한국어 구 단위화를 위한 규칙 기반 방법과 기억 기반 학습의 결합

(A Hybrid of Rule based Method and Memory based Learning for Korean Text Chunking)

박 성 배 † 장 병 탁 ††
(Seong-Bae Park) (Byoung-Tak Zhang)

요 약 한국어나 일본어와 같이 부분 어순 자유 언어에서는 규칙 기반 방법이 구 단위화에 있어서 매우 유용한 방법이며, 실제로 잘 발달된 조사와 어미를 활용하면 소수의 규칙만으로도 여러 가지 기계학습 기법들만큼 높은 성능을 보일 수 있다. 하지만, 이 방법은 규칙의 예외를 처리할 수 있는 방법이 없다는 단점이 있다. 예외 처리는 자연언어처리에서 매우 중요한 문제이며, 기억 기반 학습이 이 문제를 효과적으로 다룰 수 있다. 본 논문에서는, 한국어 단위화를 위해서 규칙 기반 방법과 기억 기반 학습을 결합하는 방법을 제시한다. 제시된 방법은 우선 규칙에 기초하고, 규칙으로 추정한 단위를 기억 기반 학습으로 검증한다. STEP 2000 말뭉치에 대한 실험 결과, 본 논문에서 제시한 방법이 규칙이나 여러 기계학습 기법을 단독으로 사용하였을 때보다 높은 성능을 보였다. 규칙과 구 단위화에 가장 좋은 성능을 보인 Support Vector Machines의 F-score가 각각 91.87과 92.54인데 비하여, 본 논문에서 제시된 방법의 최종 F-score는 94.19이다.

키워드 : 구 단위화, 기억 기반 학습, 규칙 기반 방법, 하이브리드 방법

Abstract In partially free word order languages like Korean and Japanese, the rule-based method is effective for text chunking, and shows the performance as high as machine learning methods even with a few rules due to the well-developed overt postpositions and endings. However, it has no ability to handle the exceptions of the rules. Exception handling is an important work in natural language processing, and the exceptions can be efficiently processed in memory-based learning. In this paper, we propose a hybrid of rule-based method and memory-based learning for Korean text chunking. The proposed method is primarily based on the rules, and then the chunks estimated by the rules are verified by memory-based classifier. An evaluation of the proposed method on Korean STEP 2000 corpus yields the improvement in F-score over the rules or various machine learning methods alone. The final F-score is 94.19, while those of the rules and SVMs, the best machine learning method for this task, are just 91.87 and 92.54 respectively.

Key words : Text Chunking, Memory-Based Learning, Rule-Based Learning, Hybrid Method

1. 서 론

구 단위화는 Ramshaw와 Marcus가 기계학습 기법을 처음으로 적용한 이래[1] 자연언어 학습 분야에서 가장 흥미 있는 문제 중의 하나가 되어 왔다. 이 문제에 적용된 기계학습 방법의 목적은 주어진 데이터를 가장 잘 설명하는 가설을 찾는 것이며, 영어에 대해서 상대적으로 높은 성능을 보여 왔다[2-5]. 이런 방법들은 단어의 단위 종류(chunk type)를 결정하기 위해서 이웃한 단어들의 어휘 정보, 품사, 문법관계 등 다양한 정보를 사용하였다. 영어에서는 어순이 구문 제약으로서 중요한 역할을 하기 때문에, 이와 같은 지역적 정보만으로도 높은 성능을 얻을 수 있었다. 즉, 영어 단위화에서는 어순 제약 덕분에 오른쪽과 왼쪽 문맥에 있는 둘 또는 세 단어만 보더라도 좋은 가설을 찾을 수 있었다.

그러나, 한국어나 일본어와 같이 부분 어순 자유 언어

· 이 논문은 과기부 NRL, BrainTech 프로그램과 교육부 BK 21 사업에 의하여 지원되었음
† 비 회 원 : 서울대학교 컴퓨터공학부
경북대학교 컴퓨터공학과
seongbae@knu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
btzhang@bi.snu.ac.kr
논문접수 : 2002년 5월 18일
심사완료 : 2003년 11월 17일

들은 어순에 의한 구문 제약이 약하기 때문에 이런 방법들이 적합하지 않다. 즉, 이런 언어들은 구문 제약 대신에 구문 관계와 구(phrase)의 구성을 제약하는 조사와 어미가 발달해 있다[6]. 이런 언어에서 조사와 어미를 사용하지 않으면서도 좋은 가설을 찾으려면 문맥의 범위를 넓혀야 한다. 하지만, 문맥의 범위를 넓히게 되면, 차원성의 저주(curse of dimensionality)에 직면하여 일반화 성능이 나빠지게 된다[7]. 한국어에서는 조사와 어미가 명사구와 동사구 단위화에서 각각 중요한 역할을 한다. 따라서, 이런 정보를 사용하는 몇 개의 간단한 규칙만으로도, 한국어 단위화의 성능은 기계학습 기법이나 통계기반 방법 등과 같은 추론 모델과 거의 비슷하게 될 수 있다[8,9]. 그러나, 비록 이 규칙들이 비교적 정확하다고 할지라도, 이 규칙 속에 포함된 지식이 가능한 모든 경우에 대해 충분하다고 할 수는 없다. 전체 자연언어처리 과정에서 보면 단위화가 구문분석보다 앞서서 행해져야 하므로, 이 단계에서의 오류는 다음 단계에서 치명적이다. 따라서, 높은 성능을 얻기 위해서는 규칙에 의해 무시된 예외들을 잘 처리하여야 한다.

본 논문의 목적은 규칙의 예외를 효과적으로 처리하기 위하여 규칙 기반 방법과 기계학습 방법을 결합하여 한국어 단위화를 해결하는 방법을 제시하는 것이다. 제시된 방법은 기본적으로 규칙에 기반을 둔다. 즉, 단위의 종류는 우선 규칙에 의해 결정되고, 기계학습 방법에 의해 검증된다. 여기에서 기계학습 방법의 역할은 현재의 문맥이 규칙의 예외 상황인지 아닌지를 결정하는 것이다. 따라서, 기계학습 방법으로는 예외를 효과적으로 처리할 수 있는 기억 기반 학습(memory-based learning)[10]이 사용되었다.

본 논문의 구성은 다음과 같다. 먼저 2장에서 관련 연구를 설명하고, 3장에서 문제를 명확히 정의한다. 4장에서 제시된 방법이 어떻게 동작하는지 설명한 후, 5장에서 실험 결과를 제시한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련 연구

한국어 단위화에 대한 연구는 크게 보아서 두 가지로 분류될 수 있다. 하나는 규칙을 사용하는 방법이고, 다른 하나는 기계학습 기법을 사용하는 것이다. [8]과 [9]는 모두 규칙을 사용하여 단위화를 하였다는 점에서 비슷하나, [9]에서는 정규식으로 명사구를 묶고 장벽 알고리듬을 사용하여 동사구를 묶고자 하였다. [11]은 세 단계로 단위화를 수행하는 방법을 제시하였다. 우선, 형태소 특성을 이용하여 기본적인 구를 묶은 후, 문맥의존문법을 이용하여 명사구를 인식하였다. 마지막으로, 말뭉치에서 추출한 연어 패턴을 사용하여 동사구를 인식하고자 하였다. 이런 방법들은 주로 명사구와 동사구만을 인식하기 위해 적용되었고, 규칙만 사용하였기 때문에 모든 언어 현상을 모두 표현하기가 힘들었다. 또한, 규칙 작성이 사람에 의해 이루어졌기 때문에 규칙들 간의 일관성을 유지하기 힘든 단점도 있다.

이에 비해, [12]와 [13]은 기계학습 기법을 한국어 단위화에 적용하였다. [12]는 결정트리 학습과 최대 엔트로피 모델을 결합하는 방법을 제시하였고, [13]은 가중적인 확률 합에 의한 단위화 방법을 제시하였다. 하지만, 이런 방법들은 제한된 문맥 정보만 사용함으로써 한국어가 가지는 특성을 무시하는 결과를 초래하였다.

3. 문제 정의

단위화의 목적은 주어진 문장을 중복이 없는 최소의 구로 나누는 것이다. 예를 들어, 아래의 문장을 보자.

또한 그 사실은 기지의 안전에 매우 좋을 수 있다.

이 문장은 15 개의 단어로 이루어져 있는데[1], 아래와 같이 여섯 개의 단위로 나누어질 수 있다.

[IP 또한] [NP 그 사실은] [NP 기지의 안전에] [ADVP 매우] [VP 좋을 수 있다] [O .]

문장의 길이를 n이라고 했을 때 자연언어 구문 분석을 위한 일반적인 차트 파서의 복잡도는 $O(n^3)$이다[14]. 일반적으로 구 단위화의 결과로 $m \ll n$ 개의 단위가 만들어지므로, 구 단위화를 통해 파싱에서 높은 효율을 기대할 수 있다.

단위화 문제는 문장 내의 단어 w_i의 입장에서 보면 다중 클래스 분류 문제이다. T를 w_i가 취할 수 있는 단위 종류의 집합이라고 하자. θ를 파라미터로 가지는 분류기 $f(w_i, \theta)$가 있을 때, w_i의 단위 종류 t^*는 다음과 같이 결정된다.

$$t^* = \arg\max_{t_j \in T} \{ f(w_i, \theta) = t_j \}$$

여기서, 파라미터 θ는 문맥 정보와 분류기 f를 학습시키는데 필요한 분류 파라미터를 모두 포함한다.

4. 한국어 단위화

그림 1은 한국어 단위화를 위해 본 논문에서 제시한 모델의 구조이다. 이 모델의 주 아이디어는 단어 w_i의 단위 종류를 결정하기 위해 규칙과 기억 기반 학습을 결합하는 것이다. 이를 위해, 규칙을 먼저 적용한 후 학습된 기억 기반 분류기를 통해 현재 문맥이 규칙의 예외 상황인지를 점검한다. 이는 현재 문맥이 규칙의 예외 상황이면 규칙의 결정을 따르지 않기 위해서이다. 따라서, 학습 단계에서는 각 문장을 규칙에 의해 분석하고

1) 편의상, 조사, 어미, 문장 부호 등을 모두 독립된 단어로 가정하였다.

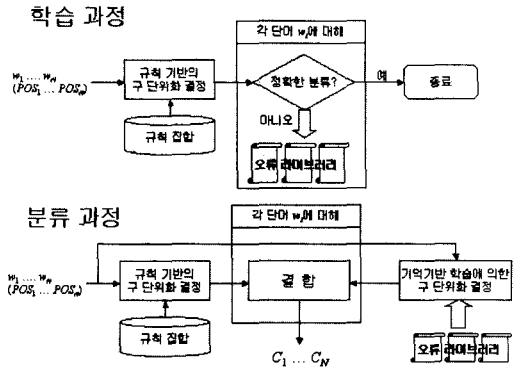

그림 1 한국어 단위화를 위한 모델의 구조

규칙에 의해 추정된 단위를 실제 단위와 비교하여 검증한다. 즉, 단어 w_i의 단위 종류를 c_i라고 하면, 학습 예제는 $\langle w_i, c_i \rangle$형태로 주어진다. 규칙에 의해 추정된 w_i의 단위 종류가 c_i와 틀리면 이 예제는 규칙의 예외로 인정된다. 그리고, 이렇게 잘못 분석된 예제는 오류 라이브러리(error library)에 기록된다. 이 오류 라이브러리에는 규칙의 예외만 기록되므로, 규칙이 예제 공간을 충분히 잘 표현한다면 이 라이브러리에 속하는 예제의 수가 적게 된다.

이 모델에서 사용되는 기계학습 방법인 기억 기반 학습은 새로운 예제의 분류가 학습 단계에서 기억된 예제들과의 유사도에 의해 결정되기 때문에 *lazy learning*이라고도 불린다. 제시된 모델에서는 기억 기반 학습이 오류 라이브러리에 있는 예제들만 학습하게 된다. 따라서, 이 라이브러리에 속하는 예제의 수가 적다면 새로운 예제의 단위 종류를 빠르게 추론할 수 있다.

그림 1의 분류 단계는 그림 2에서처럼 의사코드로 표현될 수 있다. w_i의 단위화 종류는 함께 주어진 문맥 C_i를 사용하여 결정한다. 우선, 단위 종류를 결정하기 위해 규칙을 먼저 적용한 후에, C_i가 규칙의 예외인지 아닌지를 결정한다. 만약 예외이라면, 규칙에 의해 결정된 단위 종류는 취소되고 다시 기억 기반 학습에 의해 단위 종류가 결정된다. 예외인지 아닌지를 결정하는 조건은 C_i와 오류 라이브러리 내에서 C_i와 가장 비슷한 예제 e의 유사도가 임계치(threshold) τ를 넘느냐 하는 것이다. C_i와 e의 유사도를 결정하는 함수 $\Delta(C_i, e)$는 3.2절에서 자세히 설명한다. 오류 라이브러리가 예외만 포함하고 있기 때문에, C_i가 e와 비슷할수록, C_i가 규칙의 예외일 가능성이 높다.

```
함수 결합에 의한 분류
  입력: 단어 w_i와 문맥 C_i, 임계치 τ
  출력: w_i의 단위화 종류 t

[과정 1] t ← w_i의 단위화 종류를 규칙으로 결정.
[과정 2] e ← 오류 라이브러리에서 C_i와 가장 가까운 예제.
[과정 3] If Δ(C_i, e) ≥ τ,
         then t ← w_i의 단위화 종류를 기억 기반 학습으로 결정.
```

그림 2 규칙 기반 방법과 기억 기반 추론을 합치는 알고리듬

4.1 규칙에 의한 단위화

본 논문에서는 한국어에는 명사구(NP), 동사구(VP), 부사구(ADVP), 독립구(IP)의 네 종류의 기본구가 있다고 가정한다.

4.1.1 명사구 단위화

단어 w_i의 단위 종류는 w_i의 품사가 관형사, 명사, 대명사 중의 하나일 때, 다음과 같은 일곱 개의 규칙에 의해 결정된다.

1. **If** $POS(w_{i-1})$ = 관형사 **and** $postposition(w_{i-1})$ = ∅
   Then t_i = I-NP.
2. **Else If** $POS(w_{i-1})$ = 대명사 **and** $postposition(w_{i-1})$ = ∅
   Then t_i = I-NP.
3. **Else If** $POS(w_{i-1})$ = 명사 **and** $postposition(w_{i-1})$ = ∅
   Then t_i = I-NP.
4. **Else If** $POS(w_{i-1})$ = 명사 **and** $postposition(w_{i-1})$ = 소유격
   Then t_i = I-NP.
5. **Else If** $POS(w_{i-1})$ = 명사 **and** $postfix(w_{i-1})$ = 관형격
   Then t_i = I-NP.
6. **Else If** $POS(w_{i-1})$ = 형용사 **and** $ending(w_{i-1})$ = 관형형
   Then t_i = I-NP.
7. **Else** t_i = B-NP.

여기서, $POS(w_i)$는 단어 w_i의 품사를 나타내고, $postposition(w_i)$은 w_i의 조사 종류를, $postfix(w_i)$는 w_i의 접미사 종류를, $ending(w_i)$은 w_i의 어미 종류를 나타낸다. B-NP는 명사구의 첫 번째 단어를 의미하며, I-NP는 명사구 내의 다른 모든 단어에 주어지는 단위 종류이다. 예를 들어, 앞의 예문에서 단어 '*그*'는 NP 내의 첫 번째 단어이기 때문에 이 단어의 단위 종류는 B-NP이고, '*사실*'과 조사 '*은*'에는 I-NP라는 단위 종류가 주어진다.

관형사, 명사, 대명사는 한국어에서 비슷한 구문 역할을 하므로, 조사 없이 이어진 이런 단어들은 하나의 명사구를 이룬다. 규칙 1-3이 이에 해당한다. 조사를 가지는 모든 명사는 명사구의 끝이 될 수 있지만, 거기에는 두 가지의 예외가 있다. 첫 번째는 *소유격 조사*이다. 만약 어떤 명사가 소유격 조사로 끝나면 그 명사는 여전히 명사구 내에 있게 된다(규칙 4). 따라서, 위의 예문에서 '*안전*'이 소유격 조사 '*의*'의 뒤에 나타나므로 '*안전*'은 여전히 명사구 내에 있게 된다. 이 규칙이 없었다면 '*안전*'은 새로운 명사구의 첫 번째 단어가 되었을 것이다. 또 다른 예외는 관형격 접미사 '*적*'이며(규칙 5), 이

규칙의 효과는 소유격 조사와 같다. 규칙 6은 하위 성분이 없는 관형절이 새로운 구를 구성하지 못함을 의미한다. 한국어의 형용사에는 한정적 용법이 없으므로, 이 규칙은 영어 형용사의 한정적 용법에 해당된다. 이 여섯 가지 규칙에 적용되지 않는 모든 단어의 단위 종류는 B-NP이다(규칙 7).

4.1.2 동사구 단위화

동사구 단위화는 '복합 동사 처리'라는 이름으로 오랫동안 연구되어 왔으며, 높은 성능을 보이기도 하였다. [8]과 [9]는 동사구 단위화를 위해 각각 독립적으로 유한 상태 자동자(finite state automaton)를 사용하였고, [15]는 지식 기반 규칙을 사용하였다. 본 논문에서는 명사구와의 일관성을 유지하기 위하여, 규칙을 사용하였다. 사용된 규칙은 [15]에서 제시한 것으로, 자세한 설명은 생략한다. 사용된 규칙의 수는 29 개이고, 다음은 동사구 단위화를 위한 두 예제 규칙이다.

- **If** w_{i-1} = '*을*' **and** w_i = '*수*' **and** w_{i+1} = '*있*'
  Then t_i = I-VP.
- **If** w_{i-2} = '*을*' **and** w_{i-1} = '*수*' **and** w_i = '*있*'
  Then t_i = I-NP.

4.1.3 부사구 단위화

부사가 연속으로 나타나면, 그 부사들은 하나의 부사구를 이루려는 성향이 강하다. 연속된 부사가 항상 하나의 부사구를 이루는 것은 아니지만, 많은 경우에 하나의 부사구를 이룬다. 표 1은 이 사실을 실험적으로 보이고 있다. STEP 2000 말뭉치에서 연속적인 부사의 사용을 조사하였고 270 개의 예제가 발견되었다. 이 중 189 개의 예제가 하나의 구를 이루는 데 비하여, 나머지 81 개만이 두 개의 독립적인 구를 이루었다. 따라서, 연속적인 부사가 하나의 구를 이룰 확률이 두 개의 구를 이룰 확률보다 훨씬 높다고 말할 수 있다.

표 1 부사의 연속이 하나의 단위를 이룰 확률

	예제의 수	확률
하나의 구	189	0.70
두 개의 구	81	0.30

요약하면, w_i의 품사가 부사일 때, w_i의 단위 종류는 다음과 같은 규칙에 의해 결정된다.

1. **If** $POS(w_{i-1})$ = 부사 **Then** t_i = I-ADVP.
2. **Else** t_i = B-ADVP.

4.1.4 독립구 단위화

독립구 단위화는 사전에 이미 알고 있던 독립구에 대한 지식에 기반하여 처리하였다. 하지만, 독립구와 명사구를 구분하기가 어렵기 때문에, 아무리 많은 사전 지식이 있다 하더라도 독립구와 명사구 사이에는 모호성이 존재할 수 있다. 본 논문에서는 독립구 단위화를 위해 12 개의 규칙을 정의하였으며, 다음은 이들 중 두 개의 예제 규칙이다.

- **If** w_i = '*뿐*' **and** w_{i+1} = '*만*' **and** w_{i+2} = '*아니*' **and** w_{i+3} = '*라*'
  Then t_i = B-IP.
- **If** $POS(w_i)$ = 명사 **and** $POS(w_{i+1})$ = 호격 조사
  Then t_i = B-IP.

4.2 기억 기반 학습에 의한 단위화

기억 기반 학습은 k-nearest neighbor(k-NN) 알고리듬[16]을 직접적으로 계승한 알고리듬이다. 많은 자연언어처리 문제가 수많은 예제와 서로 다른 중요도를 갖는 속성을 가지므로, 기억 기반 학습은 자연언어처리 문제에 적용되기 위해 k-NN보다 복잡한 자료 구조와 더 빠른 속도 최적화 방법을 가지고 있다[10,17].

기억 기반 학습은 *학습 단계*와 유사도 기반의 *분류 단계*로 이루어진다. 학습 단계는 n 개의 속성을 갖는 벡터인 예제들을 메모리에 저장하는 작업을 하며, 분류 단계에서는 이를 기반으로 하여 새로운 예제를 분류한다. 여기서, 한 예제 $\boldsymbol{x} = (x_1, ..., x_n)$와 메모리 내의 모든 예제 $\boldsymbol{y} = (y_1, ..., y_n)$ 사이의 유사도는 *거리 단위*, $\Delta(\boldsymbol{x}, \boldsymbol{y})$로 계산된다. $\boldsymbol{x}$의 단위 종류는 $\boldsymbol{x}$와 가장 가까운 k 개 예제들의 최다 클래스로 결정된다. 본 논문에서는 $k = 1$로 제한한다.

두 예제 $\boldsymbol{x}$와 $\boldsymbol{y}$ 사이의 거리, $\Delta(\boldsymbol{x}, \boldsymbol{y})$는 다음과 같이 정의된다.

$$\Delta(\boldsymbol{x}, \boldsymbol{y}) \equiv \sum_{i=1}^{n} \alpha_i \delta(x_i, y_i)$$

여기서, α_i는 i 번째 속성의 중요도이고, $\delta(x_i, y_i)$는 다음과 같다.

$$\delta(x_i, y_i) = \begin{cases} 1 & \text{if } x_i = y_i, \\ 0 & \text{if } x_i \neq y_i. \end{cases}$$

만약 α_i가 정보 이득[18]에 의해 결정되면, 위의 단위를 사용하는 k-NN 알고리듬을 *IB1-IG*라고 한다[17]. 본 논문에서는 IB1-IG를 사용하여 모든 기억 기반 학습을 실험하였다.

표 2는 한국어 단위화를 위한 IB1-IG의 속성을 보인다. 단어 w_i의 단위 종류를 결정하기 위해서, 주변 단어의 어휘, 품사, 단위 종류가 사용되었다. 주변 단어로는 어휘와 품사에 대해서는 각각 왼쪽과 오른쪽 문맥의 세 단어가 사용되었고, 단위 종류를 위해서는 왼쪽 문맥의 세 단어만이 사용되었다. 이는 단위화가 순차적으로 수행되기 때문에, 오른쪽 문맥에 있는 단어들의 단위 종류는 w_i를 결정할 때 알려지지 않았기 때문이다.

예를 들어, 다음과 같은 문장이 품사와 함께 주어졌다고 가정하자.

표 2 한국어 단위화를 위한 IB1-IG의 속성

속성	설명	속성	설명
W_{i-3}	w_{i-3}의 어휘	POS_{i-3}	w_{i-3}의 품사
W_{i-2}	w_{i-2}의 어휘	POS_{i-2}	w_{i-2}의 품사
W_{i-1}	w_{i-1}의 어휘	POS_{i-1}	w_{i-1}의 품사
W_i	w_i의 어휘	POS_i	w_i의 품사
W_{i+1}	w_{i+1}의 어휘	POS_{i+1}	w_{i+1}의 품사
W_{i+2}	w_{i+2}의 어휘	POS_{i+2}	w_{i+2}의 품사
W_{i+3}	w_{i+3}의 어휘	POS_{i+3}	w_{i+3}의 품사
T_{i-3}	w_{i-3}의 단위 종류	T_{i-2}	w_{i-2}의 단위 종류
T_{i-1}	w_{i-1}의 단위 종류		

언젠가(mag) **반드시**(mag) **도움**(ncn) **이**(jcc) **되**(pvg) **는다**(ef) .(sf)

여기서, 두 단어 '*언젠가*와 '*반드시*'는 모두 부사이다. 따라서, 4.1.3절의 규칙에 의해 '*언젠가*'는 B- ADVP라는 단위 종류를, '*반드시*'는 I-ADVP라는 단위 종류를 갖게 된다. 그리고, '*반드시*'의 속성은

<W_{i-3} = ∅, W_{i-2} = ∅, W_{i-1} = 언젠가, W_i = 반드시, W_{i+1} = 도움, W_{i+2} = 이, W_{i+3} = 되, $POS(w_{i-3})$ = ∅, $POS(w_{i-2})$ = ∅, $POS(w_{i-1})$ = mag, $POS(w_i)$ = mag, $POS(w_{i+1})$ = ncn, $POS(w_{i+2})$ = jcc, $POS(w_{i+3})$ = pvg, T_{i-3} = ∅, T_{i-2} = ∅, T_{i-3} = B-ADVP>

이다. 그러나, 오류 라이브러리에서 찾아진 이 속성과 가장 비슷한 예제가

<W_{i-3} = ∅, W_{i-2} = ∅, W_{i-1} = 장차, W_i = 반드시, W_{i+1} = 패망, W_{i+2} = 이, W_{i+3} = 되, $POS(w_{i-3})$ = ∅, $POS(w_{i-2})$ = ∅, $POS(w_{i-1})$ = mag, $POS(w_i)$ = mag, $POS(w_{i+1})$ = ncps, $POS(w_{i+2})$ = jcc, $POS(w_{i+3})$ = pvg, T_{i-3} = ∅, T_{i-2} = ∅, T_{i-3} = B-ADVP>

이고 이 예제의 실제 단위가 B-ADVP라면, '*반드시*'의 단위 종류는 이 두 예제가 매우 유사하기 때문에 IB1-IG에 의해 B-ADVP로 변경된다.

자연언어처리에서 규칙의 예외를 무시할 수 없는 이유는 예외를 특이치(outlier)와 구분하기가 어렵기 때문이다[10]. Danyluk과 Provost는 이 문제를 해결하기 위하여 최대-특이성 바이어스(maximum-specificity bias)를 제시하였는데[19], 기억 기반 학습은 이를 구현하는 아주 자연스러운 방법이다[20]. 또한, 기억 기반 학습은 모든 학습 예제를 기억하고 있으므로, 예외 속에 포함된 좋은 정보를 잃어버리지 않는 장점이 있다.

4.3 다른 학습 알고리듬과 비교

본 논문에서는 규칙의 오류를 발견하고 이 오류를 보정하는 방법을 제시하였다. 이런 면에서 제시된 방법은 변형 기반 학습(transformation-based learning)[21]이나 AdaBoost[22]와 비슷하다. 특히, 이 두 방법은 모두 간단한 규칙을 결합하여 분류기를 만든다는 점에서 매우 유사하다. 하지만, AdaBoost의 이론적 근거가 분명한데 비해, 변형 기반 학습은 이론적 근거가 불분명하다. 그리고, 많은 자연언어처리 문제에서 AdaBoost가 변형 기반 학습보다 좋은 성능을 보였다[23].

변형 기반 학습의 단점은 다음과 같이 세 가지로 요약할 수 있다. 첫째, 이 방법은 주어진 학습 데이터를 과도하게 학습할 가능성이 높다. 두 번째 단점은 이 방법이 추가적인 변형을 통해 항상 오류의 수를 줄이려고 하기 때문에 데이터의 노이즈에 약하다는 점이다. 마지막으로, 학습 시 추가된 변형이 일반화 성능을 향상시킬 수 있다는 어떠한 이론적 근거가 없다. 이에 비해, AdaBoost는 변형 기반 학습과 비슷하기는 하지만, 최종적으로 만들어질 분류기의 일반화 성능이 이론적으로 분석되어 있으며[22], 이 알고리듬이 동작하기 위한 최소한의 가정도 이미 잘 알려져 있다. 즉, ε_t를 AdaBoost에서 사용된 t번째 약학습자의 오류율, $\gamma_t = 1/2 - \varepsilon_t$라고 하자. $\varepsilon_t \le 1/2$라면, AdaBoost의 최종 분류기 h_{fin}은 다음과 같은 오차한계치(error bound)를 갖는다.

$$\frac{1}{N}\left|\left\{i : h_{fin}(\mathbf{x}_i) \neq c_i\right\}\right| \le \exp\left(-2\sum_{t=1}^{T}\gamma_t^2\right)$$

여기서, N은 전체 학습 데이터의 수이고, T는 AdaBoost 내에 있는 약 학습자의 수이다. 따라서, $\varepsilon_t \le 1/2$라면 AdaBoost의 성능은 나빠지지 않는다.

AdaBoost가 본 논문에서 제시된 방법과 다른 점은 AdaBoost에서는 약 학습자(weak learner)가 모두 같은 방식으로 동작하고 하나 이상의 약 학습자가 사용된다는 것이다. 일반적으로 AdaBoost의 수렴 속도는 약 학습자의 성능에 의해 결정된다. 그러므로 해결하고자 하는 문제에 대한 사전지식을 이용하여 약 학습자의 성능을 높일 수 있다면 수렴 속도가 매우 빨라질 것이다. AdaBoost가 T 개의 약 학습자로 이루어졌다고 할 때, 제시된 방법은 앞의 $(T - 1)$ 개의 약 학습자가 하는 일

을 규칙으로 대신하고, 마지막 T 번째 약 학습자의 역할을 기억 기반 학습이 하고 있는 것으로 해석될 수 있다. 따라서, AdaBoost가 T 번 과정을 거치는데 비해 제시된 방법은 단 두 번의 과정만 거치므로, AdaBoost보다 훨씬 빠르게 동작한다.

5. 실험

5.1 실험 데이터

본 논문에서 제시한 방법의 모든 실험은 STEP 2000 한국어 단위화 데이터(STEP 2000 데이터)[2]에 대해서 평가되었다. 이 데이터는 STEP 2000 과제의 결과물인 구문 부착 말뭉치에서 추출되었다. 이 말뭉치는 12,092 개의 문장, 111,658 개의 구, 321,328 개의 단어로 이루어져 있으며, 사용된 단어의 수는 16,808 개다. 표 3은 STEP 2000 데이터에 대한 통계정보이다. 이 데이터의 형태는 CoNLL-2000 데이터를 따랐다. 그림 4는 이 데이터에 있는 예제 문장이다. 이 데이터에 있는 각 단어는 두 개의 부가 태그, 즉 품사 태그와 단위 종류를 가지며, 품사 태그는 KAIST 태그셋[24]에 기초한다.

또한, 각 구는 B-XP와 I-XP의 두 개의 단위 종류를 갖는다. 이 외에도, 추가로 O 단위 종류가 있는데, 이는 어느 구에도 속하지 않는 단어들에 주어진다. 이 데이터에는 4 종류의 구가 있고 추가적으로 O 단위가 있으므로, 총 9 개의 단위 종류를 가진다.

표 3 STEP 2000 한국어 단위화 데이터에 대한 간단한 통계

정보	값
어휘의 수	16,838
단어의 수	321,328
단위 종류의 수	9
품사의 수	52
문장의 수	12,092
구의 수	112,658

또한	maj	B-ADVP
그	mmd	B-NP
사실	ncn	I-NP
은	jxt	I-NP
기지	ncn	B-NP
의	jcm	I-NP
안전	ncps	I-NP
에	jca	I-NP
매우	mag	B-ADVP
좋	paa	B-VP
을	ef	I-VP
수	nbn	I-VP
있	paa	I-VP
다	ef	I-VP
.	sf	O

그림 4 STEP 2000 단위화 데이터의 예

2) STEP 2000 한국어 단위화 데이터는 http://bi.snu.ac.kr/~sbpark/Step2000에서 다운받아 사용할 수 있다.

실험 결과는 다음과 같은 표준 기준에 의해 평가되었다[25].

$$\text{정확도 } (Accuracy) = \frac{\text{정확하게 추정된 단어의 수}}{\text{총 단어 수}}$$

$$\text{재현도 } (Recall) = \frac{\text{정확하게 추정된 구의 수}}{\text{찾아야 할 전체 구의 수}}$$

$$\text{정밀도 } (Precision) = \frac{\text{정확하게 추정된 구의 수}}{\text{추정된 구의 수}}$$

$$F_\beta - score = \frac{(\beta^2+1)\cdot \text{재현도} \cdot \text{정밀도}}{\beta^2 \cdot \text{재현도} + \text{정밀도}}$$

모든 실험에서는 재현도와 정밀도에 같은 가중치를 주기 위해, $\beta = 1$로 설정하였다. 이 평가 기준에 의하면, 정확도는 단어 수준에서 평가하는 것이고, 나머지는 구 수준에서 평가하는 것이다.

5.2 규칙 기반 방법의 성능

표 4는 규칙만 사용했을 때의 성능이다. 규칙만 사용하였을 때, 우리는 97.99%의 정확도와 91.87의 F-score를 얻었다. 비교적 높은 정확도에도 불구하고, F-score는 정확도에 비해 높지 않은 편이다. 구 단위화의 응용 분야에서 중요한 단위는 구이므로, F-score가 정확도보다 훨씬 중요한 단위이다. 따라서, 아직 F-score에는 더 향상시킬 여지가 많다.

표 5는 규칙에 의한 오류의 종류와 그 분포를 보인다. 예를 들어, 오류 종류 'B-ADVP I-ADVP'는 실제 단위 종류가 B-ADVP인데 I-ADVP로 잘못 추정된 오류들을 포함한다. 이 표에는 모두 여덟 종류의 오류가 있지

표 4 규칙만 사용하였을 때의 실험 결과

종류	정밀도	재현도	F-score
ADVP	98.67%	97.23%	97.94
IP	100.00%	99.63%	99.81
NP	88.96%	88.93%	88.94
VP	92.89%	96.35%	94.59
평균	91.28%	92.47%	91.87

표 5 규칙으로 잘못 추정된 단위 종류에 따른 오류 분포

오류의 종류	오류의 수	비율 (%)
B-ADVP I-ADVP	89	1.38
B-ADVP I-NP	9	0.14
B-IP B-NP	9	0.14
I-IP I-NP	2	0.03
B-NP I-NP	2,376	36.76
I-NP B-NP	2,376	36.76
B-VP I-VP	3	0.05
I-VP B-VP	1,599	24.74
합계	6,463	100.00

표 6 여러 기계학습 알고리듬의 실험 결과

학습 알고리듬	정확도	정밀도	재현도	F-score
결정트리	97.95±0.24 %	92.29±0.94 %	90.45±0.80 %	91.36±0.85
SVM	98.15±0.20 %	93.63±0.81 %	91.48±0.70 %	92.54±0.72
MBL	97.79±0.29 %	91.41±1.24 %	91.43±0.87 %	91.38±1.01
규칙	97.99 %	91.28 %	92.47 %	91.87

만, 대부분의 오류는 명사구와 관련되어 있다. 우리는 이 현상에 대한 다음과 같은 두 가지의 이유를 발견하였다.

1. 명사구의 시작점을 찾는 것이 어렵다. 조사 없이 이어진 모든 명사가 하나의 명사구를 이루는 것은 아니다. 하지만, 본 논문에서 제시한 규칙에 따르면 이들은 항상 하나의 명사구가 된다.

2. 접속 조사 '와'가 매우 모호하다. '와'가 접속 조사로 사용되었을 때에는 이 조사와 다음 명사의 단위 종류가 "I-NP I-NP"이어야 하지만, '와'가 단순히 부사격 조사일 때는 단위 종류가 "I-NP B-NP"가 되어야 한다.

5.3 여러 가지 기계학습 방법과의 성능 비교

우선, 규칙만 사용했을 때의 성능을 여러 가지 기계학습 알고리듬과 비교하였다. 표 6은 한국어 단위화에 대한 세 종류의 기계학습 알고리듬에 대한 10회 교차검증[3] 실험 결과를 보인다. 비교 대상이 되는 기계학습 알고리듬은 (i) 기억 기반 학습(MBL), (ii) 결정트리(Decision Tree), (iii) Support Vector Machines (SVM)이다. 기억 기반 학습은 결정을 해야될 순간까지 주어진 학습 데이터를 처리하지 않는 방법으로, 전체 학습 데이터의 공간을 추정하는 가설(hypothesis)을 만들지 않는다. 대신에, 주어진 질의에 대하여 항상 목표 함수에 대한 지역적인 추정을 통해 답을 결정한다. 반면에, 결정트리와 SVM은 주어진 학습 데이터를 통해 전체 공간을 추정하는 가설을 만든 후, 이 가설을 통해 분류를 하는 학습 방법이다. 결정트리는 예제 공간을 나누어 감으로써 가설을 만들고, SVM은 마진(margin)이 가장 큰 초평면(hyperplane)을 찾음으로써 가설을 만든다.

본 논문에서는 결정트리 학습 알고리듬으로 C4.5 release 8[18]을 사용하였고, SVM으로는 SVM^{light}[26]를, 기억 기반 학습 알고리듬으로는 *TiMBL*[17]을 사용하였다. 결정트리와 SVM은 기억 기반 학습(표 2)과 같은 속성을 사용하였다. SVM은 기본적으로 이진 분류기이므로, 이를 다중 클래스 문제에 적용하기 위해서는 SVM을 확장하여야 한다. 본 논문에서는 이를 위해 쌍분류(pairwise classification) 방법을 사용하였다. 주어진 문제가 k 개의 클래스를 가지고 있다고 할 때, 이 방법의 기본 아이디어는 모든 가능한 쌍의 종류에 대하여 $k \times (k-1) / 2$ 개의 분류기를 만든 후, 다수결(majority voting)로 최종 결정을 하는 것이다. 즉, 한국어 단위화에는 9개의 클래스가 있으므로, $\frac{9\times 8}{2} = 36$개의 분류기를 만들었다. 일반적으로, 이진 분류기를 다중 클래스 문제로 확장할 때 자주 쓰이는 "one vs. all others" 방법 대신에 쌍분류 방법을 사용한 것은 쌍분류 방법이 SVM에서는 더 좋은 성능을 보이는 것으로 알려져 있기 때문이다[27].

어휘 정보로는, 말뭉치에서 10 번 이상 나타난 단어만 사용하여서 2,210 개만 실험에서 사용하였다. 이 알고리듬들 중 기억 기반 학습과 결정트리는 규칙보다 낮은 성능을 보였다. 규칙의 F-score가 91.87인데 비하여, 기억 기반 학습과 결정트리의 F-score는 각각 91.38과 91.36이다. 반면에, SVM은 규칙보다 조금 더 나은 성능을 보였다. SVM의 F-score는 92.54로, 규칙보다 0.67 높은 성능을 보였다.

표 7은 기억 기반 학습이 사용되었을 때 얻어진 속성의 중요도이다. 이 표의 각 값은 거리 단위 $\Delta(\boldsymbol{x}, \boldsymbol{y})$를 계산할 때의 α_i 값에 해당한다. 속성이 중요하면 할수록, 중요도는 더 커진다. 이 표에 따르면, 17 개의 속성 중 가장 중요한 것은 바로 앞 단어의 단위 종류인 C_{i-1}이다. 반면에, 가장 중요하지 않은 속성은 W_{i-3}과 C_{i-3}이다. 이 사실은 단어들이 w_i에서 멀어질수록 w_i의 단위

표 7 IB1-IG에서 속성의 중요도

속성	중요도	속성	중요도
W_{i-3}	0.03	POS_{i-3}	0.04
W_{i-2}	0.07	POS_{i-2}	0.11
W_{i-1}	0.17	POS_{i-1}	0.28
W_i	0.22	POS_i	0.38
W_{i+1}	0.14	POS_{i+1}	0.22
W_{i+2}	0.06	POS_{i+2}	0.09
W_{i+3}	0.04	POS_{i+3}	0.05
C_{i-3}	0.03	C_{i-2}	0.11
C_{i-1}	0.43		

3) *k회 교차검증*은 주어진 알고리듬을 k 번 실행한 결과의 평균으로 평가하는 방법을 말한다. 초기의 학습 데이터를 k 개로 나눈 후, k-1 개의 부분 데이터로 학습하고 나머지로 알고리듬을 평가한다. 서로 다른 k 개의 부분 데이터에 대해서 이 과정을 반복한 후, 결과의 평균으로 알고리듬을 평가한다.

종류를 결정하는데 영향을 덜 미칠 것이라는 직관과 일치한다. 즉, 어휘 정보에 대해서 중요한 순서대로 정렬하면, $<W_i, W_{i-1}, W_{i+1}, W_{i-2}, W_{i+2}, W_{i+3}, W_{i-3}>$이 된다. 같은 현상이 품사와 단위 종류에서도 발견된다. 품사와 어휘정보를 비교해 보면, 품사가 어휘정보보다 더 중요한데, 이는 어휘 정보가 너무 희박하기 때문인 것으로 추정된다.

5.4 규칙과 기억 기반 학습 방법이 결합되었을 때의 성능

표 8은 규칙과 기억 기반 학습을 결합한 방법에 대한 최종 결과를 보인다. F-score는 평균적으로 94.19이고, 이는 규칙 기반보다 2.32, SVM보다 1.65, 그리고 기억 기반 학습보다 2.81만큼 향상된 결과이다. 또한, 이 성능은 영어에 대한 단위화 결과[4]와 비슷하다.

그림 3은 각 구에 대해 F-score가 얼마나 향상되었는지를 보인 것이다. 명사구의 평균 F-score는 94.54로 규칙 기반 방법보다 훨씬 향상되었다. 이는 규칙의 예외들이 기억 기반 학습에 의해 잘 처리되었음을 의미한다. 하지만, 명사구와 동사구에 대해서는 성능 향상이 많았으나, 동사구와 독립어구에 대해서는 성능 향상이 크지 않았다. 이 결과는 부사구와 독립어구에 대한 예제의 수가 너무 적었기 때문으로 해석될 수 있다. 이런 구에 대한 규칙의 정확도가 이미 충분히 높기 때문에 대부분의 예제가 규칙에 의해 처리되었고, 기억 기반 학습은 규칙의 예외 상황만 처리하기 때문에 성능 향상에 크게 도움이 되지 않았다.

표 8 규칙 기반 방법과 기억 기반 학습을 합친 방법의 최종 결과

정확도	정밀도	재현도	F-score
98.07±0.27 %	94.47±1.20 %	93.91±0.99 %	94.19±1.08

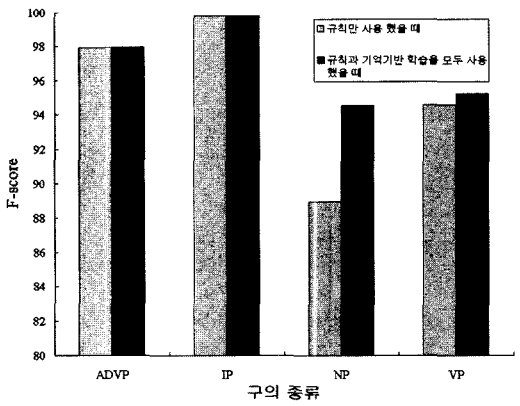

그림 3 규칙 기반 방법과 MBL을 합침으로써 얻은 각 구의 성능 향상

그림 4는 주어진 문맥이 규칙의 예외인지를 결정하는데 사용되는 임계치 τ의 다양한 값에 대해 언제 가장 좋은 성능을 보이는지를 나타낸다. 표 7의 모든 중요도의 합은 2.48이다. 따라서, 만약 $\tau>2.48$이면, 이 임계치로는 예외가 생기지 않기 때문에 규칙만 적용된다. 반대로, $\tau=0.00$ 일 때에는 기억 기반 학습만 사용된다. 이 경우에는 기억 기반 학습이 규칙의 예외에 기초해서 w_i의 단위 종류를 결정한다. 따라서, F-score가 32.03으로 매우 낮은 성능을 보였다. 마지막으로, 가장 좋은 성능은 τ가 1.95일 때 얻어진 94.19이다.

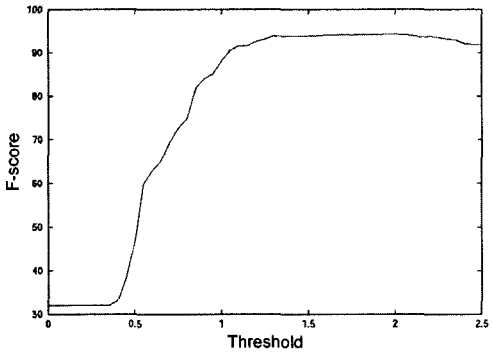

그림 4 임계치 값에 따른 F-score 값의 변화

6. 결 론

본 논문에서 우리는 규칙 기반 방법과 기억 기반 학습을 결합함으로써 한국어 단위화를 위한 새로운 방법을 제시하였다. 이 방법은 규칙에 기반하여 단위화를 처리하고, 규칙에 의해 추정된 단위를 기억 기반 학습으로 검증하였다. 기억 기반 학습이 규칙의 예외를 잘 처리할 수 있기 때문에 규칙의 예외에 대한 결정을 도와줌으로써 규칙을 보조하였다.

STEP 2000 데이터에 대한 실험 결과, 제시된 방법은 규칙 기반 방법보다 2.32, 기억 기반 학습보다 2.81만큼 F-score가 향상되었다. 구 단위화에 대해 지금까지 가장 좋은 성능을 보여온 기계학습 기법인 SVM과 비교하였을 때에도 F-score는 1.65만큼 향상되었다. 이 성능 향상은 한국어의 네 가지 종류의 구 중에서 주로 명사구에서 만들어졌다. 이는 규칙의 오류가 대부분 명사구와 관련되어 있기 때문이다. 명사구의 오류가 많기 때문에, 기억 기반 학습은 명사구에 대한 규칙의 오류를 잘 처리할 수 있었다. 또한 언제 규칙을 적용하고 언제 기억 기반 학습을 적용해야 하는지를 결정하는데 사용되는 임계치 τ 값을 실험적으로 결정하였다.

제시된 방법은 충분히 일반적이기 때문에, 품사 태깅

[20]이나 전치사 접속 결정[28]과 같은 다른 자연언어처리 문제에 적용될 수 있다. 기억 기반 학습이 이런 문제에서 비교적 높은 성능을 보이기는 했으나, 최고의 성능을 보이지는 않았다. 우리는 제시된 결합 방법이 성능을 최고 수준까지 높일 수 있을 것으로 기대한다.

참 고 문 헌

[1] L. Ramshaw and M. Marcus, "Text chunking using transformation-based learning," In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pp. 82-94, 1995.

[2] S. Argamon, I. Dagan, and Y. Krymolowski, "A memory-based approach to learning shallow natural language patterns," In *Proceedings of COLING/ACL 98*, pp. 67-73, 1998.

[3] T. Kudo and Y. Matsumoto, "Use of support vector learning for chunk identification," In Proceedings of the 4th Conference on Computational Natural Language Learning, pp. 142-144, 2000.

[4] T. Zhang, F. Damerau, and D. Johnson, "Text chunking using regularized Winnow," In Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics, pp. 539-546, 2001.

[5] G. Zhou and J. Su, "Error-driven HMM-based chunk tagger with context-dependent lexicon," In *Proceedings of EMNLP/VLC-2000*, pp. 71-79, 2000.

[6] M. Shibatani, *The Languages of Japan*, Cambridge University Press, 1990.

[7] V. Cherkassky and F. Mulier, *Learning from Data: Concepts, Theory, and Methods*, John Wiley & Sons, Inc., 1998.

[8] 김미영, 강신재, 이종혁, "규칙과 어휘정보를 이용한 한국어 문장의 구묶음(Chunking)", *제12회 한글 및 한국어 정보처리 학술대회 논문집*, pp.11-17, 2000.

[9] 신효필, "최소자원 최대효과의 구문분석", *제11회 한글 및 한국어 정보처리 학술대회 논문집*, pp. 242-244, 1999.

[10] W. Daelemans, A. Bosch, and J. Zavrel, "Forgetting exceptions is harmful in language learning," *Machine Learning*, Vol. 34, No. 1, pp. 11-41, 1999.

[11] J.-T. Yoon, K.-S. Choi and M.-S. Song, "Three types of chunking in Korean and dependency analysis based on lexical association," In *Proceedings of the 18th International Conference on Computer Processing Languages*, pp. 59-65, 1999.

[12] 박성배, 장병탁, "최대 엔트로피 모델을 이용한 텍스트 단위화 학습", 제13회 한글 및 한국어 정보처리학술대회 논문집, pp. 130-137, 2001.

[13] Y.-S. Hwang, H.-J. Chung, Y.-J. Kwak, S.-Y. Park, and H.-C. Rim, "Shallow Parsing by Weighted Probabilistic Sum," In *Proceedings of the 19th International Conference on Computer Processing Languages*, pp. 236-241, 2001.

[14] M. Kay, "Algorithm Schemata and Data Structures in Syntactic Processing," In *Readings in Natural Language Processing*, pp. 35-70, Morgan Kaufmann, 1970.

[15] 김기철, 이기오, 이용석, "형태소 분석 주도의 한국어 복합동사 처리", *정보과학회 논문지*, 제22권, 제9호, pp. 1384-1393, 1995.

[16] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, Vol. 13, pp. 21-27, 1967.

[17] W. Daelemans, J. Zavrel, K. Sloot, and A. Bosch, "TiMBL: Tilburg Memory Based Learner, version 4.1, Reference Guide," Technical Report ILK 01-04, Tilburg University, 2001.

[18] R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[19] A. Danyluk and F. Provost, "Small disjuncts in action: Learning to diagnose errors in the local loop of the telephone network," In *Proceedings of the 10th International Conference on Machine Learning*, pp. 81-88, 1993.

[20] W. Daelemans, J. Zavrel, P. Berck, and S. Gillis, "MBT: A memory-based part of speech tagger-generator," In *Proceedings of the 4th Workshop on Very Large Corpora*, pp. 14-27, 1996.

[21] E. Brill, "Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging," *Computational Linguistics*, Vol. 21, No. 4, pp. 543-566, 1995.

[22] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," In *Proceedings of the 13th International Conference on Machine Learning*, pp. 148-156, 1996.

[23] S. Abney, R. Schapire, and Y. Singer, "Boosting applied to tagging and PP attachment," In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pp. 38-45, 1999.

[24] 최기선, 남영준, 김진규, 한영균, 박석문, 김진수, 이춘택, 김덕봉, 김재훈, 최병진, "한국어정보베이스를 위한 형태, 통사 태그 표준에 관한 연구", *인지과학*, 제7권, 제4호, pp. 43-61, 1996.

[25] CoNLL, *Shared Task for Computational Natural Language Learning (CoNLL)*, http://lcg-www.uia.ac.be/conll2000/chunking, 2000.

[26] T. Joachims, "Making large-scale SVM learning practical," Technical Report LS8, Universitaet Dortmund, 1998.

[27] B. Scholkopf, C. Burges, and A. Smola, *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999.

[28] J. Zavrel, W. Daelemans, and J. Veenstra, "Resolving PP attachment ambiguities with memory-based learning," In *Proceedings of the Conference*

on Computational Language Learning, pp. 136-144, 1997.

박 성 배

1994년 2월 한국과학기술원 전산학과(학사). 1996년 2월 서울대학교 컴퓨터공학과(석사). 2002년 8월 서울대학교 전기컴퓨터공학부(박사). 2004년 3월~현재 경북대학교 컴퓨터공학과 전임강사. 관심분야는 기계학습, 자연언어처리, 정보검색, 바이오인포매틱스

장 병 탁

1986년 2월 서울대학교 컴퓨터공학과(학사). 1988년 2월 서울대학교 컴퓨터공학과(석사). 1992년 7월 독일 Bonn 대학교 컴퓨터과학과(박사). 1992년 8월~1995년 8월 독일국립정보기술연구소(GMD) 연구원. 1995년 9월~1997년 2월 건국대학교 컴퓨터공학과 조교수. 1997년 3월~현재 서울대학교 컴퓨터공학부 조교수, 부교수. 2001년 1월~현재 서울대학교 바이오정보기술연구센터(CBIT) 센터장. 2002년 6월~현재 과기부 국가지정 바이오지능연구실 실장. 관심분야는 Bio-intelligence, Molecular Computation, Learnable and Evolvable Machines