빈발단어집합을 이용한 NaiveBayes의 정확도 개선☆

An Improvement of Accuracy for NaiveBayes by Using Large Word Sets

이 재 문*
Jae-Moon Lee

요 약

본 논문은 연관규칙탐사 기술에서 사용되는 빈발항목집합을 변형하여 문서분류의 문서에서 빈발단어집합을 정의하고, 이를 사용하여 문서분류 방법으로 잘 알려진 NaiveBayes에 적용하여 이 방법의 정확도를 개선한다. 이 기술의 적용을 위하여 하나의 문서는 여러 개의 문단으로 나뉘어졌으며, 각 문단에 나타나는 단어들의 집합을 트랜잭션화하여 빈발단어집합을 찾을 수 있도록 하였다. 제안한 방법은 AI::Categorizer 프레임워크에서 구현되었으며 로이터-21578 데이터를 사용하여 그 정확도가 측정되었다. 문단에서의 라인수와 학습문서의 크기를 변화하면서 정확도를 측정하였다. 측정된 결과로부터 제안된 방법이 기존의 방법에 비하여 정확도를 개선한다는 사실을 알 수 있었다.

Abstract

In this paper, we define the large word sets which are noble variations the large item sets in mining association rules, and improve the accuracy for NaiveBayes based on the defined large word sets. In order to use them, a document is divided into the several paragraphs, and then each paragraph can be transformed as the transaction by extracting words in it. The proposed method was implemented by using AI:Categorizer framework and its accuracies were measured by the experiments for reuter-21578 data set. The results of the experiments show that the proposed method improves the accuracy of the conventional NaiveBayes.

☞ Keyword : Accuracy, Machine Learning, NaiveBayes, Document Classification, Large Item Sets

1. 서 론

최근 웹 문서 등 전자 문서의 급증으로 이들을 관리하는 정보관리시스템 분야에서 기계 학습에 의한 문서 분류 연구가 활발히 진행되고 있다[1-9]. 문서 분류란 미리 정해진 분류의 집합이 있을 때 특정 문서가 어느 분류에 속하는지를 판단하는 것을 말한다. 문서 분류의 초기 연구에서는 규칙에 기초하여 문서를 분류 하였으나 최근에는 컴퓨팅 기술의 발전으로 기계학습 방법에 의한 연구가 활발히 진행되고 있다[1,2,3].

문서 분류에 대한 연구의 방향은 크게 두 가지로 나뉜다. 하나는 분류의 정확도를 높이는 기술에 관한 연구이고[1,3,4,9], 다른 하나는 문서 분류의 속도를 높이는 기술에 관한 연구이다[1,2,5,8]. 문서 분류에 대한 대부분의 연구는 전자에 집중되어 왔다. NaiveBayes, SVM, kNN 등 여러 기법들이 연구되어 왔고, 각 기법은 주어진 환경에 따라 다른 성능을 나타낸다. Naive Bayes [1,2] 방법은 단순하고, 빠른 응답을 주는 반면, 분류의 정확도가 다른 기법에 비해 떨어지는 편이다. SVM[1]은 상당히 높은 정확도를 주는 기법이나 알고리즘이 복잡하고 속도가 느리다. kNN [1,5]은 가장 간단한 기법 중의 하나이면서 비교적 높은 분류 정확도를 보이지만, 실행 속도가 매우 느리다.

기계학습에 의한 문서 분류는 학습(훈련)단계와 분류(시험)단계로 나뉜다. 학습단계는 미리

* 정 회 원 : 한성대학교 멀티미디어공학과 교수
jmlee@hansung.ac.kr
[2006/01/16 투고 - 2005/01/19 심사 - 2006/02/24 심사완료]
☆ 본 연구는 2005년 한성대학교 교내 연구비 지원 과제임

정해진 분류 집합과 각 분류별로 전문가에 의하여 정확히 분류된 학습 문서 집합을 입력으로 받아 학습하는 과정이다. 이 단계의 결과는 방법별로 차이는 있으나 궁극적으로 분류단계에서 최적의 성능을 얻을 수 있도록 입력 데이터인 학습 문서를 가공하는 것이다. 분류단계는 이러한 가공된 데이터와 새로운 문서를 입력 받아, 입력된 문서가 어느 분류에 속하는지를 판단하는 단계이다. 본 논문에서는 학습단계와 분류단계에서 문서에 나타나는 빈발단어집합을 이용하여 NaiveBayes의 정확도를 개선하는 알고리즘을 제시한다.

지금까지 연구된 대부분의 NaiveBayes에서는 문서에 나타나는 개별적인 단어에 대한 가중치만 고려하여 문서 분류를 하였다. 본 논문에서는 단어간에 나타나는 정보를 찾고 이 정보에 가중치를 적용하여 NaiveBayes의 분류 정확도를 높이고자 하는 것이다. 이를 위하여 연관규칙탐사 등 데이터마이닝에서 사용하는 빈발항목집합(Large Item Sets 또는 Frequent Item Set)[10,11,12,13]을 이용하여 단어간 가중치를 계산한다. 즉, 연관규칙탐사에서 사용하는 빈발항목집합과 같은 개념으로 문서에 나타나는 빈발단어집합을 이용하는 것이다. 이를 위하여 문서에서의 빈발단어집합을 정의하고, 연관규칙탐사에서의 빈발항목집합을 찾는 방법을 이용하여 빈발단어집합을 찾는 방법을 제안하였다. 제안된 알고리즘은 잘 알려진 문서분류 프레임워크상에서 구현되었으며, 로이터-21578 데이터를 사용하여 실험되었다. 실험을 통하여 제안된 알고리즘이 기존의 NaiveBayes보다 정확도를 개선한다는 사실을 알 수 있었다.

2장에서는 NaiveBayes에 대한 전반적인 소개와 알고리즘을 소개하며 또한 빈발단어집합을 소개한다. 3장에서는 문서에서의 빈발단어집합을 정의하고, 빈발단어집합을 찾는 알고리즘을 소개한다. 4장에서는 기존의 NaiveBayes와 제안된 Naive Bayes사이의 성능 비교를 하며, 5장에서 결론을 논한다.

2. 기존 연구

2.1 NaiveBayes

NaiveBayes는 확률에 근거한 문서 분류 방법이다. 이는 임의의 문서에 대하여 분류별 속할 확률을 계산하여 계산된 확률 중 가장 높은 확률을 가지는 분류를 선택하는 것을 말한다. 이때 확률은 다음 식에 의하여 계산된다.

$$P(c_j|d_i) = \frac{P(c_j)P(d_i|c_j)}{P(d_i)} \quad (1)$$

상기식의 의미는 문서 집합 $d_1,d_2,...,d_{|D|}$중에서 문서 d_i가 분류 집합 $c_1,c_2,...,c_{|C|}$중에서 c_j에 속할 확률을 의미한다. 여기서 $P(d_i)$는 문서 집합에서 임의로 추출한 문서가 d_i일 확률이고, $P(c_j)$는 문서 집합에서 임의로 추출한 문서가 분류 c_j에 속할 확률이다. $P(d_i|c_j)$는 분류 c_j에 속하는 문서 집합에서 임의로 추출한 문서가 d_i가 될 확률이다. 따라서 NaiveBayes는 주어진 문서 d_i에 대하여 $P(d_i|c_1)$, $P(d_i|c_2)$, ..., $P(d_i|c_{|C|})$의 확률을 구하여 가장 높은 확률을 가지는 분류를 선택하는 것이다. 이것을 일반적으로 다음과 같이 표시한다.

$$C_{best} = ArgMax_{c_j \in C} \{ \frac{P(c_j)P(d_i|c_j)}{P(d_i)} \} \quad (2)$$

C_{best}를 구하는 과정에서 $P(d_i)$는 모든 c_j에 대하여 항상 일정한 값을 가지므로 생략할 수 있다. 그러나 $P(d_i|c_j)$는 단어간 의존성을 고려하여야 하기 때문에 쉬운 일이 아니다. 따라서 대부분의 경우 계산을 효율적으로 하기 위하여 하나의 문서에서 단어간 의존성은 없다고 가정하여 계산한다[1,2]. 이러한 가정 하에 문서 d_i가 (t_{i1}, t_{i2}, ..., t_{im})으로 표현된다고 하자. 여기서 t_{ik}는 문서 d_i가 포함하고 있는 용어이고 m은 포함하고 있는 용어의 개수이다. 이 경우 $P(d_i|c_j)$는 $\prod_{k=1}^{m} P(t_{ik}|c_j)$로 표현

되고, $\prod_{k=1}^{m} P(t_{ik}|c_j)$에서 $P(t_{ik}|c_j)$가 0인 경우를 피하기 위하여 라플라스 스무더(Laplace Smoother)를 적용하여 이를 $\frac{1+TF(t_k,c_j)}{|T|+\sum_{s=1}^{M_c} TF(t_s,c_j)}$ 로 표현한다[1,2].

여기서 $TF(t,x)$는 x에서 용어 t의 가중치를 의미한다. 또한 $\sum_{s=1}^{M_c} TF(t_s,c_j)$ 는 분류 c_j에 나타나는 모든 용어에 대한 가중치의 합이고 $|T|$는 학습 문서 집합에 나타나는 전체 용어의 집합의 크기이다. $P(t_{ik}|c_j)$가 항상 1보다 작은 값을 가지는 확률이므로 $\prod_{k=1}^{m} P(t_{ik}|c_j)$는 너무 작은 값을 가진다. 이를 피하기 위하여 식 (2)에 *log*를 적용하여 계산한다. 또한 시험 문서 d에 나타나는 용어의 가중치를 적용하기 위하여 식 (3)과 같이 $TF(t_k, d)$를 적용한다[1, 2].

$$C_{best} = ArgMax_{c_j \in C} \{ \log[P(c_j)] + \sum_{k=1}^{m} \log [\frac{1+TF(t_k,c_j)}{|T|+\sum_{s=1}^{M_c} TF(t_s,c_j)}] \times f(TF(t_k,d))] \} \quad (3)$$

상기 식에서 $f(x)$는 x의 값을 변환하는 함수로써 대부분의 경우 $f(TF(t_k,d))=1$ 또는 $f(TF(t_k,d))=TF(t_k,d)$를 사용한다. 전자의 경우 시험 문서 d에 나타나는 용어의 가중치를 동일한 값으로 반영한 경우이며[1,2], 후자의 경우 가중치에 대한 특별한 변환 없이 반영하는 경우이다. 본 본문에서는 후자의 경우라 가정한다.

2.2 빈발항목집합

빈발항목집합(Large Itemset 또는 Frequent Itemset)은 연관 규칙 탐사 문제에서 정의된 것이다[10-13]. $I=i_1,i_2,...,i_m$를 m개의 중복이 없는 항목의 집합이라 하자. 트랜잭션 T는 고유한 식별자를 가지고 있으며 I의 부분집합인 항목들의 집합이다. 데이터베이스 D는 이러한 트랜잭션 T의 집합이라고 하고, $|D|$를 D에 포함된 트랜잭션들의 개수라 할 때 $X \subseteq I$인 X가 $\sup(X) \geq |D| \times s$를 만족하면 항목집합 X를 ***빈발항목집합*** 이라 말한다[10]. 여기서 $\sup(X)$는 D의 트랜잭션 중 X를 포함하는 트랜잭션들의 개수를 의미하고, s는 사용자가 지정하는 최소지지도이다.

〈표 1〉 예제 데이터베이스

트랜잭션	T_1	T_2	T_3	T_4	T_5	T_6
구성항목	A:C	B:C:E	A:B:C:E	B:E	A:C:E	B:C:D:E

예를 들어 <표 1>의 예제 데이터베이스를 고려하자. 여기서 T_i는 트랜잭션을 의미하고, $i_1:i_2:...:i_k$의 의미는 트랜잭션 T_i가 $i_1:i_2:...:i_k$의 항목으로 구성되었다는 것이다. 여기서 사용자에 의하여 주어진 최소 지지도(s)가 50%이라고 하자. 즉 $|D| \times s$가 3이라고 하자. 이때 <표 1>의 데이터베이스에 포함된 빈발항목집합은 {*A*}, {*B*}, {*C*}, {*E*}, {*A:C*}, {*B:C*}, {*B:E*}, {*C:E*}, {*B:C:E*}이다. 여기서 {*A*}, {*B*}, {*C*}, {*E*}를 1-크기 빈발단어집합이라 하고, {*A:C*}, {*B:C*}, {*B:E*}, {*C:E*}는 2-크기, {*B:C:E*}는 3-크기 빈발단어집합이라 한다.

3. NaiveBayes에 빈발단어집합의 적용

앞장의 예제 데이터베이스에서 *A, B, C, D, E*는 문서에 포함된 단어 종류이고, T_1~T_6은 문서에 나타난 문단이라고 하고, <표 1>은 각 문단에 나타나는 단어들의 집합이라 가정하자. 즉, 첫 번째 문단이라 할 수 있는 T_1은 단어 *A*, *C*를 포함하고 있고, 두 번째 문단이라 할 수 있는 T_2는 *B*, *C*, *E*를 포함하고 있다고 가정하자. 이러한 가정을 할 때 {*A*}, {*B*}, {*C*}, {*E*}, {*A:C*}, {*B:C*}, {*B:E*}, {*C:E*}, {*B:C:E*}를 예제

데이터베이스에서 빈발단어집합이라 할 수 있다. 이 경우 $\{A\}$, $\{B\}$, $\{C\}$, $\{E\}$는 기존의 NaiveBayes에서 사용되는 단어가 되고 이들의 $\sup(X)$가 식 (3)에서의 $TF(X,d)$로 사용될 것이다. 이러한 관점에서 보면 기존의 NaiveBayes 방법은 단순히 1-크기의 빈발단어집합만 사용하여 문서 분류를 하였다고 할 수 있다. 본 논문의 핵심 아이디어는 1-크기 빈발단어집합뿐만 아니라, k-크기의 빈발단어집합도 NaiveBayes에 적용함으로써 Naive Bayes 방법의 정확도를 높이고자 하는 것이다.

3.1 문서에서의 트랜잭션 정의

문서에 나타나는 빈발단어집합을 정의하고 탐사하기 위하여 단어집합(I), 트랜잭션(T), 데이터베이스 (D)를 정의하여야 한다. 문서는 여러 개의 문단으로 구성되어 있고, 하나의 문단은 한 개 이상의 문장으로 구성되어 있다. 또한 하나의 문장은 한개 이상의 단어로 구성되어 있다. 문서에서의 문단, 문장, 단어의 계층적 관계는 문서 작성에 대한 기본적인 규칙이다. 물리적인 의미로써 문단, 문장, 단어들은 각각 독립된 트랜잭션으로 고려될 수 있다. 그러나 대부분의 문서에서는 문단 단위로 독립된 의미를 가지고 있다. 이는 역으로 하나의 문장은 독립된 의미를 가지기 힘들다는 것을 의미한다. 즉, 여러 개의 문장들이 모였을 때 특정 의미를 나타내는 문단을 이룬다고 할 수 있다. 이러한 관점에서 문단이 트랜잭션으로 고려되는 것이 가장 적합하다고 할 수 있다. 본 논문에서도 하나의 문단을 하나의 트랜잭션으로 고려한다. 이 경우 하나의 문단에 속하는 모든 단어들은 그 트랜잭션에 속하는 단어들이 된다. 또한 단어집합(I)는 모든 문서에 나타나는 단어들의 전체 집합이 되며, 데이터베이스는 문단으로 구성된 문서가 된다.

<그림 1>의 예제문서는 2005년 8월 29일자 조선일보의 인터넷판 기사 중의 하나이다. 이 문서는 4개의 문단으로 구성되어 있다. 따라서 앞의 정의에 따라 4개의 트랜잭션으로 구성된다고 할 수 있고, 각 트랜잭션에 포함된 단어들은 고유 명사와 동사들을 제외하는 특정 규칙을 적용하였다고 할 때 <표 2>와 같이 찾을 수 있다. <표 2>에서 가중치는 단어의 발생수로 하였다.

'현대차에 이어 기아차 노조의 파업으로 국내 자동차 생산에 큰 차질이 빚어질 전망이다. 이에 따라 소비자들은 현대차의 그랜저 및 쏘나타를 주문하고 최소 한 달 이상을 기다려야 하고, 현재 주문이 밀려 있는 기아차의 스포티지, 프라이드, 그랜드 카니발도 인도시까지 대기 시간이 훨씬 더 길어지게 됐다.

기아자동차는 28일 "노조가 예정대로 29일부터 5일간 48시간의 파업을 강행하고 휴일 특근과 잔업을 전면 거부할 경우 6,730대의 생산 차질이 발생, 1,000억여원의 매출 손실이 예상된다"고 밝혔다. 기아차 노조가 파업에 들어가면 지난 91년부터 15년 연속 파업이다.

기아차는 현재 5만4,000대의 수출 주문이 밀려 있는데다, 내수에서도 스포티지 5,000대, 프라이드 2,000대, 그랜드 카니발 1500대의 주문이 적체된 상태여서 노조가 파업을 강행할 경우 내수와 수출에 큰 타격이 예상된다.

현대차도 지난 25일부터 이틀간 파업으로 8,403대의 생산 차질에 매출 손실 1,207억원의 피해를 입고 있다.'
2005. 8. 29. 조선일보

〈그림 1〉 예제문서

〈표 2〉 예제 문서에 대한 데이터베이스

트랜잭션	<단어, 가중치> 목록
T_1	<국내,1><노조,1><대기,1><생산,1> <소비자,1><시간,1> <인도,1><전망,1><주문,2><차,5><차질,1> <파업,1>
T_2	<강행,1><거부,1><노조,1><매출,1> <생산,1><손실,1> <잔업,1><차,2><차질,1><특근,1><파업,1> <휴일,1>
T_3	<강행,1><내수,1><노조,1><수출,2><적체,1> <주문,2><차,1><파업,1>
T_4	<매출,1><생산,1><손실,1><차,1><차질,1> <파업,1><피해,1>

3.2 빈발단어집합을 찾는 알고리즘

2장에서 설명한 NaiveBayes 알고리즘의 식 (3)에서 가장 기본적인 데이터는 $TF(t_k, c_j)$, $TF(t_k, d)$이다. 전자는 학습단계에서 계산되어지는 것이고, 후자는 분류단계에서 계산되어지는 것이다. 전자가 계산되기 위해서는 학습단계에 사용되는 각각의 학습문서에 대해서 $TF(t_k, d)$가 구해져야 한다. 따라서 NaiveBayes 알고리즘에서는 학습/분류단계에 관계없이 주어지는 문서에 대하여 $TF(t_k, d)$를 구하여야 한다. $TF(t_k, d)$는 문서별 <*단어, 가중치*>로 표현한다. 따라서 본 논문에서는 NaiveBayes 알고리즘과 별개로 임의의 문서에 대하여 단순히 <*빈발단어집합, 가중치*> 목록을 찾는 방법만 제시하기로 한다. 단, 가중치는 빈발단어집합의 발생수로 한다.

빈발단어집합을 찾는 방법으로 [12]에서 제시한 *Partition* 방법을 사용한다. 이 방법은 임의의 후보 단어집합의 서브 빈발 단어집합의 트랜잭션 식별자(TID) 목록들을 교집합하여 교집합 목록의 TID 수에 따라 빈발 단어집합 여부를 판별하는 것이다. 이 방법은 비교적 구현이 쉽고, 계산 속도도 매우 빠르나 데이터베이스의 모든 내용이 메모리내에 상주하여야 한다[12]. 대부분의 연관규칙탐사에서는 대용량 데이터베이스를 가정하기 때문에 이방법의 사용에 상당한 제한이 있으나, 본 논문과 같이 하나의 문서가 하나의 데이터베이스로 고려되는 경우 매우 작은 데이터베이스이기 때문에 적합한 알고리즘이라 할 수 있다. *Find_LargeWordSet*은 [12]에서 제시한 *Partition* 방법을 변형한 빈발단어집합을 찾는 알고리즘이다. 알고리즘의 입력은 문서분류에서 사용되는 문서와 최소지지도이고, 출력은 빈발단어집합이다. 빈발단어집합은 그들의 발생수와 TID 목록을 포함하고 있기 때문에 이들로부터 최종적인 <*빈발단어집합, 가중치*>를 계산할 수 있다.

Find_LargeWordSet(입력 : 문서 d , 최소지지도 s 출력 : 빈발단어집합 L)

스텝 01: 하나의 문서를 분석하여 문단별(TID) <단어, 발생수>의 목록을 만든다.

스텝 02: 문단별 <단어, 발생수>를 단어별 <TID, 발생수>의 목록으로 변환하고, 이들 중 목록의 요소수가 $s*|T|$보다 큰 경우 집합 L_1에 포함한다.

스텝 03: for(k=2; $|L_{k-1}|$!= 0; k++){

스텝 04: forall $l_1 \in L_{k-1}$ {

스텝 05: L_x={ c| $c \in L_{k-1}$, c[1]= l_1[1]^...^ c[k-2]= l_1[k-2]^ c[k-1] ≠ l_1[k-1]}

스텝 06: forall l_2 in L_x {

스텝 07: c= l_1[1] l_1[2]... l_1[k-1] l_2[k-1]로 구성된 c를 생성한다.

스텝 08: c의 k-1 서브 단어집합들이 L_{k-1}에 모두 없으면 스텝06을 수행한다.

스텝 09: l_1의 TID목록과 l_2의 TID목록을 교집합하여 c의 TID목록을 생성한다.

스텝 10: c의 TID목록의 개수가 $s*|T|$보다 크면 L_k= $L_k \cup$ { c}를 한다.

스텝 11: }

스텝 12: }

스텝 13: }

스텝 14: L={ L_1, L_2, ..., L_k}을 출력한다.

스텝 01에서 하나의 문서를 스캔하여 문단별 <단어, 발생수>를 만든다는 것은 문서를 여러 개의 논리적 문단으로 나뉜 후, 각 문단에 발생하는 단어와 그들의 발생수를 찾는 것이다. 스텝 02는 *Partition* 방법에서 요구하는 자료구조로 문단별 <단어, 발생수> 목록을 단어별 <문단(TID), 가중치>로 변환하는 것으로 단순히 행렬 전치로 구할 수 있다. 스텝 02, 스텝 10에서 $|T|$는 트랜잭션의 수를 의미한다. 예를 들어 <표 2>의 데이터베이스에서 최소지지도(s) 75%을 만족하는 단어에 대하여 이를 단어별 <TID, 가중치>로 변환하면 <표 3>을 얻을 수 있다.

〈표 3〉 단어별 〈TID, 가중치〉 목록

단어	<TID, 가중치> 목록
노조	<1, 1><2, 1><3, 1>
생산	<1, 1><2, 1><4, 1>
차	<1, 5><2, 2><3, 1><4, 1>
차질	<1, 1><2, 1><4, 1>
파업	<1, 1><2, 1><3, 1><4, 1>

<표 3>에서 '차'의 <TID, 가중치> 목록인 '<1, 5><2, 2><3, 1><4, 1>'의 의미는 '차'라는 단어는 1, 2, 3, 4 문단에서 발생하였으며, 각각은 5, 2, 1, 1번씩 나타났다는 것을 의미하고 있다. $s*|T|$가 3이고, '노조', '생산', '차', '차질', '파업'는 각각 3, 3, 4, 3, 4번 다른 문단에서 나타나기 때문에 L_1={'노조', '생산', '차', '차질', '파업'}이 된다. 스텝 05에서 $c[i]$, $l_1[i]$는 각각 c, l_1의 단어집합 중 i번째 단어를 의미한다. (표 3)의 목록을 이용하여 '노조:차'의 TID 목록을 얻기 위해서는 '노조', '차'의 TID 목록인 {<1, 1><2, 1><3, 1>}와 {<1, 5><2, 2><3, 1><4, 1>}을 TID 관점에서 교집합하여 {<1, 6><2, 2><3, 2>}이라는 '노조:차'의 TID 목록을 얻게 된다. 이 경우 {<1, 6><2, 2><3, 2>}는 $s*|T|$를 만족하므로 L_2에 속하게 된다. 여기서 가중치는 합하는 것으로 하였다. 반면, '노조:생산'의 TID 목록을 같은 방법으로 계산하면 {<1, 2><2, 2>}를 얻게 되고 이는 $s*|T|$를 만족하지 못하므로 L_2에 속하지 못하게 된다. (표 4)는 (표 3)에 대하여 *Find_LargeWordSet*을 이용하여 구해진 단어집합별 <TID, 가중치> 목록이다.

〈표 4〉 단어집합별 〈TID, 가중치〉 목록

L_x	단어집합	<TID, 가중치> 목록
L_2	노조:차	<1, 6><2, 2><3, 2>
	노조:파업	<1, 2><2, 2><3, 2>
	생산:차	<1, 6><2, 3><4, 2>
	생산:차질	<1, 2><2, 2><4, 2>
	생산:파업	<1, 2><2, 2><4, 2>
L_3	노조:차:파업	<1,8><2,4><3,4>
	생산:차:차질	<1,8><2, 5><4, 4>
	생산:차:파업	<1, 8><2, 5><4, 4>
	생산:차질:파업	<1, 2><2, 4><4, 4>
L_4	생산:차:차질:파업	<1, 16><2, 10><4, 8>

4. 실험 및 성능비교

4.1 실험 환경

제안된 알고리즘과 기존의 알고리즘을 문서분류의 실험 데이터로 잘 알려진 로이터-21578 데이터를 사용하여 실험을 통하여 성능을 비교 한다[3,6]. 이를 위하여 제안된 알고리즘을 [2]에서 개발한 AI::Categorizer 프레임워크를 사용하여 구현하였다. AI::Categorizer는 객체 지향 문서분류 프레임워크로 문서분류에 필요한 다양한 기능 및 일반적으로 잘 알려진 문서분류 알고리즘의 구현하여 제공한다. 이 프레임워크를 사용함으로써 문서의 토큰화, 벡터 모델, 차원 축소 등 부수적인 구현을 생략할 수 있었으며, 실험을 위해서 AI::Categorizer::Document 객체로부터 상속된 AI::Categorizer::Document::MiningAssociationRules만 구현 하였다. 알고리즘의 구현 언어로는 AI::Categorizer에서 사용한 Perl을 사용하였으며, 실험은 펜티엄III 512MB의 리눅스 시스템 상에서 실행하였다. 실험에 사용한 데이터는 로이터-21578 ApteMod 버전[6]이다. 로이터-21578 데이터는 총 10,788개의 문서로 구성되어 있다. 시험문서는 전체 10,788 문서로부터 임의적으로 선택한 788개의 문서로 고정하였으며, 학습문서 집합은 10가지 다른 문서집합을 시험문서를 제외한 10,000 문서로부터 임의적으로 선택하여 구성하였다. 학습문서집합은 DXK로 표시되는데 이것의 의미는 이 학습문서 10,000 문서 중 X 천개의 문서가 임의적으로 선택되어 학습문서로

사용되었다는 것이다. 즉, D2K는 학습문서가 2,000개라는 것을 의미한다.

4.2 성능 측정 요소

문서 분류에서 정확도의 측정은 각 시험 문서에 대하여 가능한 결과를 (표 5)와 같이 a, b, c, d를 측정하여 리콜(Recall), 정밀도(Precision)를 계산한다.

〈표 5〉 정확도 측정 요소

		전문가	
		YES	NO
분류기	YES	a	b
	NO	c	d

예를 들어 임의의 문서 d에 대하여 전문가는 $\{C_1, C_2, C_3\}$으로 분류 하였고, NaiveBayes와 같은 분류기는 $\{C_2, C_3, C_4\}$으로 분류 하였다고 하자. 이때 C_1에 대해서는 전문가는 YES이고 분류기 NO이므로 c의 값이 1 증가한다. C_2, C_3에 대해서는 전문가와 분류기가 모두 YES이므로 a가 2 증가하고, C_4에 대해서는 같은 방법으로 b를 1 증가 시킨다. 모든 시험 문서에 대하여 <표 5>와 같이 a, b, c, d 요소를 측정하여 정밀도 P와 리콜 R은 각각 식 (4), (5)와 같이 계산한다[1,8].

$$P = \frac{a}{a+b} \tag{4}$$

$$R = \frac{a}{a+c} \tag{5}$$

식 (4), (5)에서 각각 a+b = 0, a+c = 0인 경우에는 P=0, R=0이 된다. 대부분의 논문에서 정밀도와 리콜외 종종 F_1을 사용하는데 이것은 식 (6)과 같이 표현된다[1,8].

$$F_1 = \frac{2PR}{P+R} \tag{6}$$

매크로는 분류집합의 각 분류별 리콜과 정밀도를 측정한 후 이들의 평균을 구하는 것이고, 마이크로는 전체에 대하여 리콜과 정밀도에 대한 평균을 구하는 것이다[1,8]. 본 논문에서 마이크로 리콜, 마이크로 정밀도, 마이크로 F_1, 매크로 리콜, 매크로 정밀도, 매크로 F_1을 각각 miR, miP, miF1, maR, maP, maF1로 표기한다.

4.3 성능 측정

XML로 구성된 RCV1[4]과 달리 로이터-21578은 단순한 텍스트 문서이다. 따라서 문서에 대한 특별한 사전 지식 없이 문단을 구별해 낸다는 것은 불가능한 일이다. 본 실험에서는 이러한 문제를 해결하기 위하여 문서에서 단순히 몇 라인씩 문장들을 그룹화함으로써 문단을 구성하도록 하였다. 실험은 먼저 하나의 문서를 문단 단위로 나눌 때 문단당 적절한 라인수를 찾는 실험을 실행하고, 이 실험을 통하여 선택된 문단당 라인수를 사용하여 학습문서의 구성에 따라 Naive Bayes의 정확도를 측정하였다.

문단당 적절한 라인 수를 결정하기 위하여 동일한 학습문서와 시험문서를 사용하여 <표 6>과 같이 라인수를 변경하면서 실험을 하였다. 학습문서로 D10K를 사용하였고, 시험문서로 앞에서 언급한 788개의 문서를 사용하였다.

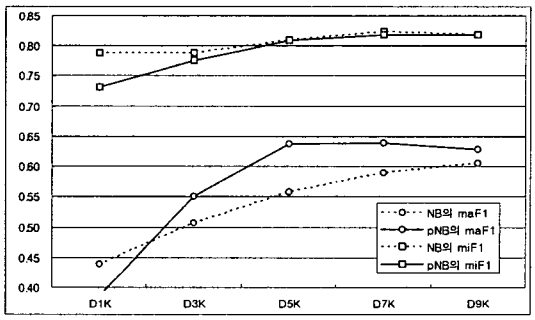
〈표 6〉 문단당 라인수별 측정된 정확도

라인수	maR	maP	maF1	miR	miP	miF1
1	0.6269	0.7296	0.6409	0.7336	0.9110	0.8127
2	0.6301	0.7207	0.6418	0.7427	0.9281	0.8251
4	0.6207	0.7120	0.6291	0.7205	0.8947	0.7982
8	0.5972	0.6965	0.6050	0.7175	0.8899	0.7944

〈표 7〉 학습문서별 측정된 정확도

학습문서	maR		maP		miR		miP	
	NB	*pNB*	*NB*	*pNB*	*NB*	*pNB*	*NB*	*pNB*
D1K	0.4048	0.4299	0.5671	0.4765	0.7227	0.6656	0.8667	0.8132
D3K	0.4763	0.5401	0.6346	0.6558	0.7255	0.7053	0.8631	0.8598
D5K	0.5229	0.6094	0.6716	0.7275	0.7366	0.7316	0.8979	0.9040
D7K	0.5583	0.6298	0.6693	0.7269	0.7487	0.7346	0.9138	0.9215
D9K	0.5716	0.6215	0.6858	0.7077	0.7497	0.7356	0.9017	0.9205

실험은 <표 6>에서와 같이 문단당 라인수를 1~8까지 변경하면서 NaiveBayes의 정확도를 측정하는 것이다. 제안된 방법에 대하여 1-8까지 라인수를 변경하면서 측정하였다. <표 6>로부터 라인수의 변경에 따라 정확도는 변화하고 있음을 알 수 있다. 문단당 라인수가 많다는 것은 문서당 트랜잭션수가 적다는 것을 의미하고, 따라서 빈발단어집합이 많지 않다는 것을 의미한다. <표 6>으로부터 문단당 라인수가 2일 때 가장 좋은 성능을 보이는 것을 알 수 있다.



〈그림 2〉 학습문서의 크기(DnK)에 따른 F_1

다음 실험은 <표 6>에서 선택한 문단당 라인수로써 2를 선택한 후 학습문서를 변경하면서 제안된 방법의 정확도를 측정하는 것이다. 학습문서의 크기에 따른 miR, miP, maR, maP의 변화는 <표 7>에서 보이고 있으며, 식 (6)과 <표 7>의 값에 따른 F_1의 변화는 <그림 2>에서 보이고 있다. <표 7>과 <그림 2>에서 DnK는 학습문서의 종류를 의미하며, NB는 기존의 Naive Bayes에 대한 정확도를 나타내며, pNB는 제안한 NaiveBayes에 대한 정확도를 나타낸다. <표 7>으로부터 miR, miP에서는 NB에 비하여 pNB의 정확도 개선효과가 거의 없다는 것을 알 수 있다. 반면, maR, maP에서는 NB의 정확도에 비하여 pNB의 정확도가 많이 개선되었음을 알 수 있다. <그림 2>는 학습문서의 크기에 따른 F_1의 변화를 그래프로 나타낸 것이다. F_1은 식 (6)을 통하여 알 수 있듯이 miR과 miP의 효과를 동시에 나타내는 것이다. <그림 2>에서 x축은 학습 문서의 크기를 나타내며, y축은 F_1의 값을 나타낸다. <그림 2>를 통하여 알 수 있듯이, 마이크로의 경우 NB와 pNB가 거의 비슷한 정확도를 나타내고 있다. 이것은 기존의 NB의 정확도가 0.7~0.85로 너무 높아 제안된 방법의 개선효과가 거의 없는 것으로 판단된다. 그러나 매크로의 경우 확실히 pNB가 NB보다 높은 정확도를 준다는 것을 알 수 있다. 특히, 제안된 방법의 경우 학습문서의 크기가 클 때 더 좋은 정확도 개선효과를 준다는 것을 알 수 있다.

5. 결 론

본 논문은 연관규칙탐사 기술에서 사용되는 빈발항목집합을 변형하여 문서분류의 문서에서

빈발단어집합을 정의하였고, 이를 잘 알려진 NaiveBayes에 적용하여 이 방법의 정확도를 개선하였다. 이를 위하여 하나의 문서를 여러 개의 문단으로 나누었으며, 각 문단에 나타나는 단어들의 집합을 트랜잭션화하여 빈발단어집합을 찾을 수 있도록 하였다. 제안한 방법은 기존의 잘 알려진 문서분류 프레임워크에서 구현되었다. 또한 로이터-21578 데이터를 사용하여 그 정확도가 측정되었다. 로이터-21578 데이터의 특성에 따라 하나의 문서를 여러 문단으로 나누는 문제를 문단당 라인수로 제한하였고, 이러한 제한으로 실험은 먼저 적절한 문단당 라인수를 찾는 것으로 하였다. 실험으로부터 문단당 2라인으로 할 때 가장 좋은 정확도 개선 효과를 얻을 수 있었으며, 이를 바탕으로 학습문서의 크기를 변화하면서 기존의 NaiveBayes와 제안한 방법의 정확도를 측정하였다. 실험으로부터 마이크로 측정에서는 두 방법이 거의 유사한 정확도를 나타내었고, 매크로 측정에서는 제안한 방법이 많은 정확도 개선을 준다는 것을 알 수 있었다.

문서분류는 적용되는 문서의 종류 즉, 뉴스 데이터, 의학 데이터, 패션데이터 등에 따라 동일한 문서분류 방법도 다른 결과를 가져다 준다. 따라서 본 논문에서 제안된 알고리즘은 향후 새로운 데이터에 확정 적용될 것이고, 또한 kNN, SVM 등에도 적용될 것이다.

참 고 문 헌

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

[2] Williams K. and R. A. Calvo, "*A Framework for Text Categorization*," 7th Australian Document Computing Symposium, Dec., 2002.

[3] 김한준, "*텍스트 마이닝 기술을 적용한 대용량 온라인 문서 데이터의 계층적 조직화 기법*", 서울대학교 대학원 박사학위 논문, 2002.

[4] Calvo, R. A. and J. M. Lee, "*Coping with the News : the machine learning way*," The 9th Australian World Wide Web Conference(AUSWEB 03), 2003.

[5] 이재문, "휴리스틱을 이용한 kNN의 효율성 개선", 정보처리학회논문지B, 제10-B권 제6호, 2003.

[6] Reuters-21578 Document Collection, http://about.reuters.com/researchandstandards/corpus.

[7] R. Calvo, J.M. Lee, X. Li, "Managing Content with Automatic Document Classification", Journal of Digital Information, Volume 5, 2004.

[8] J.M. Lee, R. Calvo, "Scalable document classification", Intelligent Data Analysis: An International Journal, Vol. 9, No. 4, pp 365-380, 2005.

[9] 이재문, 김동혁, "연관 규칙 기법을 이용한 NaiveBayes의 정확도 개선", 한국인터넷정보학회 추계학술발표대회, 401-405, 2005

[10] R. Agrawal, T. Imielinski and A. Swami, "Database Mining: A Performance Perspective", IEEE Trans. On Knowledge and Data Engineering, Vol. 5, No. 6, pages 914-925, 1993.

[11] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases", Proceedings of the 20th International Conference on Very Large Databases, 1994.

[12] A. Savasere, E. Omiecinski and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases", Proceedings of the 21th International Conference on Very Large Databases, pages 432-444, 1995.

[13] J.S. Park, M.-S. Chen and P.S. Yu, "An Effective Hash-Based Algorithm for Mining Association Rules", Proceedings of ACM SIGMOD, pages 175-186, 1995.

◐ 저 자 소 개 ◑

이 재 문 (Jae-Moon Lee)
1986년 한양대학교 전자공학과 졸업(학사)
1988년 한국과학기술원 전기및전자공학과 졸업(석사)
1992년 한국과학기술원 전기및전자공학과 졸업(박사)
1994~현재 한성대학교 멀티미디어공학과 교수
관심분야 : 데이터베이스, 기계학습, 게임 AI, etc.
E-mail : jmlee@hansung.ac.kr