계층구조 카테고리를 가지는 텍스트 분류 시스템

박지호[U] 김진상
계 명 대 학 교 컴 퓨 터 공 학 과
trois@jinri.kmu.ac.kr, jsk@kmu.ac.kr

A Text Classification System for Hierarchical Categories

Ji-Ho Park[O] Jin-Sang Kim
Dept. of Computer Engineering, Keimyung University

요 약

인터넷의 발전으로 온라인 문서들의 양이 급증하여 문서의 자동 분류 기술의 중요성이 증대되고 있다. 문서를 미리 정의된 카테고리로 분류할 때 카테고리는 평면구조보다 계층구조를 갖도록 하는 것이 사용자의 측면에서 볼 때 훨씬 더 자연스럽다. 본 논문에서는 계층구조 카테고리를 가지는 문서를 분류하는 방법을 연구하고 실제 20개의 유스넷 뉴스그룹 문서들을 분류하도록 시험하였다. 여기서 사용한 알고리즘은 하이퍼링크 정보를 이용하여 웹 문서 분류를 목적으로 개발된 IBM의 TAPER(taxonomy and path enhanced retrieval system) 알고리즘을 변형한 것이다.

1.서론

최근 인터넷과 인트라넷의 발전으로 온라인 문서의 양은 지속적으로 증가하고 있으며 온라인 문서의 양이 늘어날수록 유용한 정보를 발견하기 위한 검색(retrieval), 여과(filtering), 관리(management)의 중요성 또한 증대된다. 인터넷과 인트라넷에 포함된 온라인 문서들은 email, 뉴스그룹의 아티클, 웹 문서 등이 있는데, 본 논문에서는 뉴스그룹 아티클의 처리에 초점을 두고 있다.

현재 뉴스그룹은 수동으로 분류가 되고 있는데, 사용자는 미리 분류가 되어 있는 뉴스그룹으로 가서 문서를 게시(posting)하거나 읽을 수 있다. 만약 뉴스그룹에서 자동으로 문서를 분류하여 준다면, 사용자가 올린 글은 자동으로 분류가 되어 분류된 뉴스그룹에 게시되고 원하는 분야의 문서를 손쉽게 찾아서 읽을 수 있을 것이다.

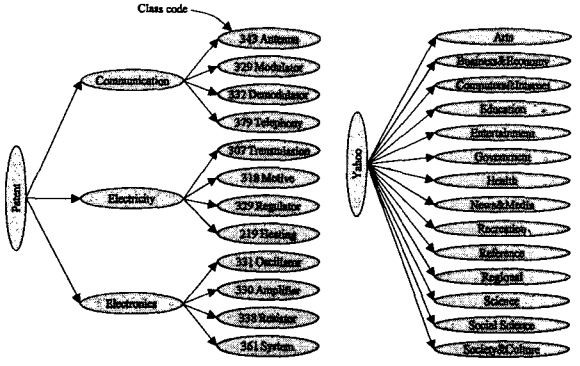
지금까지 온라인 문서들을 분류하려는 노력은 많이 있었다. 여러 가지 방법 중 일반적으로 높은 정확도를 가지는 방법이 베이시안 학습법이다. 본 논문에서는 기본적으로 베이시안 학습법[1]을 이용한 분류법을 사용한다. 현재 많은 문서 분류 시스템이 평면적 구조를 가지지만 우리가 접하는 대부분의 온라인 문서들은 개념적으로 계층을 가진 디렉토리 구조이다. 이 구조는 검색과 여과는 쉬우나 구조유지가 어려운 단점이 있다. 따라서 본 논문에서는 디렉토리 구조를 가진 텍스트 문서를 베이즈 학습법을 개선하여 분류하는 방법인 TAPER(taxonomy and path enhanced retrieval system) 알고리즘을 변형하여 구현하며[2,3], 이를 테스트하기 위해 20개의 뉴스그룹을 지정해서 각각의 뉴스그룹에 1,000개의 문서를 할당하여 학습과 분류를 수행해 그 결과를 분석한다.

2.관련연구

2.1 분류트리(taxonomy)

계층별 토픽을 분류 트리라고 하는데, [그림-1]은 분류 트리의 예를 보여 주는데 이와 같은 분류 트리는 검색과 접근은 용이하지만 온라인 텍스트 수가 지수승으로 증가

한다면, 수작업으로는 분류 트리의 관리가 거의 불가능해지기 때문에 자동 문서 분류 시스템이 요구된다[2].



[그림-1] 분류 트리의 예(a. 미 특허 문서, b. Yahoo의 첫 번째 레벨)

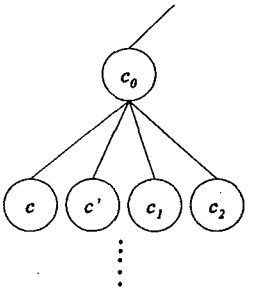
2.2 텍스트 분류

TAPER에서는 문서를 임의의 토큰들의 멀티셋(multiset)으로 표현하는데, 문서 분류의 식을 위한 기호들은 다음과 같다.

- 문서 d의 길이는 $n(d, c) = \Sigma_t n(t, d, c)$
- 클래스 c인 훈련 문서의 길이는 $n(c)$
- 클래스 c인 훈련 문서에서 단어 t의 총 빈도수
- d에서 t의 빈도수 $f(t, d, c) = n(t, d, c) / \Sigma_t n(t, d, c)$
- 특정 단어 t가 적어도 한번 나타난 클래스 c인 훈련 문서의 수는 $m(t, c)$
- 클래스 c인 훈련 문서의 수는 $|c|$

TAPER는 훈련 예제 문서로부터 클래스 트리를 학습해서 분류가 안된 새로운 문서가 주어졌을 때, 가장 잘 맞는 클래스를 트리로부터 찾아준다. 클래스 트리의 각 내부 노드는 결정점을 나타내고 분류기는 내부에서 가장 잘 맞는 자식 노드로 구체화 시켜 나간다.

학습을 위한 훈련은 먼저 문서를 검사하면서 각 단어들의 통계치를 수집한다. 문서의 검사 다음 단계가 특성 단어의 선택인데, TAPER에서는 임의의 부분집합을 찾는 것이 아니라 Fisher 테이블을 만들어서 각 단어마다 점수를 할당한다. 이 점수에 따라 역순으로 정렬을 하여 높은 점수의 단어를 선택한 것이 특성 단어이다. [그림-2]와 같이 내부 노드 c_0에서 c, c', c_1, c_2 등의 자식노드가 존재할 때, 각 단어의 점수는 식 (1)로 구할 수 있다.



[그림-2] 내부 노드 c_0에서 c, c', c_1, c_2 등의 자식노드

$$Fisher(t) = \frac{\sum_{c_1, c_2} (\mu(c_1, t) - \mu(c_2, t))^2}{\sum_c \frac{1}{|c|} (x(d, t) - \mu(c, t))^2} \quad (1)$$

$$\mu(c, t) = \frac{1}{|c|} \sum_{d \in c} x(d, t)$$

새로운 문서가 들어왔을 때 사후확률을 이용해 가장 높은 확률값을 가지는 노드로 이동을 해서 단말노드가 클래스 값이 된다. 사후 확률을 구하는 방법은 식(2)와 같다.

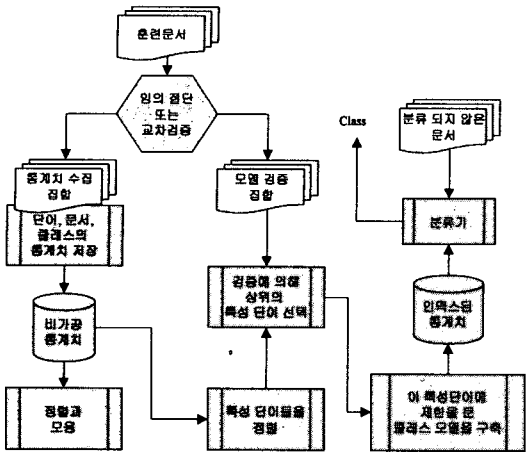
$$\Pr[d \in d\, c_0, F] = \frac{\pi(c) \prod_{t \in d \cap F} \theta(c, t)^{n(d, t)}}{\sum_{c'} \pi(c') \prod_{t \in d \cap F} \theta(c', t)^{n(d, t)}} \quad (2)$$

- $\theta(c, t)$는 클래스 c에서 t가 나타날 확률(베르누이 정리)
- $\pi(c)$는 사전확률
- $n(d, t)$는 문서 d에서 t가 나타나는 회수
- F는 Fisher 테이블에 속하는 단어

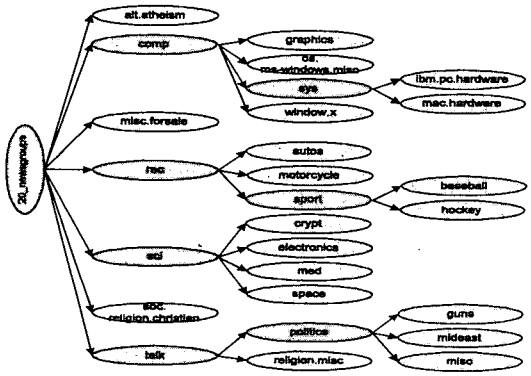
3.구현 및 결과

전체 시스템 구조는 [그림-3]과 같으며, 실제 분류에 적용한 20개의 뉴스그룹은 다음과 같고 [그림-4]는 이를 분류 트리 형태로 나타낸 것이다.

0	alt.atheism	10	rec.sport.hockey
1	comp.graphics	11	sci.crypt
2	comp.os.ms-windows.misc	12	sci.electronics
3	comp.sys.ibm.pc.hardware	13	sci.med
4	comp.sys.mac.hardware	14	sci.space
5	comp.windows.x	15	soc.religion.christian
6	misc.forsale	16	talk.politics.guns
7	rec.autos	17	talk.politics.mideast
8	rec.motorcycles	18	talk.politics.misc
9	rec.sport.baseball	19	talk.religion.misc



[그림-3] 시스템 구성도



[그림-4] TAPER를 위한 트리구조의 카테고리

20개의 뉴스그룹 각각에 1000개의 문서를 이용해 학습에 70%를 사용하고 30%는 분류의 정확도를 확인하기 위한 테스트로 사용하였을 때의 결과는 [그림-5]와 같다. 문서가 영어로 작성되어 있기 때문에, 학습과 분류시에 각각의 단어에 대해서 단어의 원형으로 바꿔주는 Poter 스테머를 이용하였다[4].

[그림-5]에서 보듯이 TAPER방법 알고리즘의 성능은 아직 완전하지 못하다. "comp.os.windows.misc"와 같은 경우 다른 클래스에 비해 현저하게 정확도가 낮은데, 그 이유는 특성 단어 추출시 각 클래스를 대표하는 단어를 선택하는데, 식 (1)에 의해 각 내부 클래스에서 자식 클래스 사이에 동시에 발생하지 않는 것이 Fisher 인덱스가 높게 된다. 그러나 "comp.os.windows.misc"는 이런 단어의 수가 훨씬 적고, 다른 자식 클래스에서 발생하는

Dataset : W20_newsgroupsW
Test ration : 30%
- Row is actual, Column is predicted(last column is 'unknown')

class	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	accuracy
0	258	0	0	0	0	0	2	1	0	0	0	0	1	0	2	26	0	0	2	7	1	86.29%
1	18	145	2	12	2	13	60	2	2	1	3	10	13	2	8	4	0	2	0	1	0	48.33%
2	6	34	3	84	12	50	71	2	3	3	1	6	4	2	8	6	0	0	4	0	1	1.00%
3	9	5	2	142	11	1	103	6	1	1	1	2	10	0	2	2	1	0	1	0	0	47.33%
4	7	5	0	27	110	1	117	6	1	0	0	4	10	3	6	1	0	0	0	1	1	36.79%
5	2	27	0	14	6	194	30	2	1	1	0	8	4	2	7	0	1	0	0	0	1	64.88%
6	5	1	2	7	4	1	250	6	3	3	1	1	4	2	3	0	1	0	0	1	5	84.75%
7	10	1	2	1	2	0	38	220	8	1	1	2	2	0	3	0	6	1	1	1	0	73.33%
8	4	1	0	0	0	0	8	12	266	1	0	0	1	1	1	2	3	0	0	0	0	88.67%
9	5	0	0	0	0	0	12	8	6	261	4	0	1	0	0	1	1	0	1	0	0	87.00%
10	5	0	0	1	0	1	2	3	6	21	257	1	0	2	0	1	0	0	0	0	0	85.67%
11	2	1	1	1	0	2	10	0	2	0	0	235	17	4	3	7	8	0	8	0	1	78.60%
12	5	6	1	9	10	0	89	19	4	3	0	15	127	3	3	3	0	0	1	2	0	42.33%
13	33	2	1	0	0	0	11	5	10	1	2	4	16	180	7	17	7	0	2	1	1	60.20%
14	13	4	0	0	0	1	6	1	0	0	0	5	3	4	247	7	2	1	5	1	0	82.33%
15	8	0	0	0	1	0	3	0	0	1	0	0	0	0	0	282	1	0	2	4	0	94.00%
16	6	0	0	0	0	0	9	1	4	0	2	10	0	1	2	8	218	3	20	16	0	72.67%
17	26	0	0	1	0	0	3	1	3	2	0	2	0	1	0	9	6	227	18	3	0	75.67%
18	15	0	0	0	0	0	3	0	5	1	2	2	0	2	2	7	47	8	170	35	1	56.86%
19	100	0	0	0	0	0	5	0	4	1	1	1	0	3	4	88	12	1	8	72	0	24.00%

Correct : 3864 out of 6000
Accuracy average : 64.40%

[그림-5] TAPER를 이용한 분류 결과

단어와 비슷하게 되기 때문에 다른 클래스와의 연관도가 높아져서 분류의 정확도가 떨어진다고 볼수 있다.

4.결론

본 논문에서는 문서의 자동분류 기법중 하나인 TAPER 시스템에서 소개한 알고리즘을 변형하여 시스템을 구현하였으며, 20개의 뉴스그룹 아티클을 대상으로 분류를 수행하였다. 그 결과 평균 정확율은 약 64%였으며, 계층구조를 무시한 경우 약 69%였다. 반면 계층구조를 지원하지 않은 NBC의 경우는 77%정도로 나타났는데, 아직 TAPER 알고리즘이 NBC에 미치지 못하지만 계층구조를 지원한다는 장점이 있다. 하지만 웹문서의 경우 계층구조 카테고리가 유리하기 때문에 앞으로 계층구조 카테고리화로 전환이 필요하다.

5.참고문헌

[20] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[2] S. Chakrabarti, B. Dom, R. Agrawal, P. Indyk, "Enhanced hypertext categorization using hyperlinks.", *SIGMOD ACM*, 1997.

[3] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies.", *VLDB Journal*, 1998.

[4] M.F. Porter, "An algorithm for suffix stripping", _Program_, Vol. 14, No. 3, 1980.