영한 기계번역에서 긴 문장의 구문 분석 정확성 향상을 위한 쉼표의 용도 분류

김성동, 박성훈[0]
한성대학교 컴퓨터공학과
{sdkim, garion}@hansung.ac.kr

Comma Usage Classification for Improving Parsing Accuracy of Long Sentences in English-Korean Machine Translation

Sung-Dong Kim, Sung-Hoon Park[0]
Dept. of Computer Engineering, Hansung University

요 약

영한 기계번역에서 긴 문장은 분석 복잡도가 높아서 정확하게 분석하기 어렵다. 본 논문에서는 영어 구문 분석의 정확성을 향상시키기 위해서 긴 문장을 구성하는 쉼표의 역할을 자동적으로 판단하는 방법을 연구하였다. 쉼표는 긴 문장을 구성할 때 많이 사용되며 하나의 긴 문장을 만들 때 다양한 역할을 한다. 긴 문장을 분석할 때 쉼표에 의해 분할되는 부분을 독립적으로 분할하고 쉼표의 역할에 따라 분석된 결과를 적절하게 결합한다면 보다 빠르고 정확하게 주어진 문장 구조를 얻을 수 있다. 본 논문에서는 쉼표의 용도가 표시된 말뭉치로부터 분포 차이를 이용하여 쉼표 분류 규칙을 생성한다. 실험을 통해 논문에서 제시한 방법과 다른 학습 방법에 의한 쉼표 분류의 정확도를 비교하여 본 논문에서 제시한 방법이 실용적 가치가 있음을 보인다.

1. 서 론

인터넷을 통해서 사람들은 점점 더 많은 영어 문서를 접하게 되며 보다 많은 사람들이 보다 용이하게 영어 문서로부터 정보를 획득하게 하기 위해서 영한 기계번역 시스템의 필요성이 날로 증가하고 있다. 이를 위한 영어 구문 분석기는 실제로 영어 문서에 많이 나타나는 긴 문장을 적절하게 분석할 수 있어야 한다. 그러나 문장의 길이가 길어질수록 분석이 어려워져서 의미를 제대로 전달할 수 있는 번역이 어려워진다. 이를 극복하기 위해서 긴 문장을 보다 짧은 분할로 나누어 분석하는 방법들이 제안되었다. Abney는 문장을 chunk라는 단위로 나누어 이를 분석하는 방법을 제안하였다 [1]. 이후로 영어-중국어 번역 [2], 일본어-영어 번역 [3], 영어-한국어 번역 [4,5] 등 많은 영역에서 문장 분할 방법이 기계번역의 다양한 분야에 적용되었다.

이러한 연구들은 주로 구문 분석의 효율성 향상을 목적으로 한 것이었으며 구문 분석의 정확성에 대해서는 많은 고려를 하지 않았다. 최근에 컴퓨터 하드웨어(특히 CPU)의 빠른 성능 향상으로 인해 구문 분석기(파서)의 부담이 많이 줄어들고 있다. 이로 인해 번역의 정확성이 극복되어야 할 문제로 대두되고 있는 실정이다. 긴 문장은 효율성 문제뿐만 아니라 정확한 번역을 얻는데도 많은 어려움이 있다. 그런데 긴 문장들은 주로 쉼표(comma)에 의해 여러 문장 구조가 연결되며 쉼표는 문장에서 문장을 구성하는데 특정한 역할을 한다. 즉 쉼표는 여러 개의 의미 있고 독립적인 구나 절을 연결하여 하나의 문장을 만들 수 있다. 따라서 효율적인 구문 분석을 위해서 쉼표에 의해 독립적으로 분석될 수 있는 문장의 부분을 찾을 수 있다. 문장의 각 부분을 독립적으로 분석한 후 정확한 번역을 얻기 위해서 분석 결과들은 적절하게 결합되어 하나의 문장 구조를 표현해야 한다. 쉼표의 역할에 대한 지식을 이용한다면 적절하게 분석 결과를 결합할 수 있으며 결과적으로 정확한 번역을 생성할 수 있을 것이다.

영어 문법에서 쉼표는 여러 가지 용도로 사용된다. 본 논문에서는 쉼표의 용도를 조사하고 이를 기계번역의 입장에서 재조명 한다. 그리고 쉼표의 용도를 자동적으로 분류하는 방법을 제시하고 이를 이용한 구문 분석 방법을 소개한다. 긴 문장을 번역할 때 쉼표에 관한 추가적인 지식이 파서(parser)에 결합되어 번역의 품질 향상에 기여한다. 본 논문에서는 쉼표의 용도가 표시된 말뭉치를 구축하고 이로부터 쉼표의 용도를 분류하는 규칙을 생성한다. 쉼표의 용도를 판단하기 위해 여러 속성(attribute)를 고려하며 각 용도마다 속성 값의 분포를 계산한다. 다른 용도와 특징적인 차이를 보이는 속성의 값의 영역을 찾고 이를 이용하여 쉼표의 용도를 분류하는 규칙을 생성한다. 실험을 통해서 본 연구에서 구축한 규칙에 의한 쉼표 용도 분류 성능과 다른 기계학습 방법들에 의한 분류 성능을 비교하였다.

2절에서는 영어 문법에서 나타나는 일반적인 쉼표의 역할을 조사하고 쉼표의 용도에 따른 구문 분석 방법을 제시한다. 3절에서는 쉼표 용도 분류 규칙의 획득 방법을 상세하게 설명하고 4장에서는 쉼표 용도 분류의 정확성을 다른 방법과의 비교를 통해 보여준다. 5장에서는 논문을 결론지으며 앞으로의 과제를 제시한다.

2. 쉼표의 용도

2.1 영어 문법에서의 쉼표의 용도

영어 문장을 작문할 때 적절하게 쉼표를 사용하면 독자가 글쓴이의 의도를 보다 쉽고 정확하게 이해할 수 있다. 그러므로, 기계번역에서도 쉼표 용도에 대한 지식이 있다면 보다 자연스럽고 이해하기 쉬운 번역을 생성할 수 있다. 이 절에서는 쉼표의 역할과 영어 작문을 할 때 쉼표를 적절하게 사용하기 위한 지침을 소개한다 [6, 7].

(1) 절(clause)를 구분하기 위해서 쉼표를 사용한다. 특히 긴 문장의 경우에는 반드시 쉼표를 사용한다.
예) We worked for several hours debating the proposition, but it became clear we would never agree.

(2) 항목을 나열할 때 항목을 분리하는 쉼표를 사용한다.
예) The characters within the play included dragons, warriors, and maidens.

(3) 대등하게 나열되는 형용사를 분리하기 위해 쉼표를 사용한다.
예) His laughter was obnoxious, offensive, and hysterical.

(4) 문장 첫머리의 긴 구(phrase)나 절을 본 문장과 분리하기 위해 쉼표를 사용한다.
예) With the worse of her ordeal yet to come, Barbara decided to forgo her lunch.

(5) 두 개의 독립적인 절을 분리하기 위해서 쉼표와 대등접속사[3](coordinating conjunction)를 사용한다.
예) The public seems eager for some kind of gun control legislation, but the congress is obviously too timid to enact any truly effective measures.

(6) 날짜, 지명, 주소 등을 쓸 때 쉼표를 사용한다.
예) Shelly signed the contract on Wednesday, March 24, 2002.

(7) 삽입 요소(parenthetical elements)들을 분리하기 위해 쉼표를 사용한다. 이것이 가장 어려운 쉼표의 용도이다. "삽입 요소"란 문장의 본질적인 의미를 손상함이 없이 제거될 수 있는 문장의 부분을 말하며 "추가 정보(added information)"이라 부른다.
예) 동격절 (appositive phrase)
Robert Frost, America's most beloved poet, died when he was 88.

예) 감탄사 (interjection)
It is always a matter, of course, of preparation and attitude.
예) 호격 (vocatives)
I'm telling you, Juantita.

(8) 인용부호 안에 쉼표를 쓴다.
예) His accent reminds me of the "out back of Australia," which is where I'd like to be right now.

우리는 위의 8가지 쉼표의 용도를 크게 세 가지로 분류하였다. 첫째, 요소를 나열하기 위한 쉼표로서 (2), (3)이 이에 해당한다. 둘째, 문장을 독립적인 구와 절로 분할하기 위한 쉼표로서 (1), (4), (5), (7), (8)이 이에 해당한다[1]. 마지막으로 (6)의 용도는 쉼표로 구성된 특수한 패턴으로 취급한다. 이들 패턴은 구문 분석 이전에 인식하기 때문에 본 논문에서는 다루지 않는다.

2.2 구문 분석 방법 (Parsing Method)

영한 기계번역에서 우리는 차트 기반의 문맥 자유 파서(chart-based context-free parser)를 이용한다. 문장에서 쉼표로 구분되는 요소는 **분할(segment)**로 지칭하고 쉼표 왼쪽의 요소는 **전분할(preceding segment)**, 오른쪽 요소는 **후분할(next segment)**이라 부른다. 쉼표를 가진 문장을 분석할 때, 분할은 하나의 단어, 구, 또는 절이 될 수 있으며 파서는 각 분할을 분석한다. 쉼표의 용도에 따라 다른 구문 분석 방법이 적용되어 문장의 구조를 생성한다. 이 절에서는 쉼표 용도에 따른 구문 분석 방법을 간단하게 설명한다.

초기에 차트는 각 단어에 대한 간선(edge)과 정점(vertex)으로 구성된다. 우선 요소를 나열하는 용도로 사용된 쉼표로 분리된 분할들을 분석하는 방법은 다음과 같다. 첫 번째 쉼표의 후분할부터 마지막 쉼표의 전분할까지 각 분할들을 독립적으로 하나씩 분석한다. 각 분할의 분석 결과는 차트에서 하나의 간선으로 나타난다 [8]. 다음에 첫 번째 쉼표의 전분할과 마지막 쉼표의 후분할에 대해 접속자(conjunct) 찾기 알고리즘을 적용한다 [9]. 알고리즘을 통해 발견된 접속자를 각각의 분할에서 분리하여 분석하고 마찬가지로 분석 결과인 간선을 차트의 적절한 위치에 추가한다. 이 단계까지 마치면 차트에는 쉼표가 연결하는 각 요소들에 대해서 하나씩의 간선이 존재하며 나머지 문장의 단어들에 대한 간선이 존재하게 된다. 이제 차트에 대해서 정상적인 차트 파싱을 수행하여 전체 문장의 결과 구조를 생성한다.

분리 용도로 사용된 쉼표에 의해 분할된 전분할과 후분할은 각각 독립적으로 분석되고 분석 결과는 차트에서 간선으로 나타난다. 차트에는 분할에 대한 간선이 존재하며 이 차트에 대해 정상적인 파싱을 적용하여 결과 구조를 생성한다. 분할들간의 관계가 문맥 자유 문법으로 충분히 기술될 수 있으므로 정상적인 파싱을 적용할 수 있다. 서로 다른 용도로 사용된 쉼표를 가지는 문장의 경우 보다 복잡한 파싱 방법이 필요하며 이는 앞으로의 과제로 결론에서 제시한다. 그림 1, 2는 기본적인 파싱 과정을 보여준다.

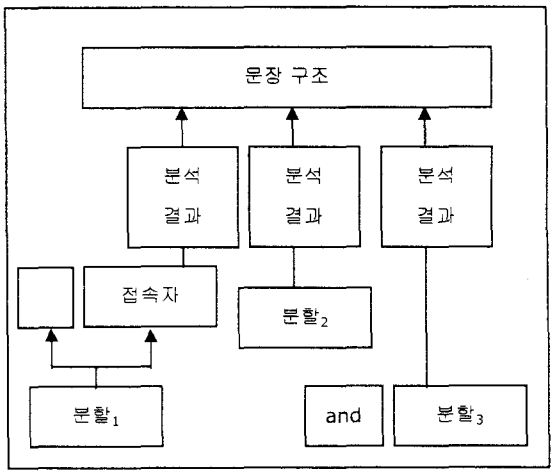

그림 1. 요소를 나열하는 쉼표에 의한 분할을 파싱하는 방법.

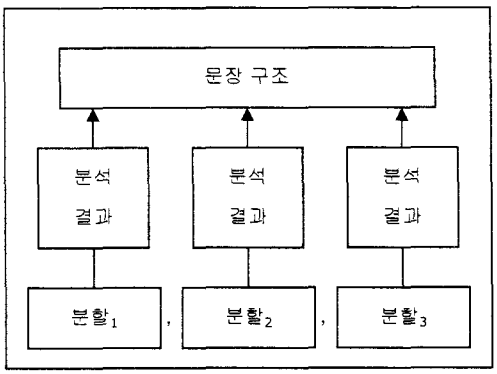

그림 2. 분리 용도로 사용된 쉼표에 의한 분할을 파싱하는 방법.

3. 쉼표 용도 분류 규칙의 생성

본 절에서는 쉼표의 용도가 부착된 말뭉치를 이용하여 쉼표의 용도를 분류하는 규칙을 생성하는 방법을 설명한다. 본 논문에서는 구문 분석의 관점에서 쉼표의 용도를 두 가지로 분류한다: 나열(listing), 분리(separation). 즉 본 연구에서는 두 개의 부류에 대한 분류 문제를 다룬다. 쉼표의 용도를 결정하기 위해서 여러 가지 속성들을 고려한다. 각 쉼표에 대해 다음과 같은 속성의 값을 수집하여 학습 데이터를 구성하였다: 문장에 나타난 쉼표의

[1] (8)의 경우 쉼표가 문장 다시 쓰기(rewriting)를 통해 인용부호 밖으로 나오게 될 수 있기 때문에 분할을 위한 쉼표라 간주할 수 있다.

개수 (N_T), 쉼표의 순서값 (N_O), 전분할과 후분할의 첫번째와 마지막 단어의 품사[2] ($POS_P^1, POS_P^n, POS_N^1, POS_N^n$), 후분할의 첫 두 단어 ($w_N^1, w_N^n$), 전분할에 인용부호의 존재여부 (*QM*), 후분할에 대등 접속사의 존재여부 (*CC*), 전분할과 후분할의 길이 (*LP, LN*). 그림 3은 학습 데이터의 구성을 보여준다.

Cat	N_T	N_O	$(POS_P^1, POS_P^n, POS_N^1, POS_N^n)$

(w_N^1, w_N^n)	QM	CC	$LP,\ LN$

그림 3. 학습 데이터의 구성.

그림 4는 말뭉치에 있는 문장의 예와 이로부터 구성한 구체적인 학습 데이터의 모습의 예를 보여준다. 말뭉치에 있는 모든 쉼표에 대해서 데이터를 생성하고 쉼표의 용도에 따라 두 가지 데이터 집합을 구축하였다.

The fasteners,/L nuts and bolts,/S are sold to the North American auto market.

(*list* 2 0 DET NOUN NOUN NOUN *nuts bolts* 0 1 2 3)

(*separation* 2 1 NOUN NOUN VERB NOUN *are market* 0 0 3 8)

그림 4. 말뭉치에서의 문장과 실제 데이터의 예.

규칙 생성의 전체적인 과정은 그림 5와 같다.

속성의 값의 분포를 두 가지 데이터 집합 각각에 대해서 다음의 식을 이용하여 계산한다.

$$p_{att}(val_k) = \frac{val_k}{\sum_{i=1}^{n} |val_i|} \qquad (1)$$

식 (1)에서 $p_{att}(val_i)$은 속성 *att*의 *i*번째 값 val_i의 분포이다. 즉, 속성 값들을 확률 변수(random variable)로 간주하고 그들의 확률 분포를 계산한다. 모든 속성 값에 대해서 두 가지 데이터 집합에서의 분포를 비교한다. 분포 차이를 이용하여 분류 규칙 구성에 사용되는 속성 값을 선택한다. 선택된 특성(feature)들의 집합(feature set: *FS*)은 식 (2)와 같이 나타낼 수 있다.

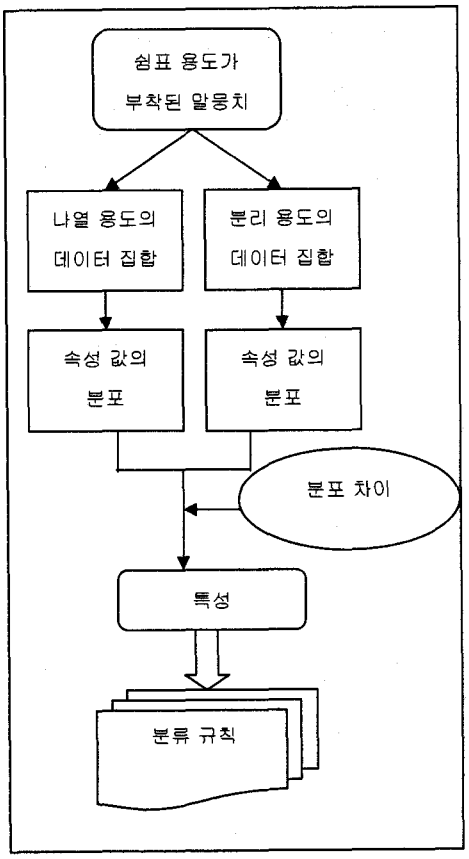

그림 5. 규칙 생성의 전체적인 과정.

$$FS = \{f_{att} \mid P_{C_1}(f_{att}) > P_{C_2}(f_{att}) + \varepsilon\} \qquad (2)$$

(식) 2에서 f_{att}는 속성의 값이고, $p_{C1}(f_{att})$은 데이터 집합 *C1*에서 값 f_{att}의 확률이며 $p_{C2}(f_{att})$는 데이터 집합 *C2*에서의 확률이다. 상수 ε은 특징있는 특성을 결정하기 위해 사용되었다.

분류 규칙은 식 (2)의 *FS*의 집합에 있는 각각의 특성을 가지고 다음과 같이 만들어진다.

if (val_{att} == $f_{att} \in FS_{C1}$)
then $RS(r_i) = p_{C1}(f_{att}) - p_{C2}(f_{att})$
else $RS(r_i) = 0$.

모든 규칙은 규칙의 점수(rule score: $RS(r_i)$)를 가진다. 총 점수를 가지고 쉼표의 용도를 결정하는데 분류 함수(classification function)은 다음과 같다.

[2] 윗첨자 1은 첫번째 단어, n은 분할의 길이가 n이라고 할 때 마지막 단어를 나타낸다. 아래첨자 P는 전분할, N은 후분할을 나타낸다.
[3] 예를 들어, and, but, or 등의 접속사를 지칭한다.

$$f(comma) = \begin{cases} C1, & \text{if } \sum_{i=1}^{n} RS(r_i) > 0 \\ C2, & \text{otherwise} \end{cases} \quad (3)$$

4. 실 험

실험을 통해 분포 차이를 이용하여 구축한 쉼표 용도 분류 규칙의 정확도(accuracy)를 보인다. 우선 쉼표의 용도가 표시된 말뭉치를 모으고 여기서 각 쉼표의 용도에 대한 학습 데이터 집합을 구축하였다: 나열, 분리. Wall Street Journal의 문장을 이용하여 말뭉치를 구축하였으며 표 1은 말뭉치와 데이터 집합의 통계를 보여준다.

표 1. 태그된 말뭉치와 데이터 집합.

	총 개수
말뭉치	3000
나열 용도의 데이터 집합	1628 (23.6%)
분리 용도의 데이터 집합	5257 (76.4%)

즉, 3,000 문장에서 나열을 위해 사용된 쉼표 1,628개와 분리를 위해 사용된 5,257개의 쉼표에 대해 특성 값을 계산하여 학습 데이터 집합(C1, C2)을 구성하였다. 학습된 규칙의 분류의 정확성은 다음의 식으로 계산한다.

$$accuracy = \frac{|f(comma) == Category(comma)|}{|f(comma)|} \times 100(\%) \quad (4)$$

식 (4)에서 *Category(comma)*는 올바른 쉼표의 용도이다. Wall Street Journal의 250 문장에 있는 613개의 쉼표, AP 말뭉치의 250 문장에 있는 570개의 쉼표, 그리고 Byte Magazine의 200 문장에 있는 462개의 쉼표에 대해서 분류 실험을 수행하였다. 표 2에 평가 결과가 나타난다.

표 2. 쉼표 용도 분류의 정확도 (%).

방법	Byte	WSJ	AP	평균
Baseline	76.4	76.4	76.4	76.4
분포차이	87.0	88.2	85.7	87
결정 트리	85.3	86.9	86.5	86.1
신경망	84.8	84.5	85.2	84.8

성능 비교를 위해서 결정 트리(decision tree) 방법으로 학습한 쉼표 용도 분류 규칙과 신경망(neural network)으로 학습한 쉼표 용도 분류 함수를 이용하여 쉼표 분류 정확도를 측정하였다. 표 2에서 'Baseline' 방법은 모든 쉼표의 용도를 '분리(separation)'로 예측한다고 가정하였을 때의 정확도이다. 결정 트리 학습은 C4.5 [10]를 이용하였으며 신경망은 2층 피드포워드(two-layered feedforward) 구조를 이용하였다. 물론, 분포 차이를 이용한 방법과 같은 학습 데이터를 사용하였다. 표 2에서 보는 바와 같이, 본 논문에서 제안한 분포 차이를 이용하여 학습한 분류 규칙이 가장 좋은 성능을 보였다. 또한 학습 데이터를 구성할 때 사용한 말뭉치와 다른 영역에 대한 테스트 결과를 통해 이 방법이 말뭉치의 영역과 무관함을 알 수 있다.

쉼표의 용도에 대한 예측이 없다면 분할들은 오른쪽에서 왼쪽으로 분석된다. 하나의 분할이 분석될 때 그것의 오른쪽 분할의 분석 결과는 차트에서 하나의 간선을 차지하게 된다. 첫 번째 분할이 분석된 후에야 전체 문장의 구문 분석 결과가 완성된다. 쉼표 용도를 예측할 수 있다면 보다 개선된 번역 결과를 얻을 수 있다. 나열을 위한 쉼표를 가지는 문장을 분석할 때, 모든 접속자(conjunct)를 판별하고 이들을 개별적으로 분서함으로써 보다 정확한 번역을 얻게 된다. 그렇지 않다면 보통 마지막 접속자가 독립적인 구나 절로 번역된다. 또한 분리 용도의 쉼표를 가지는 문장을 번역할 때, 분할의 분석 결과들은 그들의 문법 부류(syntactic category)에 따라 적절하게 결합될 수 있어서 마찬가지로 보다 자연스러운 번역을 생성할 수 있다. 예를 들어, 쉼표로 분리된 명사구(noun phrase)는 앞의 단어의 동격(*appositive*)이 될 수 있고, 관계절(relative clause)은 앞에 있는 명사구를 뒤에서 수식하는 역할을 할 수 있다. 이처럼 쉼표의 용도에 대한 지식을 활용함으로써 구문 분석의 복잡도를 줄여 보다 빠른 분석이 가능하며 또한 보다 정확한 번역을 얻을 수 있다.

5. 결 론

본 논문에서는 쉼표의 용도 분류를 위한 규칙을 생성하는 방법을 제안하였으며 이를 통해 쉼표를 가지는 긴 문장을 보다 정확하게 번역하고자 하였다. 분류 규칙을 생성하는데 있어서 본 논문에서는 규칙을 구성하는 특성을 선택하기 위해 '분포 차이'라는 비교적 단순한 방법을 적용하였다. '분포 차이'라는 방식을 통해 쉼표의 용도를 예측하는데 고려된다고 판단되는 비교적 명백한 특성을 선정할 수 있었으며 다른 기계학습 방법들과 비교하였을 때 우수한 예측 성능을 보였다. 분포 차이에 의한 특성 선택 방법을 이용하면 규칙을 빠른 시간에 구축할 수 있으며 또한 신경망 학습을 위한 매개변수의 개수를 줄일 수 있다. 이 방법은 새로운 후보 속성들을 쉽게 추가할 수 있다는 점에서 확장가능(scalable) 하다고 할 수 있다. 이 방법은 전치사 접속(preposition attachment) 문제 같은 두 부류(two classes) 분류 문제에 유효하다.

본 논문에서는 긴 문장을 보다 정확하게 번역하기 위해

예측된 쉼표의 용도에 따라 구문 분석하는 방법들을 제시하였다. 쉼표 용도에 대한 추가적인 정보를 통해 부분적으로 분석된 결과들을 보다 정확하게 결합할 수 있으며 이는 번역의 정확성 향상에 기여한다. 분류의 정확성을 개선하기 위해서는 보다 많은 특성들을 고려해야 할 것이다. 그리고 접속자 확인 방법과 분리된 분할들의 역할 파악 방법에 관한 연구가 보다 정교한 구문 분석 알고리즘을 위해 앞으로 진행되어야 한다. 이는 결과적으로 영한 번역 시스템의 성능 향상에 기여할 것이다.

참고문헌

[1] S. Abney, "Parsing by Chunks," in Principle-Based parsing, Kluwer Academic Publishers, pp. 257-278, 1991.
[2] Wei-Chuan Li et al, "Parsing Long English Sentences with Pattern Rules," in Proceedings of 1990 COLING, pp. 410-412, 1990.
[3] Y.B. Kim and T. Ehara, "A method for Partitioning of Long Japanese Sentences with Subject Resolution in J/E Machine Translation," In Proceedings of the 1994 COLING, pp. 467-473, 1994.
[4] Sung-Dong Kim, et al, "Learning-Based Intrasentence Segmentation for Efficient Translation of Long Sentences," in Machine Translation, Vol. 16, pp. 151-174, 2001.
[5] Yoon-Hyung Roh et al, "For the proper treatment of long sentences in a sentence pattern-based English-Korean MT system," MT Summit IX, pp.323-329, 2003.
[6] University of Washington, Sociology Writing Center, "Comma Usage," http://staff.washington.edu/writesoc/PDF_Files/Commas.pdf
[7] Charles Darling, "Rules for Comma Usage," http://webster.commnet.edu/grammar/commas/htm
[8] T. Winograd, "Language as a Cognitive Process:Syntax, Vol.1," Addison-Wesley, 1983.
[9] R. Agarwal and L. Boggess, "A simple but Useful Approach to Conjunct Identification," In Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics, pp. 15-21, 1992.
[10] R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann, 1992.