공학석사학위논문

Naive Bayes Boosting을 이용한 문서 여과

Text Filtering by Boosting Naive Bayes

2001년 2월

서울대학교 대학원
전기·컴퓨터공학부
김 유 환

Naive Bayes Boosting을 이용한 문서 여과

Text Filtering by Boosting Naive Bayes

지도 교수 김 영 택

이 논문을 공학석사 학위논문으로 제출함

2000년 10월

서울대학교 대학원

전기 · 컴퓨터공학부

김 유 환

김유환의 공학석사 학위논문을 인준함

2001년 12월

위원장 김 형 주

부위원장 김 영 택

위 원 장 병 탁

목 차

1 장 서론 **1**

2 장 관련 연구 **4**

1 절 벡터 스페이스 모델에 기반한 정보 여과 4

2 절 확률 모델에 기반한 문서 여과 5

3 절 기계학습에 기반한 문서 여과 9

3 장 BayesBoost에 의한 문서 분류 **11**

1 절 복수 개의 분류자 결합을 통한 문서 분류 11

2 절 AdaBoost 13

3 절 Naive Bayes 분류자 (naive Bayes Classifier) 14

4 절 BayesBoost: Boosted Naive Bayes 16

5 절 문서 길이를 고려한 BayesBoost 알고리즘 18

4 장 실험 및 결과 **22**

1 절 성능 측정 방법 22

2 절 TREC-7 과 TREC-8 필터링 트랙 데이터셋 23

3 절 실험 설계 및 결과 26

5 장 분석 **35**

1 절 다른 시스템과의 비교 분석 35
2 절 문서 길이를 고려하였을 때의 효과 분석 37

6 장 결론 **38**

표 목차

2.1 단어 출현 테이블 7

4.1 TREC-7 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (1) 30
4.2 TREC-7 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (2) 31
4.3 TREC-8 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (1) 32
4.4 TREC-8 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (2) 33

5.1 가우시안 분포 가정과 균등 분포 가정에 대한 실험 37

그림 목차

3.1 BayesBoost 알고리즘 21

4.1 TREC 배치 필터링 트랙에서 사용되는 문서의 예 24
4.2 TREC 배치 필터링 트랙에서 사용되는 질의문의 예 25
4.3 BayesBoost와 TREC-7에 제출되었던 결과 중 상위 세 개와 비교 . . 27
4.4 BayesBoost와 TREC-8에 제출되었던 결과 중 상위 세 개와 비교 . . 28
4.5 BayesBoost와 AdaBoost의 성능 비교 34

초 록

최근들어 문서 분류 및 여과에 기계학습의 도입이 활발히 이루어 지고 있다. 이중 특히 boosting 방법은 텍스트 문서 분류에서 좋은 성능을 보여주고 있다. 그러나 지금까지 문서 분류에 사용된 대부분의 boosting 알고리즘이 단어의 존재 여부에만 의존하여 문서를 분류하기 때문에 문서에서의 단어의 발생 빈도나 문서의 길이와 같은 텍스트 문서의 중요한 특징을 충분히 이용하지 못하는 단점을 안고 있었다.

문서 여과의 다른 특징 중 하나는 질의에 적합한 문서를 추출하지 못한 경우의 비용이 질의에 적합하지 않은 문서를 추출하는 비용보다 더 크다는 점이다. 그러므로 문서 여과 알고리즘을 설계할 때는 이러한 점을 이용하여야 하나 AdaBoost는 에러를 줄이는 것으로 최적화 되어 있어 비용의 개념이 없다는 단점이 있다.

본 논문에서는, 수정된 Naive Bayes분류자를 약 학습자로 사용하여 boosting한 학습 알고리즘을 제안하였다. 수정된 Naive Bayes를 사용함으로서 얻는 이점은 단어의 빈도나 문서의 길이와 같은 추가적인 정보를 이용할 수 있다는 점이다. 추가적으로 AdaBoost 알고리즘도 비용을 고려하여 다시 설계를 하여야 한다.

TREC-7과 TREC-8의 필터링 트랙 문서에 적용해 본 결과 TREC에서 좋은 성능을 보인 알고리즘과 비교하여 볼 때 제안된 학습 알고리즘은 LF1, LF2, F1, F3에서 모두 우수한 성능을 보여 주었다. 또한 결정 스텀프를 사용한 AdaBoost 알고리즘에 대해 실험해 본 결과 본 알고리즘은 AdaBoost에 비해 스케일드 선형 유틸리티로 측정하였을 때 평균 0.16의 성능 향상을 이루었음을 알 수 있다.

1 장

서론

정보 여과(Information Filtering)는 정보를 필요로 하는 사람에게 필요한 정보를 전달해 주고 필요 없는 정보는 여과하는 과정을 의미한다 [1]. 최근 들어 인터텟의 발달로 인한 전자 문서의 증가로 정보 여과는 정보 검색 분야에서 가장 중요한 응용 분야중 하나로 자리매김하였다. 정보 여과에 대한 관심 증대에 대한 일례로 정보 검색에 대한 중요한 학회 중 하나인 TREC 학회도 95년에 개최된 TREC-4 부터 필터링 트랙(Filtering Track)을 도입하여 많은 연구자들이 여기에 참가하고 있다 [30].

정보여과 시스템은 사용자들의 기호를 프로파일의 형태로 학습하여 새로운 문서가 들어오는 경우에 기존의 프로파일과 새롭게 들어온 프로파일을 비교하여 적합하다고 판단되는 문서만을 추출하는 시스템이다. 정보 여과에서 프로파일은 문서 여과 과정에서 새롭게 얻은 정보를 이용하여 갱신될 수도 있고, 한 번 생성된 이후에는 더 이상 바뀌지 않을 수 있다. 전자의 경우는 적응적 문서 여과라 불리며 후자의 경우는 배치 문서 여과라 불린다. TREC학회에서는 적응적 문서 여과 시스템을 테스트 하기 위해 필터링 트랙의 세부 영역으로 적응적 여과 트랙(Adaptive filtering track)을 두고 있고 배치 문서 여과 시스템을 위해서는 배치 필터링 트랙(Batch filtering track)을 두고 있다 [10]. 이 중 본 논문에서는 후자인 배치 문서 여과에 대해서만 다룬다.

문서 여과의 또 다른 특징 중 하나는 대부분의 경우 적합성의 여부만을 판단한다는 것이다. 이는 문서를 적합도 순으로 배열하는 라우팅 방법보다 한단계 더 나아간 것인데 이는 문서를 적합도 순으로 배열하는 작업에 더하여 적합도가 기준값(Threshold)이하인 문서들은 버리고, 그 이상인 문서들만을 추출하는 작업까지 해주어야 하기 때문이다. 다시 말하면 라우팅 문제에 기준값 설정의 문제가 더해진 형태로 볼 수 있다.

정보 여과는 문서 분류 알고리즘과 밀접하게 관련되어 있다. 앞서 언급한 배치 정보 여과의 경우는 이진 문서 분류 알고리즘으로 표현할 수 있으며, 적응적 정보 여과는 온라인 문서 분류 알고리즘으로 표현할 수 있다. 그러므로 문서 여과의 성능은 분류 알고리즘의 성능과 밀접한 관련을 가지게 된다. 최근 들어 분류 알고리즘 중 좋은 성능을 보이고 있는 알고리즘 중 하나는 boosting 알고리즘으로서, 이 중 특히 AdaBoost 알고리즘이 좋은 성능을 보이고 있다 [8] [24]. AdaBoost에서는 각각의 약 학습자 (임의로 추측하는 것보다 약간 더 잘하는 분류자)가 과거의 약 학습자가 분류하기 어려운 문서를 집중적으로 분류하게 된다. 이렇게 해서 만들어진 일련의 학습자는 가중치를 갖는 투표를 통해 최종적인 분류자를 생성하게 된다. 문서 분류에 대해서도 많은 연구자들이 AdaBoost 알고리즘을 적용하여 보았다 [6] [19] [25]. 이러한 방법들에서 사용된 약 학습자로는 C4.5와 같은 결정 트리나 결정 스텀프가 있다. 그러나 이러한 약 학습자들은 문서 표현에서 얻을 수 있는 정보를 충분히 이용하지 못한다는 단점이 있다. 다시 말해서 이러한 방법들이 단어의 존재 여부를 기반으로 문서를 분류하기 때문에 단어 가중치나 문서 길이와 같은 문서 데이터의 중요한 정보를 제대로 이용하지 못한다는 점이다.

이러한 문제를 해결할 수 있는 방안으로 본 논문에서 제시하고 있는 약 학습자는 Naive Bayes 분류자이다. 그러나 Naive Bayes 분류자는 단어의 가중치 정보를 이용하기는 하나 문서 길이를 고려하지는 않기 때문에 본 논문에서는 Naive Bayes 분류자를 약간 수정하여 사용하였다. Naive Bayes의 좋은 성질중 하나는 결정 트리와 달리 분류 결과뿐 아니라 분류의 확신도(confidance ratio)를 함께 출력하여

준다는 사실이다. 실험적으로 확신도 정보를 함께 이용한 boosting 방법이 그렇지 않은 boosting 방법보다 더 좋은 성능을 보인다는 사실이 알려져 있으므로 본 논문에서도 이러한 확신도 정보를 boosting에 이용하였다. 실험 결과 수정된 Naive Bayes를 boosting의 약 학습자(weak learner)로 사용하여 Naive Bayes를 boosting시킨 알고리즘은 텍스트 문서 여과에 효과적으로 사용될 수 있음을 알 수 있었다. 설명의 편의상 앞으로는 본 알고리즘을 BayesBoost라는 이름으로 사용할 것이다. BayesBoost의 성능을 테스트 해 보기 위해 본 논문에서는 TREC-7 과 TREC-8 필터링 트랙 데이터셋에 알고리즘을 적용해 보았다. 실험 결과 지난 2년동안의 TREC에 제출된 결과중 가장 좋은 결과 세개와 비교해 보았을때 LF1, LF2, F1, 그리고 F3 측정치(measure)에 대해 좋은 보여주었다. 특히 LF1, LF2, F1의 경우는 제출된 가장 좋은 결과와 비슷한 성능을 보여 주었다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 문서 분류에 사용되는 기계학습 방법에 대해서 논라고자 한다. 3장에서는 AdaBoost와 Naive Bayes 학습자에 대해서 설명하고 BayesBoost알고리즘을 도입하고자 한다. 4장에서는 TREC-7과 TREC-8 필터링 데이터셋에 대한 실험에 대해 알아보고, 마지막 장인 5장에서 본 연구의 결론을 내린다.

2 장

관련 연구

정보 검색을 위한 전통적인 모델은 크게 두 종류로 분류 할 수 있다. 하나는 벡터 스페이스 모델 (Vector space model)이고 다른 하나는 확률 모델 (Probabilistic model)이다. 두 방법 모두 각각 SMART시스템과 OKAPI시스템에 구현되어 있으며 웹에서 구할 수 있다.

1 절 벡터 스페이스 모델에 기반한 정보 여과

벡터 스페이스 모델은 문서를 각 차원이 단어의 가중치를 나타내는 n-차원 공간상의 한 점으로 표현한다. 단어의 가중치는 통상 $tf \cdot idf$ 혹은 이의 변형으로 나타낸다. 여기서 tf는 문서에 단어가 나타난 빈도를 나타내면 idf는 해당 단어를 포함하고 있는 문서의 개수의 역을 취한 것이다. 즉 위의 두 값을 곱하면 문서에 단어가 너무 적게 나타난다거나 다른 문서에도 자주 나타나서 특질로서의 효과가 미미한 경우는 작은 가중치를 가지게 되는 효과가 있다. 여기서 n은 전체 데이터셋에서 사용되는 단어의 개수를 의미한다. 벡터 스페이스 모델에서 질의가 주어진 경우 문서 검색은 질의도 하나의 문서로 본다면, 문서 사이의 유사도(similarity)를 측정하여 유사도가 높은 문서를 출력하는 방식을 취한다. 이 때 유사도를 구하기

위해서는 일반적적으로 코사인 측정치(cosine measure)를 사용한다. 이렇게 검색된 문서에 대해 사용자가 컴퓨터에 호, 불호의 의사 표현을 할 수 있는데 이를 적합도 피드백(relevance feedback)이라고 한다. 벡터 스테이스 모델의 적합도 피드백 방법은 대부분 Rocchio 알고리즘과 관련되어 있다. 이 중 유의할 만한 성과로는 Dynamic feedback optimization (DFO) [4], Query Zoning [27] 등이 있다. 이 중 Query Zoning은 Rocchio 식을 계산할 때 사용되는 적합하지 않은 문서의 집합을 알고 있는 모든 적합하지 않은 문서의 집합을 사용하는 것이 아니라 적합한 것으로 분류될 위험성이 높은 적합하지 않은 문서를 사용하는 방법이다. 이러한 방법은 분류 평면(separating hyperplane)의 근처에 있는 적합한 문서와 비적합한 문서들의 분류에 중점을 두는 효과가 있기 때문에 훨씬 더 좋은 성능을 보일 수 있다. 이 외에도 유의할 만한 성과로는 문서 길이를 고려한 정보 검색을 위해 Singhal이 제안한 Document length normalization [28]이 있다.

2 절 확률 모델에 기반한 문서 여과

확률 모델에 기반한 방법도 정보 검색 분야에서 오래 전부터 사용되었다 [5] [16] [22]. 확률적인 접근 방법에 따르면 추출된 문서들은 적합할 확률에 따라 정렬된다. 그러므로 추출된 문서이 실제 확률은 중요하지 않게 된다. 반면에 중요한 것은 문서의 순위(rank)이다. 이러한 사실은 확률 순위 원리(probability ranking principle)에 의해서 정당화 될 수 있다 [22]. 확률 순위 원리는 다음과 같이 표현된다.

> 만약 어떤 시스템이 추출된 문서들을 이용 가능한 데이터에 대해서 최대한 정확하게 측정된 적합도 확률(Probaility of relevance)에 대해 내림차순으로 정렬할 수 있다면, 그 시스템의 성능은 주어진 데이터에 대해서 얻을 수 있는 가장 좋은 성능이다.

확률 모델에 대한 몇 가지 기본적인 개념을을 설명하기에 앞서 몇가지 기호들을 살펴보기로 한다. d_i는 문서 집합에서 i번째 문서를 나타내며, c_j는 범주(class or

category)를 나타낸다. 본 논문은 문서 여과를 다루기 때문에 범주는 질의에 대한 적합한지의 여부에 따라 두 가지 클래스 혹은 범주만 존재하게 된다. 본 논문에서는 적합한 범주를 표시하기 위해 c_1을 사용하였으며, , 그렇지 않은 경우에는 즉 문서가 질의에 적합하지 않는 경우에는 c_0으로 표시하였다. 여기서 우리가 측정하고 싶은 확률은 바로 $P(c_j|d_i)$이다. Bayes 규칙 (Bayes rule)을 적용하면 이 확률은 다음과 같이 분해될 수 있다.

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

그러나 이 식을 그대로 사용하면 $P(d_i)$를 계산해야 한다는 부담이 따르기 때문에 이 문제를 해결하기 위해서는 로그 오드(log-odd)를 사용하면 된다. 이 때 식은 다음과 같이 변형된다.

$$\begin{aligned} \log \frac{P(c_1|d_i)}{P(c_0|d_i)} &= \log \frac{P(d_i|c_1)P(c_1)}{P(d_i|c_0)P(c_0)} \\ &= \log \frac{P(d_i|c_1)}{P(d_i|c_0)} + \log \frac{P(c_1)}{P(c_0)} \end{aligned}$$

여기서 마지막 항이 D에 독립이기 때문에 버려도 순위를 매기는 대는 영향이 없다. 많은 알고리즘이 바로 위에 제시한 일반적인 모델에 근거하여 제안되었다. 이 중 가장 기본적인 방법이 적합도 정도가 주어졌을 때 단어들 사이의 조건부 독립을 가정하는 베이지안 독립 모델(Bayesian independence model, BIM)이다. 이러한 가정은 많은 학습 알고리즘이 가지고 있는 가정으로 Naive Bayes도 분류자도 이러한 가정을 가지고 있다. Bayes 분류자를 들 수 있다. 이러한 가정에서는 문서의 확률은 다음과 같이 매우 쉽게 계산될 수 있다.

$$\frac{P(c_1|d_i)}{P(c_0|d_i)} = \sum_k \log \frac{P(w_{d_{ik}}|c_1)}{P(w_{d_{ik}}|c_0)}$$

여기서 $w_{d_{ik}}$는 문서 d_i에서 k번 째 단어의 속성을 나타낸다. 이 자체만으로도 매우 계산하기 쉽게 보이지만 위의 식은 단어가 존재하는 경우 뿐 아니라 단어가 문서에 존재하지 않는 경우에도 확률을 계산해 주어야 하는 단점이 있다. 이러한 단점

표 2.1: 단어 출현 테이블

	Relevant	Non-relevant	
Containing the term	r	n - r	n
Not containing the term	R - r	N - n - R + r	N - n
	R	N - R	N

은 단어가 문서가 존재하지 않는 경우를 자연 영점(natural zero)으로 처리함으로서 수식에 약간의 수정을 가해 주면 쉽게 해결할 수 있다. 즉, 모든 문서의 점수에서 속성들의 자연 영점때의 값을 빼 주면 된다. 이와 같은 과정을 거쳐서 원래의 순위를 그대로 유지하면서 더욱 단순해진 새로운 수식을 유도할 수 있다.

$$\frac{P(c_1|d_i)}{P(c_0|d_i)} = \sum_k \log \frac{P(a_{d_{ik}}|c_1)P(a_{d_{ik}}|c_0)}{P(a_{d_{ik}}|c_0)P(a_{d_{ik}}|c_1)}$$

만약 우리가 수식의 오른쪽을 $W(a_{d_{ik}})$로 표시한다면 식은 다음과 같이 간략화 될 수 있다.

$$\frac{P(c_1|d_i)}{P(c_0|d_i)} = \sum_k W(a_{d_{ik}})$$

$W(a_{d_{ik}} = 0)$이 항상 0이기 때문에, 속성이 0값을 가지는 경우는 무시하면 된다. 속성 $a_{d_{ik}}$가 단순이 단어의 존재 여부를 가리키는 것이라고 가정해 보자. 이런 경우에는 W식은 다음과 같이 단어가 존재하는 경우의 가중치(weight)를 가리킨다.

$$W(i) = \log \frac{p_i(1-\bar{p}_i)}{\bar{p}_i(1-p_i)}$$

그러므로 문서의 점수는 존재하는 단어에 대한 가중치의 총합으로 표시될 수 있다. 여기서 p와 $\bar{p}$의 추정은 다음과 같이 단어 출현 테이블(term incidence table)에 기반하여 추정할 수 있다 [23].

여기서 R 은 질의에 대해 적합한 문서의 개수를 나타내며, r은 특정 단어를 포함하고 있는 문서의 개수를 나타낸다. 비슷하게 N은 질의에 대해 적합하지 못한 문서의 개수를 나타내면 r은 특정 단어를 포함하지 않고 있는 문서의 개수를 나타낸다.

위의 테이블을 이용하면 p_i 와 $\bar{p}_i$ 는 다음과 같이 계산될 수 있다.

$$p_i = \frac{r}{R}$$
$$\bar{p}_i = \frac{n-r}{N-R}$$

그러므로,

$$W(i) = \log \frac{r(N-n-R+r)}{(R-r)(n-r)}$$

와 같이 표현 할 수 있다. 이러한 사실은 벡터 스페이스 모델에서 단어의 가중치를 매길 때 휴리스틱으로 사용했던 *idf* 에 대한 확률적 해석을 할 때 도 사용될 수 있다는 점이 흥미롭다. Robertson 과 Walker는 만약 현재 적합한 문서 집합을 가지고 있지 않는 경우에 가정에 약간의 수정을 가하면 *idf* 와 같은 식을 얻을 수 있음을 보였다. 추정의 정확도를 높이기 위하여 수식에 약간의 수정을 가하면 다음과 같은 식을 얻을 수 있다.

$$W(i) = \log \frac{(r+0.5)(N-n-R+r+0.5)}{(R-r+0.5)(n-r+0.5)}$$

지금까지는 문서에 단어의 존재 여부만을 가지고 식을 유도해 보았다. 이제부터는 이를 더 확장하여 문서 내의 단어의 빈도수 (Term frequency)까지 고려하는 모델을 만들어 보기로 하자. 단어의 빈도수를 모델링하는 방법을 여러가지가 있는데 이중 제일 성공적인 방법이 단어의 빈도수가 포아송 분포를 따른다고 가정하는 것이다 [9]. 여기서 설명하는 모델은 Harter의 모델을 확장한 것이다 [21]. Robertson은 각 단어는 토픽과 연관이 있다는 가정과 문서는 토픽에 대한 문서이거나 그렇지 않은 문서라는 가정을 하였다. Robertson의 연구에서 중요한 것은 문서에 나온 단어는 모호함(unpredictability)을 가지고 있다는 점이다. 예를 들어 문서 내에서 어떤 단어를 사용하였을 때 그 단어가 가리키는 주제와 상관없는 주제에 대해서 기술하는 경우라도 그 단어를 우연히 사용할 수 있기 때문이다. 문제는 우리는 어떤 문서가 단어가 가리키는 주제에 대한 것이고 어떤 문서가 그렇지 않은 것인지 모른다는 점이다. 그러므로 우리가 관찰하는 문서 내부의 단어 빈도수(within-document term

frequency)의 분포는 두개의 포아송 분포의 결합확률 문포(mixture)로 보는게 더 합당할 것이다. 설명을 진행하기 앞서 먼저 엘리트(elite)를 정의하도록 하자. 문서가 단어가 가리키는 개념이나 주제에 대한 것이지를 나타내는 속성을 단어에 대한 엘리트라 정의한다. 그리고 이것을 기호로 E로 표시하자. 예를 들어 i번째 단어에 대한 엘리트는 E_i로 표시할 수 있다. 이 때 우리는 다음과 같은 수식을 얻을 수 있다.

$$P(tf_{d_{ik}}|c_1) = P(tf_{d_{ik}}|e)P(e|c_1) + P(tf_{d_{ik}}|\bar{e})P(\bar{e}|c_1)$$

여기서 $tf_{d_{ik}}$ 는 문서 내 단어 빈도수(within-document term frequencies)를 의미한다. 이 식은 직접적으로 계산하는게 불가능하기 때문에 Robertson과 Walker는 이 식을 분석하여 좀더 간편하면서도 위의 식과 비슷한 특성을 같는 수식을 제안하였다. 이 수식은 다음과 같다.

$$W = \frac{tf_i(k_1+1)}{k_1+tf_i}w_i$$

여기서 tf_i는 단어 t_i에 대한 빈도수를 의미한다. 그리고 k_1는 경험적으로 결정될 수 있는 상수값이다. 위의 모델을 더 확장하여 문서의 길이까지를 고려하는 알고리즘을 만들 수도 있다. 자세한 증명과정을 생략하고 결과만을 보이면

$$nf = (1-b) + b\frac{dl}{avdl}$$

로 놓았을때 다음 식은 문서의 길이를 고려한 경우의 가중치를 나타내고 있다.

$$W(tf_i) = \frac{tf_i(k_1+1)}{k_1 * ((1-b)+b\frac{dl}{avdl})+tf_i}w_i,$$

여기서 dl은 문서의 길이를 나타내며, $avdl$은 평균 문서 길이를 나타낸다. 그리고 k_1와 b는 둘 다 상수이다.

3 절 기계학습에 기반한 문서 여과

최근 들어 기계학습 분야에서 연구된 성과를 바탕으로 문서 여과에 다양한 방법이 적용되었다. 예를 들어 신경회로망, k최근점 학습법, 서포트 벡터 머쉰, Naive Bayes

분류자 등이 그러한 예가 될 수 있다. Yang [31] 은 이러한 기계학습 방법을들 비교하고 다양한 통계 수치를 이용하여 평가하고 있다.

Lewis et al. [14]는 Widrow-Hoff와 EG (Exponentiated Gradient) 두 종류의 기계학습 알고리즘을 필터링에 도입하고 이 알고리즘이 모두 Rocchio알고리즘보다 경험적으로 좋은 결과를 출력한다는 사실을 밝혔다.

Naive Bayes [18] 또한 정보 검색 분야에 적용되어 비교적 좋은 성능을 보이고 있다. McCallum et al. [17] 은 Naive Bayes에 기반하고 expectation-maximization (EM) 알고리즘을 사용하여 범주 정보가 있는 데이터와 없는 데이터가 존재하는 상황에서 훈련시키는 방법을 개발하였다. REUTERS-21578에서의 실험에서 이 알고리즘은 단순한 Naive Bayes보다 더 좋은 성능을 보여 주었다.

3 장

BayesBoost에 의한 문서 분류

1 절 복수 개의 분류자 결합을 통한 문서 분류

여러 연구를 통해서 동종의 혹은 이종의 분류자를 여러개 묶어서 더 정확하고 안정적인 분류자를 만들 수 있다는 사실이 밝혀졌다. 이러한 방법들은 앙상블 머쉰(ensemble machines)이라 불리는데, 앙상블 머쉰에 만들기 위해서는 크게 두 가지 문제를 해결해야 한다. 하나는 각각의 앙상블 머쉰의 요소가 되는 서브 분류자(sub-classifier)를 어떻게 만들것인가하는 문제이고, 다른 하나는 이렇게 만들어진 서브 분류자의 출력을 어떻게 하나로 묶을 것인가에 대한 문제이다. 각각의 서브 분류자를 만들기 위해서는 크게 두 가지 방법이 사용된다. 하나는 각각의 서브 분류자를 다른 파라미터를 이용하여 훈련시키는 방버이다. 예를 들어 뉴럴 넷에서 초기 파라미터를 각각다르게 설정하는 것이 한가지 예가 될 수 있다. 다른 하나는 훈련 시킬 문서의 가중치를 변경 (document reweighting)시키거나 분포를 변경 (docuemnt resampling) 시키면서 훈련시키는 방법이다. bagging과 boosting이 이와 같은 접근 방법을 취하고 있다. 배깅 (Bagging or Bootstrap aggragating)의 경우는 주어진 데이터에서 부트스트랩 리샘플링을 통하여 정확도를 높이고자 한다. 여기서 부트스트랩 리샘플링이란 주어진 훈련 데이터에서 반복적으로 랜덤 리샘플링하는 방법니

다. 즉 n개의 훈련 데이터가 있는 경우 각각의 훈련 데이터가 나타날 확률을 $\frac{1}{n}$로 놓고 여기서 중복을 허용하여 n개의 데이터를 추출하는 방법이다. 반면 Ada Boost와 이의 더 일반적인 아킹은 문서의 난이도에 따라 문서의 가중치를 변경하거나 문서의 분포를 변경하여 새로 생성되는 서브 분류자는 이전에 잘 하지 못했던 부분을 더 잘 할 수 있도록 서브 분류자를 생성한다. AdaBoost와 아킹은 만들어진 분류자들을 결합하는 경우에도 각각의 분류자들의 가중치를 적응적으로 계산한다 [3] [24]. 이는 과거의 알고리즘에 비해 진일보한 것으로서 고전적인 앙상블 평균(ensemble averaging)의 경우는 단순히 생성된 서브 분류자의 출력의 평균을 구하는 방식을 취하고 있기 때문이다.

만약 각각의 서브 분류자들이 같은 출력을 낸다면 최종적인 성능을 향상되지 않을 것이다. 그러므로 서브 분류자의 학습과 선택은 매우 중유한 문제이다. 지금까지 알려진 바로는 서브 분류자의 선택은 서브 분류류자간의 상관계수와 많은 관련을 가지고 있다. 그래서 만일 각각의 서브 분류자들이 서로 독립적이지 않은 경우에는 약간의 성능 향상밖에 볼 수 없다는 사실이 알려져 있다 [2]. 앙상블 모델과 다른 접근 방법으로는 adaptive mixtures of local experts model이 있다. 이 방법에서 각각의 분류자 혹은 전문가(expert)는 데이터의 다른 면을 학습하게 된다. 이 방법은 앞서 언급한 방법과는 큰 차이가 있는데 앞서의 방법은 각각의 서브 분류자들이 데이터 분포의 전반적인 특성을 학습하는 반면 여기서 설명하는 모델은 전문가들이 각각 국소 분포(local distribution)만을 학습하기 때문이다 [11].

여러 연구를 통해 AdaBoost가 매우 좋은 성능을 보임이 밝혀졌다 [6] [19]. 특히 Schapire et al.는 REUTER-21578과 TREC-3에서 AdaBoost 알고리즘을 Rocchio 알고리즘과 비교함으로서 AdaBoost 알고리즘이 텍스트 필터링(text filtering)에도 좋은 성능을 보임을 실험적으로 밝혔다. 실험 결과 두 알고리즘이 모두 좋은 성능을 보였으나 TREC-3의 LF2에 대한 실험에서는 Rocchio보다 좋지 않은 성능을 보여주었다. 이는 AdaBoost 알고리즘이 정확도(Accuracy)를 높이는데 최적화 되어 있는 반면 LF2 측정 방법은 적합한 문서를 추출하는 경우와 적합하지 않은 문서를 뽑

는 경우의 비용의 차이가 크게 만들어져 있기 때문이다.

2 절 AdaBoost

AdaBoost (AdaBoost)는 분류자를 순차적으로 생성해 낸 후 이를 결합함으로서 더 낳은 성능을 보이는 분류자를 생성해 내는 것을 목적으로 한다. 이 방벙은 처음에 Shapire에 의해서 제안된 후 많은 연구자들에 의해서 연구되고 실제 문제에 응용되었다. Shapire는 또한 AdaBoost가 텍스트 분류 문제에도 잘 적용된다는 것을 보였다 [25] [26]. 여기서 'boosting'은 알고리즘이 문서의 중요도(importance factor)를 재조정함으로서 새로운 문서 중요도를 생성하기 때문에 붙여진 이름이다. 각각의 가설(hypothesis)는 이렇게 새롭게 매겨진 중요도를 바탕으로 훈련을 진행하게 된다. 'Ada'는 이렇게 만들어진 각각의 가설(θ_t)의 투표 가중치(α_t)를 적응적으로 계산할 수 있다는 의미이다. 초기의 AdaBoost는 각각의 가설 혹은 약 학습자의 출력이 -1 이나 1의 둘 중의 하나가 되도록 제한하였다.

좀 더 최근의 버전의 AdaBoost 알고리즘은 이러한 제한을 없애고 약 학습자의 출력이 실수 범위를 가질 수 있도록 하였다 [24]. 이 때 약 학습자($\theta_t(x)$)의 출력의 부호는 x에 해당하는 (-1 또는 1)의 범주를 가리키며, $|\theta_t(x)|$의 값은 이러한 추정에 대한 확신도(Confidence)를 가리킨다. 그러므로 만약 $\theta_t(x)$ 가 0에 가까우면 낮은 확신도를 가지고 추정하는 것이고 0에서 멀어질수록 높은 확신도를 가지고 추정하는 것이 된다. 이 논문에서는 아래에 설명하듯이 알고리즘의 간략화를 위해서 θ_t의 범위를 [-1,+1]로 제한했다. AdaBoost는 최종적으로 이러한 약 학습자를 결합하여 분류자를 만든다.

$$f(x) = \sum_{t=1}^{T} \alpha_t \theta_t(x),$$

여기서 T 는 반복 회수를 의미하고 α_t 는 약 학습자의 무게를 나타낸다. 만약 우리가 $\theta_t(x)$의 범위를 [-1,1]로 제한하는 경우라면, α_t를 원래 AdaBoost 알고리즘의 식을 그대로 사용할 수 있다. 그렇지 않은 경우에는 일반적으로 이진 탐색과 같은 수

치 해석적인 방법으로 찾을 수 있다.

지금까지의 연구를 보면 많은 연구자들이 약 학습자로서 C4.5 (결정 트리 학습자) [6] [19] 그리고 결정 스텀프 [26]등을 사용하였다. 이러한 방법들이 모두 텍스트 분류에서 좋은 성능을 보여 주기는 하지만 앞서 언급한 바와 같이 이러한 약 학습자들은 대부분은 정보 검색 분야의 알고리즘이 이용하고 있는 단어의 가중치 정보나 문서의 길이 정보를 사용하지 않고 단어의 존재 여부만 가지고 문서를 분류하고 있다. 그러므로 약 학습자를 텍스트 환경에 맞추어 적절히 변경시켜 준다면 더 좋은 성능을 얻을 수 있을 것이다. 또 다른 문제는 결정 트리와 같은 알고리즘은 기본적으로 문서가 어느 범주에 속하는지는 알려주지만 확신도를 알려주지 않는다는 문제점이 있다. 이러한 여러가지 문제점을 해결하기 위하여 본 논문에서는 수정된 Naive Bayes 분류자를 약 학습자로 사용하고 있다.

3 절 Naive Bayes 분류자 (naive Bayes Classifier)

Naive Bayes 분류자는 잘 알려진 전통적인 분류방법으로서 텍스트 문서 분류에 성공적으로 사용되어 왔다 [12] [17]. Naive Bayes는 통계적인 알고리즘으로서 훈련 문서에서 여러 통계 정보를 학습한다. 그리고 이렇게 얻은 통계 정보를 이용하여 새로운 문서가 주어졌을 때 분류해 줄 수 있게 된다. d_i가 단어 $w_{i1}, w_{i2}, \cdots, w_{i|d_i|}$로 이루어졌다고 하자. 이 때 Naive Bayes는 문서에서 단어들의 분포는 서로 독립(mutually independent)임을 가정하고 있다. 또한 단어가 나타날 확률은 문서 내에서 단어의 위치와도 독립으로 가정한다. 이러한 가정은 실제 문서의 특성과는 많이 어긋나지만 이러한 문제에도 불구하고 Naive Bayes는 텍스트 문서 분류에서 비교적 좋은 성능을 보여주고 있다 [12] [17].

문서 d_i가 범주 c_j에서 생성되었을 확률은 다음과 같이 표현 될 수 있다.

$$P(d_i|c_j;\theta) \tag{3.1}$$

여기서 문서 내의 단어의 분포는 서로 독립이므로 위의 식은 다음과 같이 분해될 수 있다.

$$P(d_i|c_j;\theta) = P(|d_i|)\prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j;\theta)^{N(w_{d_{ik}},d_i)}, \tag{3.2}$$

여기서 $w_{d_{ik}}$는 k번 째 단어가 d_i번 째 문서에 나타날 확률을 의미하고, $N(w_{d_{ik}}, d_i)$는 단어 가중치 $w_{d_{ik}}$가 문서 d_i에 나타날 확률을 의미한다. 또한 $|d_i|$는 문서가 갖는 단어의 개수를 의미한다. 그러므로 우리가 추정해야 할 파라미터는

$$\theta_{w_k|c_j} = P(w_k|c_j;\theta)$$

로 표시될 수 있다. 여기서

$$\sum_{k=1}^{|V|} P(w_k|c_j;\theta) = 1$$

이다. 이 파라미터는 아래와 같이 추정될 수 있다.

$$\hat{\theta}_{w_k|c_j} = \frac{1+\sum_{i=1}^{|D|} N(w_k,d_i)P(c_j|d_i)}{|V|+\sum_{k=1}^{|V|}\sum_{i=1}^{|D|} N(w_k,d_i)P(c_j|d_i)} \tag{3.3}$$

범주가 나타날 사전 확률(prior knowledge)는 다음과 같이 표현될 수 있다.

$$\theta_{c_j} = P(c_j|\theta)$$

. 이 파라미터는 다음과 같이 추정될 수 있다.

$$\hat{\theta}_{c_j} = \frac{\sum_{i=1}^{|D|} P(c_j|d_i)+1}{|D|+2}, \tag{3.4}$$

여기서 $|D|$는 문서의 전체 수를 의미한다. 이의 식에서는 라플라시안 사전 지식(laplacian prior)를 사용하였는데 이는 훈련 데이터를 이용하여 추정된 확률값을 스무딩해주는 효과가 있다. 이는 훈련 문서만을 가지고 이러한 파라미터를 추정하는 것은 문서가 편향(biased)되어 있는 경우 오차를 발생시킬 수 있기 때문이다. 이러한 파라미터 값을 알고 있다고 하고 $P(|d_i|)$를 균등 분포(uniform distribution)을 따른다고 가정하면. 문서가 어떤 범주에 속할 확률은 다음과 같이 계산될 수 있다.

$$P(c_j|d_i;\hat{\theta}) = \frac{P(c_j|\hat{\theta})P(d_i|c_j;\hat{\theta})}{P(d_i|\hat{\theta})} \tag{3.5}$$

$$= \frac{P(c_j|\hat{\theta}) \prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j;\hat{\theta})^{N(w_{d_{ik}},d_i)}}{\sum_{r=1}^{|C|} P(c_r|\hat{\theta}) \prod_{k=1}^{d_i} P(w_{d_{ik}}|c_r;\hat{\theta})^{N(w_{d_{ik}},d_i)}}$$

우리는 여기서 $\arg\max_j P(c_j|d_i;\hat{\theta})$ 인 범주를 선택하면 된다.

Naive Bayes가 확률 모델이기 때문에 충분한 적합한 문서를 갖지 않는 경우에는 좋지 않은 성능을 보일 수도 있다. 우리가 실험해 본 결과 TREC-8에 대해서 실험해 본 결과 충분한 적합한 문서 데이터를 갖지 않는 경우는 좋지 않은 성능을 보였다. Naive Bayes는 또한 생성 모델(generative model)로서 해석될 수 있는데 이러한 생성 모델은 일반적으로 문서 분류의 결과를 해석하는 등의 해석적인 측면에서는 좋으나 성능 측면에서는 적합한 문서와 적합하지 않는 문서의 차이를 적극적으로 학습하는 변별 모델(discriminative model)과 비교하였을 때는 좋지 않은 성능을 보인다는 사실이 알려져 있다. 최근 들어 변별 모델의 장점을 갖는 생성 모델에 대한 연구가 진행중이지만 자세한 내용은 본 논문의 범위를 벗어나므로 생략한다. Naive Bayes의 장점 중 하나는 문서의 적합도 여부를 판정해 줄 뿐 아니라 확신도까지 출력해 준다는 점에 있다. 이러한 특징은 AdaBoost 알고리즘에서 약 학습자의 확신도를 계산할 때 사용될 수 있다.

4 절 BayesBoost: Boosted Naive Bayes

Naive Bayes는 단어의 가중치를 고려할 뿐만 아니라 확신도를 함께 출력해 주기 때문에 결정 스텀프나 결정 트리에 비해 AdaBoost의 약 학습자로 적합하다. 그러므로 본 논문에서는 Naive Bayes를 약 학습자로 사용하는 Boosting 알고리즘을 제안하였고 이를 BayesBoost라 명명하였다. BayesBoost알고리즘은 그림 3.1에 기술되어 있다.

$P(c_1|d_i;\hat{\theta})$ 를 문서 d_i가 적합하다고 판정될 확률이라고 하자. 마찬가지로 $P(c_0|d_i;\hat{\theta})$도 정의할 수 있다. 우리는 다음과 같은 식을 이용해서 $h_t(d_i;\hat{\theta})$ 약 가설의 출력을 계산

할 수 있다.

$$h_t(d_i;\hat{\theta}) = P(c_1|d_i;\hat{\theta}) - P(c_0|d_i;\hat{\theta}) \tag{3.6}$$

이는 두 범주에 대한 확률간의 차이를 의미하고 [-1,1]의 범위를 갖게 된다. 일반적으로 Boosting에는 문서의 가중치를 조정하면서 학습하는 방법과 문서를 다시 샘플링하여 새로운 문서 분포를 만드는 방법이 있다. 본 논문에서는 문서의 가중치를 조정하는 방법을 사용하였다. 이를 위해서는 Naive Bayes 알고리즘을 다음과 같이 약간 수정해야 한다.

$$\hat{\theta}_{w_k|c_j} = \frac{1/|D| + \sum_{i=1}^{|D|} D(i)N(w_k,d_i)P(c_j|d_i)}{|V|/|D| + \sum_{k=1}^{|V|}\sum_{i=1}^{|D|} D(i)N(w_k,d_i)P(c_j|d_i)} \tag{3.7}$$

$$\hat{\theta}_{c_j} = \sum_{i=1}^{|D|} D(i)P(c_j|d_i), \tag{3.8}$$

텍스트 문서 여과 문제의 특징 중 하나는 적합한 문서를 추출하지 못했을 때의 비용이 적합하지 않은 문서를 추출하였을 때의 비용보다 더 크다는 점이다. 많은 문서 여과 측정 방법들이 이러한 점을 고려하여 제작되었다. 예를 들어 TREC-8에서 사용되는 선형 유틸리티(linear utility)가 그러한 예이다. 선형 유틸리티는 다음 장에서 더욱 자세히 설명하고 있다. 이러한 점을 감안하여 AdaBoost 알고리즘을 약간 수정함으로서 BayesBoost알고리즘의 성능을 더욱 높일 수 있다. 한가지 방법은 문서의 초기 중요도를 문서의 적합성 여부에 따라 다르게 매기는 방법이다. 이러한 방법은 적합한 문서에 더 집중하여 학습함으로서 적합한 문서가 잘 못 분류될 확률을 줄여주는 효과가 있다 [26]. 이를 위해서는 문서의 중요도를 다음과 같이 초기화 시켜 준다.

$$D_1(i) = \begin{cases} \frac{a-c}{Z_0} & \text{if } y_i = +1 \\ \\ \frac{d-b}{Z_0} & \text{if } y_i = -1, \end{cases} \tag{3.9}$$

여기서 Z_0은 정규화 요소(Normalization factor)이며 a, b, c, 그리고 d는 식 4.1에 나온 바와 같다. 그러나 이러한 방법은 일종의 휴리스틱으로서 더 정교한 방법은

AdaBoost 알고리즘은 목적함수를 수정하는 것이다. 기본적으로 AdaBoost 알고리즘은 정확도를 높이는 것(혹은 에러를 줄이는 것)을 목적으로 하고 있다. 반면 우리가 최대화하여야 할 값은 선형 유틸리티(linear utility)값이다 [7]. 목적함수를 수정하면 다음과 같은 새로운 Boosting 알고리즘이 생성되게 된다. 먼저 앞서와 같이 문서의 중요도를 초기화 시킨다. 그러나 이에 더물어 문서의 중요도를 갱신하는 식도 다음과 같이 수정해야 한다.

$$D_{t+1}(i) = \frac{D_t(i)exp\left(-\alpha_t y_i h_t(d_i)\beta(i)\right)}{Z_t}$$

여기서 $\beta(i)$ 는 비용 조정 함수(cost-adjustment function)이며, Z_t 는 정규화 요소이다. False-negative(적합하지 않다고 판단된 적합한 문서)에 대해서는 $\beta(i)$는 a가 된다. 반면 false-positives (적합하다고 판단되었는데 적합하지 않은 문서)에 대해서는 $\beta(i)$는 b가 된다. 그 외의 경우에는 $\beta(i)$는 b로 고정하였다. 여기서 a,b 는 식 4.1에 나온 바와 같다. 이러한 경우에 α_t는 다음과 같이 계산된다.

$$\alpha_t = \frac{1}{2}\log(\frac{1+r}{1-r})$$

여기서

$$r = \sum_{i=1}^{|D|} D_t(i)y_i h_t(d_i;\theta)\beta(i)$$

본 논문에서는 후자의 방법을 사용하였다.

5 절 문서 길이를 고려한 BayesBoost 알고리즘

문서 여과에서 고려하여야 할 중요한 특질 중의 하나는 문서의 길이이다. 많은 연구자들이 문서의 길이와 관련된 문제를 해결하기 위해 노력하였고, 이 중 성공적인 것은 벡터 스페이스 모델에서는 Singhal의 Document length normalization [28]과 확률 모델에서는 Robertson의 연구 등이 있다. 본 연구에서는 이러한 문서 길이 정보를 Naive Bayes 알고리즘에 삽입하고자 한다. 알고리즘을 살펴 보기 전에 먼저 짧은 문서를 이용하여 문서를 분류할 경우 발생할 수 있는 문제에 대해 살펴 보자. 예를

들어 문서 길이가 1인 'the'라는 문서를 생각해 보자. Naive Bayes는 각 범주에서 문서가 나올 확률을 비교하여 이 중 최고값을 갖는 범주를 선택하고 이 때 각 문서가 나올 확률은 각 단어가 나올 확률의 곱으로 표현된다는 사실을 기억하자. 위의 경우에는 길이가 1이므로 이 경우는 문서가 나올 확률이 곧 단어 'the'가 나올 확률과 일치하게 된다. 만일 Naive Bayes를 학습시킨 결과 우연히 적합한 범주에서 'the'가 나올 확률이 적합하지 않은 범주에서 'the'가 나올 확률보다 더 크다면 이 문서는 적합한 문서로 분류 되게 된다. 그러나 실제로는 'the'라는 문서는 대부분의 경우 어떤 질의에도 부합하지 않을 것이다. 이러한 문제는 Naive Bayes에서 문서를 분류할 때 적은 수의 통계치만을 이용하기 때문에 발생한다. 예를 들어 위의 경우 하나의 통계치만을 이용하였다. 또 하나 유의할 점은 문서의 길이가 너무 짧은 경우 대부분 적합하지 않은 문서일 가능성이 크다는 점이다. 이는 다음과 같은 가설에 의해 추론될 수 있다. 문서의 길이가 너무 짧은 경우 매우 협소한 내용을 다루는 질의문에만 적합하게 된다. 반면 긴 경우는 더 많은 주제를 다루기 때문에 더 많은 질의문에 적합할 것이다.

이러한 점을 감안하여 알고리즘에 적용하면 문서 길이가 짧을 수록 적합하지 않은 문서로 분류될 확률을 높게 해 주는 것이다. 이러한 일을 수행하기 위해 우리는 문서 길이에 대한 기존의 균등 분포 가정을 없애고 대신 식 5에 나온 바와 같이 가우시안 분포(gaussian distribution)를 따른다고 가정하였다. 이렇게 하였을 경우 적합한 문서의 평균 길이는 적합하지 않은 문서의 평균 길이보다 더 크기 때문에 이는 문서의 길이가 짧은 경우 균등 분포 가정을 사용한 Naive Bayes에 비해 추출될 확률을 낮추어 주는 효과가 있다. 그러나 가우시안을 따른다고 가정하였을 경우 문서의 길이가 매우 긴 경우는 역으로 문서가 적합하다고 잘 못 분류될 확률이 더 커지게 된다. 이러한 점을 방지하기 위해 우리는 200단어 이상의 문서에 대해서는 균등 분포를 가정하였다. 가우시안 분포의 식은 다음과 같다.

$$P(|d_i|) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(|d_i|-|\bar{d}_i|)^2/\sigma^2}$$

여기서 $|\bar{d}_i|$는 문서의 평균 길이를 의미한다. 이상적으로 우리가 $P(c_j|d_i;\hat{\theta})$를 계산

할 때, $|\bar{d}_i|$는 c_j값에 따라 적합하거나 적합하지 않은 문서들의 평균 길이로 정해진다. 그러나 현실적으로는 이러한 질의문에 해당하는 문서들이 조금씩 밖에 없기 때문에 추정에 오류가 일어날 가능성이 매우 크다. 이러한 문제를 해결하기 위해서는 일종의 스무딩작업이 필요한데 이를 위해 본 연구에서는 전체 질의문에 대해 평균을 취하는 방법을 사용하였다.

그림 3.1: BayesBoost 알고리즘

1. Given: $(d_1, y_1), ..., (d_m, y_m)$,
   where $d_i \in X, y_i \in \{-1, +1\}$.

2. Initialize a distribution using Equation (3.9).

3. For $t = 1, \cdots, T$:

   - Train a naive Bayes classifier and get a weak hypothesis h_t by Equation (3.6), where $\hat{\theta} = (\hat{\theta}_{w_k|c_j}, \hat{\theta}_{c_j})$ is computed by Equations (3.7) and (3.8)
   - Choose $\alpha_t = \frac{1}{2}\log(\frac{1+r}{1-r})$, where

   $$r = \sum_{i=1}^{|D|} D_t(i) y_i h_t(d_i; \theta) \beta(i).$$

   - Update the distribution:

   $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(d_i; \hat{\theta})\beta(i))}{Z_t},$$

   where Z_t is a normalization factor.

4. Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(d_i; \hat{\theta})\right).$$

4 장

실험 및 결과

1 절 성능 측정 방법

지금까지의 텍스트 문서 여과에 대한 연구를 통해 시스템의 성능을 평가하기 위한 다양한 방법을 연구해 왔다. 이러한 연구를 통해 제안된 층적법은 precision, recall등이 있다. 그러나 precision과 recall은 서로 따로 떨어져서 생각할 수 없으므로 이 둘을 결합하여 하나의 수치로 표현한 break-even point [15]도 있다. 그러나 Singhal et al.이 지적한 바와 같이, break-even point 는 필터링 알고리즘의 성능을 측정하는데는 썩 좋은 방법이 아니다 [26]. 최근에 TREC 학회에서는 필터링 트랙의 성능을 측정하기 위해서 선형 유틸리티(linear utility)를 새로운 측정 방법으로 사용하고 있는 추세이다 [10]. 그러므로 본 연구에서도 전통적인 break-even point 대신 선형 유틸리티를 측정치로 사용하였다.

R_+ 를 적합하고 추출된 문서의 개수라고 놓고 N_+ 를 적합하지 않지만 추출된 문서의 개수로 놓자. 마찬가지로 R_- 를 적합하지만 추출되지 않은 문서의 개수로 N_-를 적합하지도 않고 추출되지도 않은 문서의 개수로 놓는다. 그러면 선형 유틸리티는 다음과 같이 정의된다.

$$\text{Linear utility} = aR_+ + bN_+ + cR_- + dN_- \tag{4.1}$$

여기서 a, b, c, 그리고 d 는 상수 계수이다.

TREC-8에서는 , LF1과 LF2 가 사용되었는데 이는 다음과 같이 정의된다.

$$\text{LF1} = \text{F1} = 3R_+ - 2N_+ \tag{4.2}$$

$$\text{LF2} = 3R_+ - N_+. \tag{4.3}$$

TREC-7에서는 F1과 F3이 사용되었는데 F1 은 TREC-8의 LF1과 같고 F3은 다음과 같이 정의된다.

$$\text{F3} = 4R_+ - N_+. \tag{4.4}$$

선형 유틸리티에 대해서 유의해야 할 점 중 하나는 각 질의문(topic)에 대해 얻은 선형 유틸리티를 평균하는 것은 의미가 없다는 점이다. 그렇게 하면 선형 유틸리티의 값이 매우 큰 몇몇이 평균 점수를 좌우하기 때문이다. 그러므로 시스템의 전반적인 성능을 효율적으로 측정하고자 할 경우에는 스케일드 선형 유틸리티(scaled linear utility)를 사용하여야 한다 [10].

$$\text{Scaled linear utility} = \frac{\max\{u(S,T), U(s)\} - U(s)}{MaxU(T) - U(s)}$$

여기서 $u(S,T)$ 는 시스템 S에서 질의문(topic) T에 대한 선형 유틸리티값을 의미하며, $MaxU(T)$는 질의문 T에 대해 얻을 수 있는 가장 높은 선형 유틸리티 값을 의미한다. $MaxU(T)$는 문서 여과의 대상이 되는 코퍼스에서 질의문에 적합한 문서의 개수에 의해 결정되게 된다. $U(s)$ 는 s개의 적합하지 않은 문서를 추출하였을 경우의 선형 유틸리티를 의미한다.

2 절 TREC-7 과 TREC-8 필터링 트랙 데이터셋

트렉-7 필터링 트랙의 데이터셋은 1988-1990년동안의 AP기사와 1-50번까지의 토픽, 1988년의 AP 문서는 훈련 집합으로서 사용되고 1989-1990년까지의 문서는 테스트 셋으로 사용된다. 트랙 8은 1992-1994년까지의 파이낸셜 타임스(FT)기사로

그림 4.1: TREC 배치 필터링 트랙에서 사용되는 문서의 예

```
<DOC>
<DOCNO> AP880213-0001</DOCNO>
<FILEID>AP-NR-02-12-88 2344EST</FILEID>
<FIRST> u i AM-Vietnam-Amnesty 02-12 0398</FIRST>
<SECOND>AM-Vietnam-Amnesty, 0411</SECOND>
<HEAD>Reports Former Saigon Officials Released from Re-education Camp</HEAD>
<DATELINE>BANGKOK, Thailand(AP)</DATELINE>
<TEXT>
More than 150 former officers of the overthrown
South Vietnamese government have been released
from a re-education camp after 13 years of detention
the official Vietnam News Agency reported.
<중 간 생 략>
</TEXT>
</DOC>
```

이루어져 있고 351-400번까지의 질의문으로 이루어져 있다. 여기서 1992년도의 FT 문서는 훈련 데이터로서 사용되고 1993-1994년도의 문서는 테스트 집합으로서 사용된다. 이는 각각 TREC-7 과 TREC-8 배치 필터링 트랙에서의 환경과 정확히 일치한다. 트랙에서 사용되는 문서와 질의문의 예는 그림 4.1과 4.2 에서 확인할 수 있다. 일반적으로 질의문도 학습 대상에 포함시키지만 본 실험에서는 질의문은 학습 대상에 포함시키지 않았다. 반면 TREC에 참가한 다른 시스템은 대부분 질의문을 학습 대상에 포함시켰음을 밝힌다.

그림 4.2: TREC 배치 필터링 트랙에서 사용되는 질의문의 예

```
<top>
<num> Number: 351
<title> Falkland petroleum exploration

<desc> Description
What information is available on petroleum exploration
in the South Atlantic near the Falkland Islands?

<narr> Narrative:
Any document discussing petroleum exploration in the
South Atlantic near the Falkland Islands is considered
relevant. Documents discussing petroleum exploration in
continental South America are not relevant.
</top>
```

3 절 실험 설계 및 결과

전처리를 위하여 먼서 문서를 포터(Porter)의 알고리즘을 이용하여 스테밍(stemming)하고 스탑 리스트(stop-list)나 길이나 너무 짧은 단어들을 제거하였다. 단어의 가중치는 벡터 스페이스 모델에서 흔히 사용되는 표준 $tf \cdot idf$ 가중치를 사용하였다. 이 외에 시소러스(thesaurus)나 다른 데이터셋은 전혀 사용하지 않았다. 전체 데이터 셋에 있는 단어의 총 개수는 160,000개도 넘지만 이들을 모두 다 사용하지 않고 TREC-7 데이터 셋에서는 5251개의 단어만을 사용하였고 TREC-8에 대해서는 4071개의 단어만 사용하였다. 이 때 사용한 기준은 문서 빈도이다. 예를 들어 트랙-7의 경우에는 문서 빈도가 큰 순서대로 정렬하여 상위 5251개의 단어만을 추출하였다.

각각의 질의문에 대해서 Boosting 알고리즘을 실행시킬 때 LF1, F1 의 경우 40개, F3, LF2의 경우 100개의 가설이 만들어지거나 훈련 에러가 0이 될 때까지 훈련을 진행하였다. 비교의 목적으로 그림 4.3과 4.4에 각각 TREC-7 과 TREC-8에 제출되었던 결과 중 상위 세 개만을 같이 표시하였다. 여기서 x축은 훈련 문서에서의 적합한 문서(Relevant documents)의 개수에 따라 오름차순으로 정렬되었다. y축은 해당 질의문에 대한 BayesBoost의 결과에서 다른 비교 대상이 되는 시스템의 차이를 나타내었다. 그러므로 y축의 값이 양수 있는 것이 많을 수록 BayesBoost가 성능이 더 좋음을 의미한다. 성능은 선형 유틸리티로 측정되었는데 이 때 100개의 적합하지 않은 문서를 추출하였을 경우에는 0이 되도록 설정하였다. 그러나 주의할 점은 이미 제출되었던 결과들과 본 알고리즘의 결과의 정확한 비교는 불가능하다는 점이다. 이는 트렉에서 사용하고 있는 트렉 풀링(Pooling)이라는 기술과 밀접한 관련이 있다. 여기서 잠시 트렉 풀링에 대해서 설명하고 넘어가기로 한다.

트렉에서는 사용자가 결과를 제출하면 이를 모은 후 각각의 제출 결과에 대해서 n개의 문서를 임의로 추출한다. 이렇게 해서 추출된 문서들을 모아서 이들만을 적합도 평가를 하며 여기에 참가하지 못한 문서는 적합하지 않다고 가정한다. 이러한 과정을 거치는 이유는 트렉에서 많은 사용자들이 제출한 결과를 모두 평가할 수 없기 때문이다. 트렉 풀링 과정은 비록 각각의 시스템의 절대적인 성능을 평가하긴

그림 4.3: BayesBoost와 TREC-7에 제출되었던 결과 중 상위 세 개와 비교

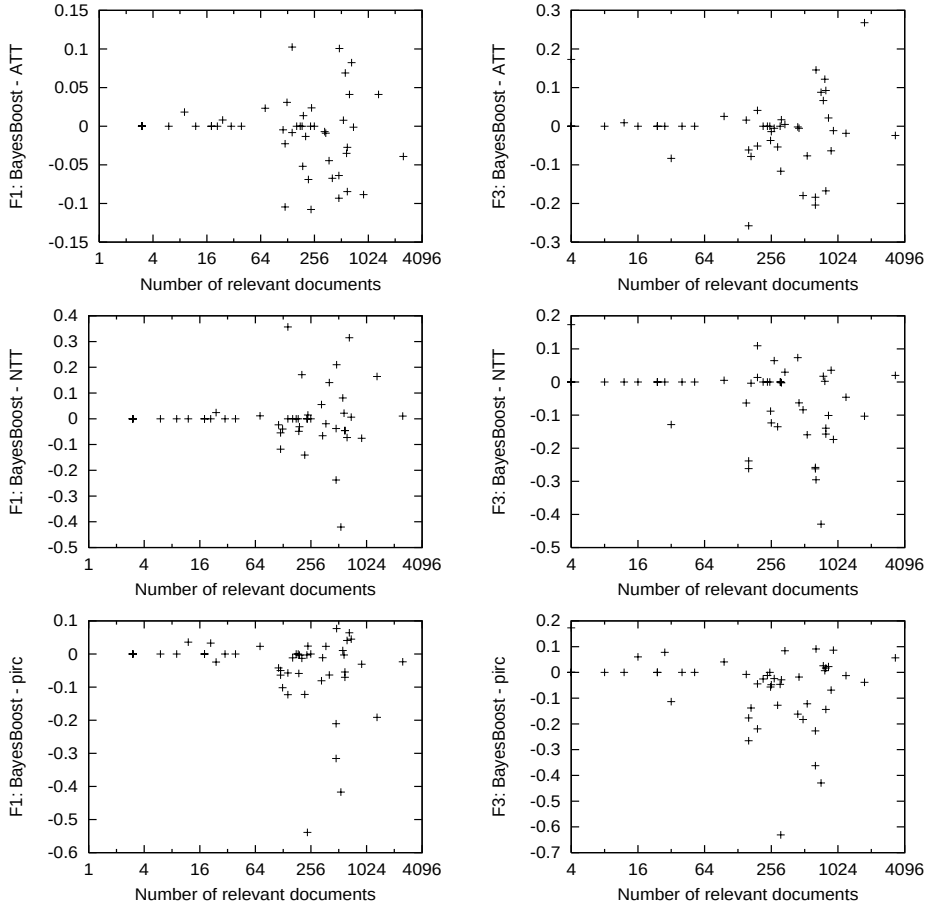
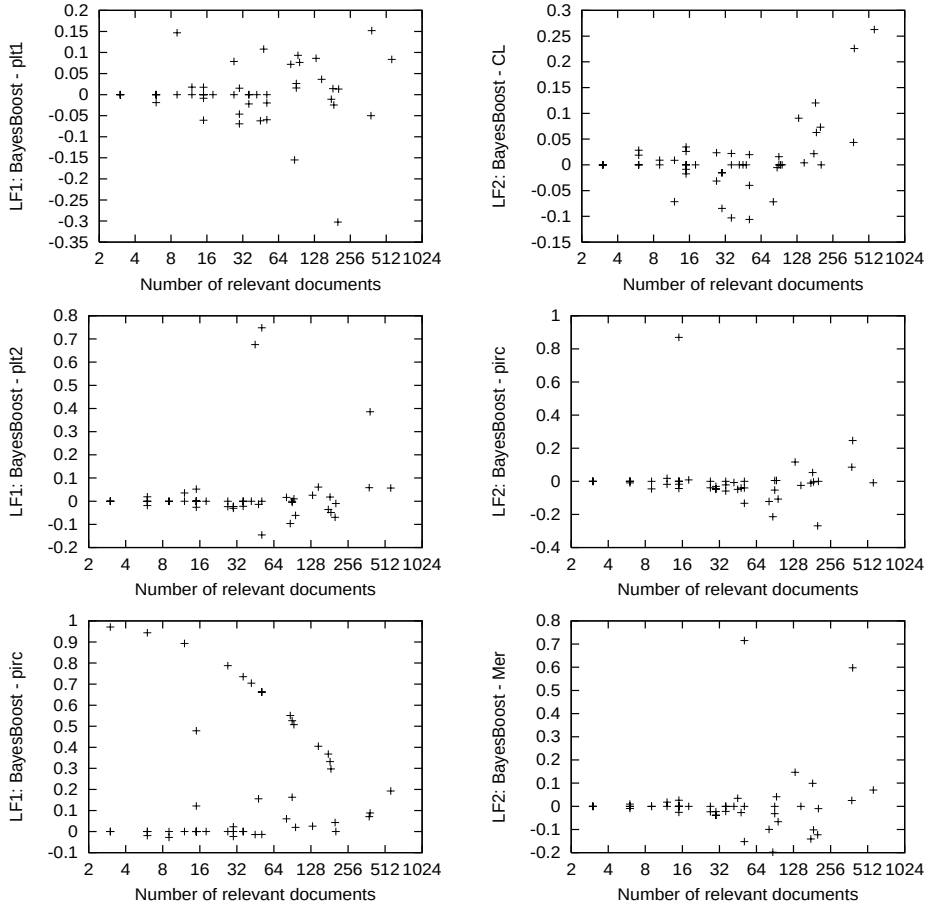


그림 4.4: BayesBoost와 TREC-8에 제출되었던 결과 중 상위 세 개와 비교



어렵지만 상대적인 성능은 평가하기 용이하다는 장점이 있다. 불행히도 우리가 진행한 실험 결과는 트렉 풀링에 참가하고 있지 않기 때문에 다른 실험 결과와 동일한 비교가 불가능하다 [30]. 그러므로 우리는 두 가지 환경에서 테스트를 하였는데 한 가지 방법은 테스트 문서에 있는 모든 평가되지 않은 문서들을 무시하는 것이다. 이러한 세팅에서는 우리 시스템이 다른 시스템에 비해 더 유리한 위치에 있게 된다. 다른 방법으로는 모든 평가되지 않은 문서들은 적합하지 않은 문서로서 간주하는 것이다. 이러한 환경에서은 우리의 시스템이 다른 시스템에 비해 약간 더 불리한 위치에 있게 된다 [10].

그림에 대한 더 자세한 분석은 TREC-7, TREC-8에 대해서 각각 테이블 4.1, 4.2과 테이블 4.3, 4.4를 참조하면 된다. 이 표에 나온 다른 결과들은 TREC-7과 TREC-8 배치 필터링 트랙에 참가했던 시스템 중 상위 세 개의 시스템에 대한 실험 결과로서 각각 TREC-7과 TREC-8에서 발췌한 내용이다. 아래의 'asl'은 평균 스케일드 선형 유틸리티를 나타내고 있다. BayesBoost알고리즘의 결과 중 괄호 안에 있는 내용은 평가되지 않은 문서들(unjudged documents)를 무시한 경우의 결과를 나타낸다. 표에서 확인할 수 있듯이 BayesBoost가 다른 시스템보다 더 불리한 조건에서 평가되고 있음에도 불구하고 좋은 성능을 보여주고 있음을 알 수 있다. 예를 들어 FT의 경우에는 LF1에 대해서는 1위와 동점을 이루고 있으며 LF2의 경우에도 1위와 근소한 차이만 보이고 있음을 알 수 있다. AP에 대해서도 F1의 경우는 3위보다 더 좋은 성능을 보이고 있다. 이 결과에 대한 더 자세한 분석은 1장으로 미룬다.

우리는 BayesBoost알고리즘을 결정 스텀프를 약 학습자로 사용하는 AdaBoost 알고리즘과도 비교해 보았다. 실험 결과는 TREC-7에 대해서는 그림 4.5(왼쪽)에 있고 TREC-8에 대해서는 그림 4.5(오른쪽)에 기술되어 있다. 그림에서 알 수 있듯이 BayesBoost는 AdaBoost 보다 훨씬 더 좋은 성능을 보이고 있다. 그림에서 더 많은 적합한 문서가 훈련 데이터에 포함되어 있을 수록 BayesBoost 알고리즘이 더 좋은 성능을 보인다는 사실도 유의할 만하다.

표 4.1: TREC-7 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (1)

	F1				F3			
	BayesBoost	ATT	NTT	pirc	BayesBoost	ATT	NTT	pirc
1	150(196)	189	-91	101	323(368)	386	288	391
2	-27(17)	-8	5	11	52(106)	-31	193	37
3	82(102)	136	220	204	165(178)	316	359	333
4	3(3)	-1	14	4	19(19)	24	63	36
5	24(24)	47	50	38	12(12)	79	80	81
6	134(136)	193	166	183	235(236)	385	360	364
7	1(53)	25	-14	3	237(303)	130	235	231
8	-3(-1)	-2	2	6	5(7)	1	21	7
9	8(10)	0	3	0	7(10)	0	8	19
10	94(120)	89	363	361	224(260)	152	576	576
11	41(147)	130	117	72	397(520)	421	457	413
12	145(591)	86	-90	419	716(1020)	213	910	788
13	0(0)3	36	0	180	0(0)	48	0	260
14	-33(-7)	-11	12	6	26(54)	47	79	76
15	-13(-13)	-38	-143	1	-7(-7)	-19	-39	6
16	-3(-3)	1	26	2	8(8)	11	43	18
17	76(104)	110	5	108	198(226)	247	300	276
18	-4(-4)	0	-56	0	-23(-15)	-21	-47	-14
19	30(30)	0	83	0	24(24)	4	119	3
20	14(14)	17	-10	49	45(45)	46	5	133
21	3(3)	2	0	6	4(4)	15	21	19
22	683(1353)	786	655	745	2235(1724)	1807	1655	1528
23	170(170)	207	192	353	255(255)	391	446	523
24	124(166)	78	70	117	368(417)	311	363	346
25	2(6)	-5	11	25	21(22)	42	22	58
26	0(0)	0	0	3	0(0)	0	0	8

표 4.2: TREC-7 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (2)

	F1				F3			
27	0(0)	0	0	0	0(0)	0	0	0
28	0(0)	0	0	0	0(0)	0	0	4
29	0(0)	-2	0	0	0(0)	-1	0	0
30	0(0)	0	0	0	0(0)	0	0	0
31	0(0)	0	0	0	0(0)	0	0	0
32	0(0)	0	0	0	0(0)	0	0	0
33	0(0)	0	0	0	0(0)	0	0	0
34	0(0)	0	0	0	0(0)	0	0	0
35	0(0)	0	0	0	0(0)	0	0	0
36	0(0)	0	0	0	0(0)	0	0	0
37	0(0)	0	0	-4	0(0)	0	0	-10
38	49(85)	50	44	13	184(210)	196	363	95
39	0(0)	0	0	-4	0(0)	0	0	-7
40	62(74)	3	-61	17	142(149)	33	363	74
41	0(0)	0	0	0	0(0)	0	0	0
42	-107(-99)	-116	-139	-129	-31(-27)	-33	-44	-68
43	0(0)	0	0	1	0(0)	0	0	19
44	54(56)	75	63	43	39(39)	146	89	148
45	-147(-139)	-175	-167	-178	-82(-83)	-143	-123	-136
46	-2(-2)	0	-2	28	3(3)	18	-1	67
47	-12(0)	3	2	2	-6(0)	7	25	14
48	0(0)	-4	-2	-4	0(0)	-5	-1	-8
49	-3(-3)	2	9	8	5(5)	21	67	51
50	0(0)	0	0	0	0(0)	0	0	0
asl	0.454 (0.481)	0.461	0.452	0.449	0.467 (0.467)	0.460	0.505	0.509

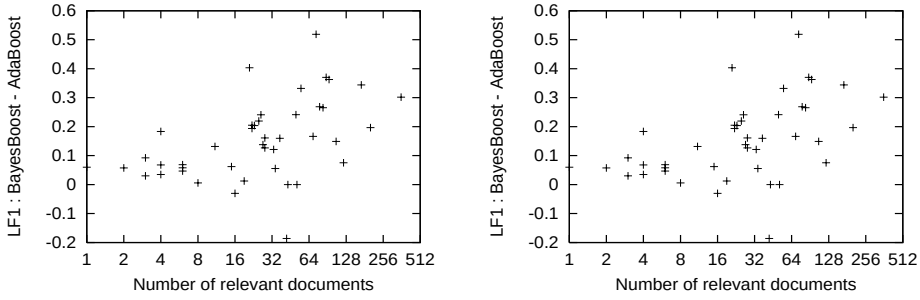
표 4.3: TREC-8 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (1)

	LF1				LF2			
	BayesBoost	plt1	plt2	pirc	BayesBoost	CL	pirc	Mer
351	13(13)	22	25	25	8(8)	24	28	-136
352	138(142)	82	115	125	232(235)	56	238	185
353	3(3)	32	42	28	4(4)	5	44	41
354	0(0)	-9	-9	-7	0(0)	-1	6	0
355	0(0)	0	0	0	0(0)	0	0	0
356	-2(0)	7	7	4	-1(0)	-1	6	-6
357	51(53)	47	41	46	84(86)	50	69	56
358	0(0)	0	0	0	0(0)	0	0	0
359	0(0)	-5	-5	0	0(0)	0	-1	0
360	0(0)	0	0	0	0(0)	0	3	3
361	0(0)	0	0	0	0(0)	-3	0	0
362	0(0)	0	0	0	0(0)	2	0	0
363	0(0)	0	0	-4	0(0)	-1	-2	-2
364	0(0)	0	0	0	0(0)	0	0	-1
365	0(0)	-16	-28	4	0(0)	-1	5	0
366	0(0)	0	0	3	0(0)	6	6	23
367	3(3)	0	5	4	3(4)	0	13	9
368	0(0)	0	0	0	0(0)	0	0	0
369	0(0)	0	0	0	0(0)	0	0	0
370	-2(-2)	-20	-20	-13	-1(-1)	-1	-2	-9
371	0(0)	0	0	0	0(0)	0	0	0
372	0(0)	3	3	3	0(0)	-3	8	0
373	0(0)	0	0	0	0(0)	14	0	0
374	2(2)	5	5	-19	3(3)	-3	6	42
375	0(0)	-2	1	3	0(0)	2	6	5
376	0(0)	3	3	0	0(0)	-3	0	0

표 4.4: TREC-8 배치 필터링 데이터셋에 대한 BayesBoost알고리즘의 결과 (2)

	LF1				LF2			
	BayesBoost	plt1	plt2	pirc	BayesBoost	CL	pirc	Mer
377	0(0)	6	6	2	3(4)	11	6	3
378	-5(-5)	-25	-14	-12	15(17)	-6	-12	-19
379	0(0)	0	0	0	0(0)	-4	0	0
380	0(0)	0	0	0	0(0)	0	0	0
381	-2(0)	0	0	0	-1(0)	-3	0	0
382	0(0)	-1	-2	-2	0(0)	8	2	0
383	6(12)	97	113	72	22(25)	0	103	59
384	0(0)	0	0	0	0(0)	0	0	0
385	22(22)	9	14	28	13(14)	26	35	31
386	0(0)	0	0	-2	0(0)	0	-1	0
387	0(0)	-10	-10	-1	0(0)	-3	0	3
388	0(0)	0	0	0	0(0)	0	1	0
389	219(229)	145	113	76	323(330)	213	203	32
390	0(0)	-4	-4	0	0(0)	0	0	3
391	12(30)	36	-18	-35	80(98)	59	39	68
392	-17(-1)	-32	-32	2	2(10)	2	23	15
393	0(0)	7	7	1	0(0)	1	5	3
394	0(0)	0	0	0	0(0)	-4	0	0
395	-15(-15)	-8	-1	-50	15(16)	-3	16	44
396	0(0)	1	1	-1	0(0)	-3	2	0
397	0(0)	-2	-2	0	0(0)	0	0	-3
398	0(0)	9	9	0	0(0)	2	4	5
399	0(0)	0	0	3	0(0)	4	5	0
400	3(3)	-13	-8	-9	5(6)	5	11	9
asl	0.715 (0.719)	0.712	0.713	0.714	0.733 (0.737)	0.721	0.734	0.720

그림 4.5: BayesBoost와 AdaBoost의 성능 비교



5 장

분석

1 절 다른 시스템과의 비교 분석

그림 4.3과 그림 4.4, 그리고 테이블 4.1, 4.2, 4.3, 4.4에서 보인 결과에 의하면, BayesBoost알고리즘이 TREC-7과 TREC-8데이타에 대해 다른 시스템의 결과와 뒤지지 않는 성능을 보임을 확인할 수 있다. 전에 언급했듯이 우리의 실험 결과는 다른 결과보다 불리한 조건에서 실험한 것이다. 그러나 그러한 불이익에도 불구하고 결과는 TREC에서 좋은 성능을 보인 시스템의 결과보다 더 좋은 경우도 있었다. 그림 4.3 과 그림 4.4이 보여주듯이, 또 하나 주목할 점은 Schapire et al. 의 AdaBoost 실험에서는 LF2에 대해 좋지 않은 성능을 보였지만 본 실험에서는 LF2에 대해서도 좋은 성능을 보여 주었다는 점이다. [26]. 앞서 Shapire는 Ada브스트가 Rocchio 알고리즘과 비교할 때 LF1 측정법에 대해서는 매우 좋은 성능을 보이지만 LF2에 대해서는 그다지 좋지 않은 성능을 보임을 밝힌 바 있다. 이는 앞서도 설명하였듯이 AdaBoost 알고리즘이 선형 유틸리티에 대해 최적화되어 있지 않기 때문이다.

이제 비교 대상이 되는 다른 시스템에 대해서 잠시 살펴보고 넘어가기로 하자. 먼저 'pirc'는 여섯 개의 선형 함수를 만들어서 유전 알고리즘을 이용하여 함수의 파라미터를 결정하였다 [13]. 이렇게 해서 만들어진 함수들은 위원회를 구성한 뒤 새

로운 문서가 들어오면 투표를 통해 문서의 적합성을 판단하게 된다. AT&T (ATT)는 2단계 Rocchio 알고리즘(2-pass Rocchio algorithm)을 이용하였다. 이는 앞서 설명한 Query zoning기법의 다른 이름에 다름아닌데 첫번째 단계은 적합도 정보가 있는 문서만을 대상으로 학습을 학고 두번째 단계는 적합도 정보가 있는 문서에 더하여 적합도 상위 5000개의 문서를 이용하여 Rocchio 식을 학습하였다. 이 때 적합도 정보가 없는 문서는 모두 적합하지 않다고 가정하였다 [29]. 이러한 방법은 분류가 잘 되지 않는 부분에 집중한다는 점에서 Boosting과도 관련이 있다. 그러나 Boosting은 이론적으로도 근거가 있고 사용자가 정해 주어야 할 파라미터가 적다는 점에서는 더 유용하다고 하겠다. NTT도 마찬가지로 2단계 Rocchio 알고리즘을 사용하였다. 그러나 AT&T와는 달리 이에 더하여 Dynamic Feedback Optimization (DFO)와 같은 기술을 추가로 사용하였다. 실험에서 보듯이 이러한 기술은 AT&T에 비해서 그렇게 큰 성능 향상을 보인것 같지는 않았다.

BayesBoost 는 더 많은 적합한 문서가 주어질 수록 다른 시스템보다 더 좋은 성능을 보여주고 있다. 반면 적합한 문서가 적은 경우 다른 예보다 약간 떨어지는 성능을 보여주고 있다. 이는 Naive Bayes를 Boosting한 알고리즘의 복잡도(complexity)가 Rocchio와 같은 선형 함수의 복잡도보다 더 크기 때문인 것으로 추정된다. 통계 학습 이론에 따르면 데이터가 적은 경우는 오히려 복잡도가 더 적은 알고리즘의 성능이 더 좋게 나올 수 있기 때문이다.

마지막으로 False-positive와 False-negative의 비용의 차이가 큰 경우에 발생할 수 있는 문제에 대해 잠시 언급하고 넘어가기로 한다. 두 값이 차이가 큰 경우 False-positive에 대한 β값이 작아지게 되고 이는 곧 BayesBoost에서 새로운 분포를 생성해 낼 때 새로운 분포가 기존의 분포와 그다지 차이가 없도록 해주는 효과아 있다. 즉 β가 신경회로망에서의 학습률(learning rate)과 같이 작용하는 것이다. 이러한 원치 않는 효과 때문에 LF2나 F3에 대해서 학습하는 경우 LF1에 대해 학습하는 것보다 더 많은 시간을 소모하게 된다. 본 실험에서도 LF1에 대해서는 40개의 양 학습자만 생성해도 대부분의 경우 에러가 0으로 떨어지는 반면 F3의 경우는 100개 정

표 5.1: 가우시안 분포 가정과 균등 분포 가정에 대한 실험

$P(d_i)$	LF1		LF2	
	AP	FT	AP	FT
gaussian	0.480(0.454)	0.719(0.715)	0.467(0.446)	0.722(0.714)
uniform	0.470(0.427)	0.712(0.682)	0.464(0.436)	0.718(0.682)

도의 학습자를 결합하여도 에러가 0으로 떨어지지 않는 경우가 많았다. TREC-7에 대해 실험하였을 때 F3에 대해서 좋지 않은 결과를 보인 것도 훈련 데이터에 대해 학습이 제대로 되지 않았기 때문으로 추정하고 있다. 이러한 문제점에 대한 해결은 추후의 과제이다.

2 절 문서 길이를 고려하였을 때의 효과 분석

문서 길이를 고려하였을 때의 효과를 분석하여 보기 위해서 문서 길이에 대한 기존과 같은 균등 분포 가정을 취한 Naive Bayes 분류자와 가우시안 가정을 취한 Naive Bayes 분류자를 서로 비교하여 보았다. 테이블 5.1에서 보여진 바와 같이 가우시안 분포를 가정한 방법은 문서 길이를 고려하지 않는 것에 비해 많은 성능 향상을 보임을 알 수 있다.

6 장

결론

TREC-7과 TREC-8 배치 필터링 트랙에 대한 실험 결과는 BayesBoost알고리즘이 트렉에 참가한 다른 시스템과 경쟁력이 있음을 보여주었다. 특히 LF1, LF2, F1에 대한 실험 결과는 각 트렉에서 제일 좋은 성능을 보인 시스템과 비슷한 성능을 보여주었다. BayesBoost의 결과에 약간의 불이익이 있다는 점을 감안하면 이러한 결과는 매우 고무적으로 볼 수 있다. F3에 대해서는 그다지 좋지 않은 성능을 보였는데 이는 앞서도 언급하였듯이 훈련이 충분히 이루어지지 못한 것으로 추정된다.
또한 BayesBoost 와 결정 트리를 사용한 AdaBoost 알고리즘과의 비교 실험에서도 BayesBoost알고리즘이 AdaBoost알고리즘보다 훨씬 더 좋은 성능을 보인다는 사실을 밝혔다. 이러한 점은 AdaBoost에비해 BayesBoost알고리즘이 단어의 가중치 정보나 문서의 길이 정보 등 텍스트 데이터의 특성을 충분히 활용할 수 있기 때문이다. 반면 AdaBoost 알고리즘은 단순히 단어의 존재 여부만을 가지고 문서를 분류하였기 때문에 본질적인 한계를 가질 수 밖에 없다.

참고문헌

[1] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin?. *Communications of the ACM*, 35(12):29-38, 1992.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123-140, 1996.

[3] L. Breiman. Arching the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[4] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-95), pp. 351-357, 1995.

[5] W. S. Cooper. On selecting a measure of retrieval effectiveness. *Journal of the American Society for Information Science* 24:87-100, pp. 413-424, 1973.

[6] H. Drucker and C. Cortes. Boosting decision trees. In *Advances in Neural Information Processing Systems 8*, pp. 479-485, 1996.

[7] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. AdaCost: Misclassification cost-sensitive boosting. In *Proc. Int. Conf. on Machine Learning* (ICML-99), 1999.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proc. 13th Int. Conf. on Machine Learning*, pp. 148-156, 1996.

[9] S. P. Harter. A probabilistic approach to automatic keyword indexing. Parts1 and 2. *Journal of the American Society for Information Science*, 26:197-206 and 280-289, 1975.

[10] D. Hull. The TREC-8 filtering track: Description and analysis. In *Proc. 7th Text Retrieval Conf.* (TREC-7), pp. 33-56, 1998.

[11] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, Adaptive mixture of local experts, *Neural Computation* vol. 3, pp. 79-87, 1991.

[12] T. Joachims. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In *Proc. Int. Conf. on Machine Learning* (ICML-97), pp. 143-151, 1997.

[13] K. L. Kwok, L. Grunfeld, M. Chan, N. Dinstl, and C. Cool. TREC-8 ad-hoc, query and filtering track experiments using PIRCS. In *Proc. Text Retrieval Conf.* (TREC-8), pp. 107-116, 1998.

[14] D. Lewis, R. E. Schapire, J. P.Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-96), pp. 298-306, 1996.

[15] David Lewis. Evaluating and optimizing autonomous text classification systems. In *Proc. 18th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-95), pp. 246-255, 1995.

[16] M. E. Maron and J. L. Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the Association for Computing Machinery*, 7:216-244, 1960.

[17] A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proc. Int. Conf. on Machine Learning* (ICML-98), pp. 350-358, 1998.

[18] T. M. Mitchell, *Machine Learning*, The McGraw-Hill, 1997.

[19] J. R. Quinlan, Bagging, boosting and C4.5. In *Proc. AAAI-96*, pp. 725-730, 1996.

[20] G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, *McGraw-Hill*, 1983.

[21] S. E. Robertson and S. Walker. Some simple effective approximations to the 2 Poisson model for probabilisticweighted retrieval. In *Proc. 17th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-94), pp. 232-241, 1994.

[22] S. E. Robertson. The probabilistic ranking principle in IR. *Journal of Documentation*, 33:294-304, 1977.

[23] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129-146, 1976.

[24] R. E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297-336, 1999.

[25] R. E. Schapire, Y. Freund, P. Barlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651-1686, 1998.

[26] R. E. Schapire, Yoram Singer, and Amit Singhal. Boosting and Rocchio applied to text filtering. In *Proc. 21st Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-98), pp. 251-223, 1998.

[27] A. Singhal, M. Mitra, and C. Buckley. Learning routing queries in a query zone. In *Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, (SIGIR-96), pp. 21-29, 1996.

[28] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proc. 19th Annual Int. ACM SIGIR Conf. on Research and Development in Inforamtion Retrieval*, (SIGIR-96), pp. 21-29, 1996.

[29] A. Singhal, J. Chio, D. Hindle, D. Lewis, and F. Pereira. AT&T at TREC-7. In *Proc. 7th Text Retrieval Conf.* (TREC-7), pp 239-252, 1998.

[30] D. K. Harman. Overview of 8th Text Retrieval Conference (TREC-8). In *Proc. 8th Text Retrieval Conf.* (TREC-8), pp. 1-19, 1999.

[31] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. 22nd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-99), pp. 42-49, 1999.

[32] Y-H. Kim, S-Y. Hahn, and B-T. Zhang, Text filtering by boosting naive Bayes, In *Proc. 23rd Annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval* (SIGIR-2000), pp. 168-175, 2000.

Abstract

Several machine learning algorithms have recently been used for text categorization and filtering. In particular, boosting methods such as AdaBoost have shown good performance applied to real text data. However, most of existing boosting algorithms applied to text filtering are based on classifiers that use binary-valued characteristics. Thus they do not fully make use of the important information such as term frequencies and lengths of documents

Another feature of text filtering worth noting is that the cost of not retrieving relevant documents is higher than the cost of retrieving non-relevant documents. Thus, we should design filtering algorithms considering these features. Unfortunately, AdaBoost is focused on minimizing the error, which is not suitable for text filtering problems.

In this thesis, we present a boosting-based learning method for text filtering that uses modified naive Bayes classifiers as a weak learner. The use of modified naive Bayes allows the boosting algoritm to utilize the features of text filtering such as term frequency and document length information. Additionally, we adopted a refined Boosting algorithm optimized for cost-sensitive text filtering.

Applied to TREC-7 and TREC-8 filtering track documents, the proposed method obtained competitive results in LF1, LF2, F1, and F3 measures compared to the best results submitted by other TREC entries. Furthermore our algorithm showed a significant improvement over AdaBoost algorithms based on decision stumps.

감사의 글

대학원에 들어온 지 벌써 2년이 지났습니다. 지금 이 자리에 서서 지나온 날들을 되짚어 보니 많은 사건들과 사람들이 머릿속에 스쳐 지나갑니다. 정화랑 걸어다니던 여의도, 남도학숙에서의 생활, 그리고 대학원 시절의 모든 것들이 저에게는 잊고 싶지 않은 추억으로 자리매김할 것입니다. 저는 지금 이 자리를 빌어 이러한 추억이 즐거운 추억이 되도록 도와 주신 많은 분들게 감사를 드리고 싶습니다.

먼저 저의 미래에 대한 비전을 제시하여 주신 장병탁 교수님께 진심으로 감사를 드립니다. 무엇보다도 교수님을 통해서 진정한 학자의 모습을 배우게 된 것을 기쁘게 생각합니다.

지난 2년동안 같은 방에서 생활하며 학문적으로 정서적으로 많은 도움을 준 성동이형을 비롯한 419호 분들, 약간 멀리 떨어져 있어서 추울 때나 심심할 때 신문 읽으러 가끔 들렀던 451호 분들, 마찬가지로 심심할 때마다 들렀던 417호 분들께도 감사를 드립니다. 남도학숙에서 생활하며 인생의 여러 모습을 보여준 종길이, 남수형, 상록이, 승환이, 용준이, 훈이, 진희.. 그리고 남도학숙에 지금도 계신 여러 선생님들께도 고마운 마음을 전합니다. 특히 3년동안 저와 같이 있으면서 정서적 학문적으로 많은 도움을 준 사랑스런 정화에게 감사의 마음을 전합니다.

마지막으로 제가 직접 만나보지는 못하고 책을 통해서만 간접적으로 접할 수 있었지만 저의 인생관을 확립하는데 많은 도움을 준 니체, 노자, 마광수, 강준만 등에게도 심심한 감사를 표합니다.