재현율이 더욱 향상된 점은 ICT-TCT의 클러스터링 수행 구조가 이행적 폐쇄 이론에 의해 유사도 판별 여부에서 엄격하지 않는 특성 때문이다. 그리고 최근에 k-means 알고리즘에 대하여, centroid 설정을 조절하거나 색인어에 가중치를 부여하는 기법을 적용하여 성능을 개선하는 연구가 이루어지고 있다. 본 실험에서는 2.1절에서 기술한 전통적인 k-means 알고리즘을 적용하여, 정확률 및 재현율에 있어서 다소 성능이 저하되는 특성이 있다.

한편, 초기 클러스터링 수행속도는 IDC-TCT가 AL 기법에 비해, 문서수 2000, 4000, 6000, 8000, 10000개의 문서에 대하여 평균 199% 향상되었음을 그림 2를 통해 알 수 있다. 이것은 클러스터내의 모든

	정확률	재현율	F-Measure
k-means	0.48	0.27	0.35
AL	0.83	0.78	0.8
IDC-TCT	0.67	0.74	0.7

그림 1. 정확률, 재현율, F-Measure 비교 결과

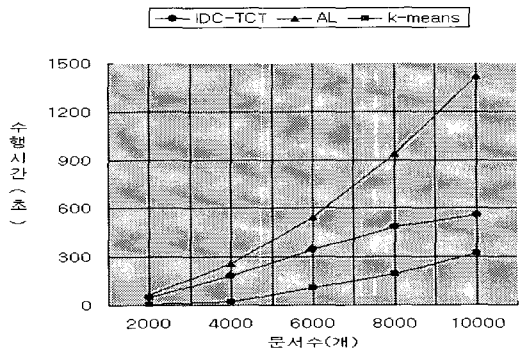

그림 2. 초기 클러스터링 수행속도 비교

문서와의 유사도를 계산하는 AL 기법에 비해 IDC-TCT는 문서간 유사성을 유추함으로써 유사도 비교 회수를 대폭적으로 감소 시켰기 때문이다. 이론적으로, IDC-TCT는 이행적 폐쇄 이론에 따라 초기 문서 비교 회수가 $nlogn$번이 필요하고, 이후에 클러스터의 개수에 따라 급격하게 줄어드는데 반하여, AL 기법에서는 최소한 n^2logn번의 비교회수가 필요하다.

끝으로, IDC-TCT의 점증성을 측정하기 위하여, 6000개의 문서를 기점으로 새로운 문서의 삽입을 2000개씩 추가하여 k-means 알고리즘과 IDC-TCT와의 수행속도를 비교한다. 그림 3에서 알 수 있듯이 IDC-TCT의 점증적 클러스터링의 수행속도는 오히려 k-means 알고리즘 보다 더 빠르다. 이것은 k-means 알고리즘의 수행속도는 O(문서수*k*반복수)이면서, 새로운 문서의 삽입 때마다 다시 클러스터링을 수행해야 하지만, IDC-TCT는 이전의 클러스터링 정보를 이용함으로써 각 클러스터의 공통키워드 테이블과의 유사도를 계산하는 시간만 거의 필요하기 때문이다.

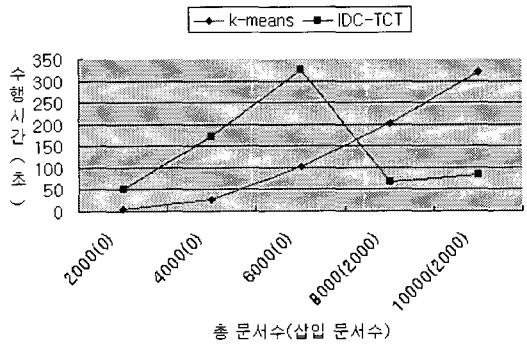

그림 3. 점증적 수행속도 비교

5. 결 론

본 논문에서는 기존의 k-means 알고리즘에 비해 정확률 및 재현율을 향상시키고, 계층적 집괴 클러스터링 알고리즘의 기법 중에 하나인 AL 기법에 비해 수행속도를 향상시킨 새로운 문서 클러스터링 알고리즘인 IDC-TCT를 제안하였다. IDC-TCT는 이행적 폐쇄 이론을 근간으로 하여 문서 유사도 비교 회수를 대폭적으로 감소시키고 그에 따른 정확률 및 재현율의 희생을 감소시키기 위한 메카니즘을 고려한 알고리즘이다. 그리고 IDC-TCT는 문서 클러스터를 생성하고 나서 공통 키워드 테이블을 유지함으로써, 이후의 문서의 삽입과 삭제에 대하여 문서 유사도를 재계산하지 않는 점증적인 클러스터링이 가능한 기법이다.

제안한 알고리즘의 성능평가를 통해 k-means 알고리즘에 비해 정확률과 재현율을 향상 시켰으며, AL 기법에 비해 초기 클러스터링 수행 속도가 향상되었음을 알 수 있다. 그리고 점증적 클러스터링의 성능 측정을 통해, k-means 알고리즘에 비해 새로운 문서의 삽입에 대하여 신속한 클러스터링이 가능함을 알 수 있었다.

IDC-TCT는 웹 문서 분류뿐만 아니라, 보편적인 분류의 질(quality)을 유지하면서, 신속하게 유사문서끼리 클러스터링하기위한 e-mail 분류, 기사 분류, 일반 문서 분류에 활용이 가능하다.

향후 연구로는 IDC-TCT에서 정확률을 세밀하게 조절 할 수 있는 알고리즘의 개발과 점증성을 극대화 할 수 있는 구(Phrase) 단위 기반 클러스터링 알고리즘에 대한 연구가 필요하다.

참 고 문 헌

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

[2] M. Steinbach, G. Karypis, and V. Kumar, "A Comparison of Document Clustering Technique," *Proc. of the KDD-2000 Workshop TextMining*, pp. 146-149, Aug. 2000.

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

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

[5] T. Kanung, "The analysis of a Simple k-Means Clustering Algorithm," *Proc. of ACM Symposium on Computational Geometry*, pp. 12-14, Jun. 2000.

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

[7] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental Clustering and Dynamic Information Retrieval," *Proc. 29th Ann. ACM Symp. Theory of Computing*, pp. 626-635, 1997.

[8] Khaled M. Hammouda and Mohamed S. Kamel, "Efficient phrase-based document Indexing for web document clustering," *IEEE tran. on knowledge and data engineering*, Vol. 16, No. 10, pp. 1279-1296, Oct. 2004.

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

[10] 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.

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

[12] 강동혁, 주길홍, 이원석, "대용량 문서 데이터베이스를 위한 효율적인 점진적 문서 클러스터링 기법," 정보처리학회논문지 D, 제 10-D권, 제1호, pp. 57-66, 2003. 02.

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

고 석 범

1996년 동서대 컴퓨터공학 학사
1998년 아시카가 공대 정보시스템공학 석사
2001년~2004년 육군사관학교 전자계산학과 전임강사
현재 부경대 전자계산학 박사과정

관심분야 : 분산데이터베이스, 텍스트마이닝, 문서 클러스터링, XML

윤 성 대

1980년 경북대 컴퓨터공학 학사
1984년 영남대 전자계산학 석사
1997년 부산대 전자계산학 박사
1986년~현재 부경대 전자컴퓨터정보통신공학부 교수

관심분야 : 병렬운영체제, 데이터마이닝, Multi-threaded architecture