제21회 한국정보처리학회 춘계학술발표대회 논문집 제11권 제1호 (2004. 5)

분석의 최종 판단자로서의 구문 분석기

여상화*
*경인여자대학 컴퓨터정보기술학부
e-mail : shyuh@kic.ac.kr

Parser as An Analysis Finisher

Sang Hwa Yuh*
** Div. of Computer Information Technology, Kyung-In Women's College

요 약

통상적인 언어 처리의 분석 과정은 전처리, 형태소분석, 품사 태깅, 복합 단위 인식, 구문 분석,그리고 의미 분석 등의 여러 단계로 이루어진다. 분석의 매 단계에서 중의성(Ambiguity)가 발생하며, 이를 해결하기 위한 노력으로 구문 분석 이전의 분석 단계에서도 정확률(Precision)을 높이기 위해, 어휘(Lexical) 정보, 품사정보 그리고 구문 정보 등을 이용한다. 각 단계에서 고급 정보로서의 구문 정보 이용은 구문분석의 중복성과 분석 지식의 중복성을 야기한다. 또한, 기존의 처리 흐름에서는 각 분석 단계에서의 결과는 최종적인 것으로, 이로 인해 다음 분석 단계에 분석 오류를 전파한다.

본 논문에서는 구문 분석기를 분석 결과의 최종 판단자로 이용할 것을 제안한다. 즉, 구문 분석 전단계의 모든 분석 정보는 구문 분석기에 제공되고, 구문분석기는 상향식 구문분석을 수행하면서 이들 정보들로부터 최종의 그리고 최적의 분석 후보를 결정한다. 이를 위해 구문분석기는 한 문장 단위를 입력 받는 기존의 제한을 따르지 않는다. 제안된 방법은 구문분석 앞 단계에서의 잘못된 정보 제공(예: 문장 분리 오류, 품사 오류, 복합단위 인식 오류 등)으로부터 자유로우며, 이를 통해 분석 실패의 가능성을 최대로 줄인다.

1. 서론

통상적인 언어처리 과정은 전처리(Preprocess)를 통한 문장 분리와 분석 대상 어휘인 토큰(Token) 분리, 형태소 분석을 통한 원형 복원과 품사 정보 획득, 품사 태깅(Tagging)을 통한 최적 품사 결정, 숙어나 고정 표현을 포함한 복합 단위(Compound Unit) 인식을 통한 다중어(Multi-Word) 인식, 그리고 구문 분석을 통한 올바른 구문 구조 결정을 거친다. 이러한, 전통적인 접근 방법은 각 단계에서 분석의 부담을 나눠 가짐으로써 분석기의 부담을 줄이고, 모듈들의 개별적인 성능 향상을 가능하게 하여 대부분의 언어처리 시스템에서 적용하고 있다. 인간의 언어를 컴퓨터로 분석하는 일련의 분석 과정들은 중의성(Ambiguity)을 보다 적절히 해결하기 위하여 새로운 알고리즘을 도입하여 다양한 분석을 시도하며, 학습이나 분석의 주요 정보로서 보다 상위 단계의 고급 정보를 사용하거나 어휘(Lexical) 정보를 도입하고 있다. [9][10]에서는 전처리 과정에서의 문장 단위 인식을 위해, 문장 종결기호 전후의 단어들의 품사정보를 사용하며, [16]에서는 품사 태깅 과정에서 어휘 정보와 구문 정보를 사용하였으며, [4][5][12]에서는 복합 단위 인식 과정에서 가변 성분에 대한 구문 제약 정보를 사용하여 복합 단위 인식기의 정확률을 높이고 있다. 분석의 낮은 단계에서 보다 높은 정확성을 위해 상위 단계의 정보를 이용하려는 경향은 뚜렷하며, 이러한 시도는 앞 단계에서의 분석 실패가 상위의 다음 단계에서의 분석 실패를 야기하므로, 당연하다고 할 수 있다. 그러나, 이로 인해 기계번역 시스템과 같이 일련의 분석 과정을 포함하는 응용시스템에서는 각 단계에서의 중복적인 처리에 따른 오버헤드, 분석 지식의 과다, 일관성 결여와 중복성 문제 등이 발생한다.

또한, 기존의 분석 시스템에서는 각 분석 단계의 결과를 다음 단계에 최종적인 결과로 제공함으로써, 여러 분석 단계를 가치는 동안 분석 오류가 누적되어 전파된다는 문제점이 있고, 이로 인해 상위 단계의 분석 실패를 야기할 수 있다는 단점이 있다. 즉, 구문 분석 전 단계에서 문장 분리, 형태소분석, 품사 태깅,

복합단위 인식과 같은 4 단계의 분석 과정을 거치고, 각 분석 모듈의 정확률이 99%라면, 최종적으로 구문 분석기의 입력은 96.06%의 정확률을 가지게 되어, 약 4%의 오류를 가진 입력이 주어지게 된다. 이러한 오류들은 구문 분석기의 분석 성공률을 떨어뜨리고, 구문 분석기의 성능에 절대적으로 의존하는 많은 언어처리 응용 시스템(예: 자동번역기, 자연어 질의시스템 등)의 성능을 저하시키게 된다.

따라서, 본 논문에서는 구문분석기를 분석 결과의 최종 판단자로 이용할 것을 제안한다. 즉, 문장 분리를 포함한 구문 분석 전단계의 모든 분석 정보는 구문 분석기에 우선 순위를 가진 후보 정보들로서 제공되고, 구문분석기는 구문분석을 수행하는 과정에서 이들 정보들로부터 최종의, 최적의 분석 후보를 결정한다. 따라서, 본 논문에서 제안하는 구문분석기는 한 문장 단위를 입력 받는 기존의 제한을 따르지 않는다.

분석의 모든 단계에서는 상위 단계의 정보를 중복 처리 없이 사용할 수 있어 보다 높은 정확률을 기대할 수 있다. 또한, 단어의 수가 늘어나면 기하급수적으로 늘어나는 분석 시간을 줄이기 위해, 분석 과정에서 분리 가능한 문장으로 인식되면 이들을 분리하고 나머지 입력에 대해 분석을 계속해 나간다.

제안된 방법은 구문분석 앞 단계에서의 잘못된 정보 제공(예: 문장 분리 오류, 품사 오류, 복합단위 인식 오류 등)으로 인한 분석 실패부터 자유로울 수 있으며, 이를 통해 구문 분석 실패의 가능성을 최대로 줄인다.

2. 기존 분석시스템의 문제점

이산적이고(Discrete) 절차적인(Procedural) 기존의 언어처리 분석에서는 상위의 정보를 이용에 따른 오버헤드를 가지며, 다음 단계의 분석에서 치명적인 오류를 야기할 수 있는 오류의 가능성을 가진 분석 결과를 생성한다. 이러한 오류의 유형은 분석의 모든 단계에서 발생한다. 따라서, 구문 분석 전단계에서 여러 단계를 거치면서 누적된 오류로 인해 구문분석기의 분석 성공률은 저하된다. 각 분석 단계에서의 중의적인 현상을 살펴보면 다음과 같다.

2.1 문장 분리에서의 모호성

기존의 언어처리 시스템은 정확히 분리된 한 문장을 입력 단위로 가정한다[1]. 그러나, 실제 문서(Real Text)에서는 '.', '?', '!', ':' 등의 문장 종결기호만을 가지고 문장을 분리할 수 있는 것은 아니다. 마침표의 경우, 약 10% 정도는 문장 종결기호로 사용되지 않는다[14]. Etc., Calf.와 같이 마침표를 포함하는 단어가 문장의 마지막에 나타나면 별도의 문장 기호를 사용하지 않는다[2]. 따라서, 분석의 첫 단계인 문장 단위의 인식에서도 중의성이 발생하며, 잘못된 문장 단위 인식 결과는 구문 분석에서 치명적인 분석 실패를 야기하게 된다. 이를 해결하기 위한 노력으로, [1]에서는 정규 표현(Regular Expression)을 사용하였으며, [14].에서는 통계적인 분류 트리(Statistical Classification Tree)를 사용하였으며, [9] [10]에서는 마침표 전후 일부 단어들의 품사정보를 뉴럴 네트워크(Neural Network)로 학습시켜 문장 분리를 하였다. [Mikheev98]에서는 Maximum Entropy 를 사용하였으며 99.25%의 정확률을 보였다.

2.2 Token 분리와 형태소 분석의 모호성

영어의 형태소 분석에서는 주로 축약형, 대소문자, 하이픈(Line-Breaking Hyphen), 고유명사의 인식, 미등록어 등의 처리에서 중의성이 발생한다[2][8].

(예 1) Tom's (Tom is vs. Tom has)

(예 2) The New York-New Haven railroad

(예 1)에서는 's 를 is 로 볼 것인지, has 로 볼 것인지가 문제가 되며, (예 2)에서는 York-New 가 하이픈으로 연결되어 한 단어로 인식할 수도 있으며, New York 을 복합 단위로 인식할 수도 있다[8].

2.3 품사 태깅에서의 중의성

영어의 많은 단어들은 하나 이상의 품사를 가지는 다품사 어휘이며, 특히 거의 모든 명사 단어들은 또한 동사로 사용될 수 있다. 중의성 해결을 위한 품사 태깅 알고리즘으로는 HMM(Hidden Markov Mode), Neural Network, Transformational-based Learning, Maximum Entropy 모델 등이 사용되며, 현재, 최고 성능을 보이는 것으로는 [Ratnaparkhi96]의 Maximum Entropy 모델을 사용한 것으로 99%의 정확률을 보고하고 있다. 여러 모델의 혼합형(Hybrid) 모델을 사용하여 정확률을 높이거나[16], 구문 분석기의 입력으로 n-Best 후보를 제공하는 것이 일반적이다[3][16]

2.4 복합단위 인식에서의 모호성

복합단위란, 한 단위로 처리할 수 있는 Collocation, Idiom, 날짜 표현 등을 일컫는다[4][12]. 복합 단위 인식기는 구문 분석기의 입력의 수를 줄이고, 불필요한 중의성을 감소시켜 구문 분석기의 부담을 줄이기 위해 구문 분석 전단계에 둔다. 인식의 모호성으로 인해 대부분 구문분석기와 연동된다[12][17]. 이 과정에서 복합 단위의 인식 정확률을 높이기 위해 부분 파싱(Partial Parsing)을 적용하기도 한다[5][12]. [12]에서는 복합단위 인식에서 부분 파싱을 도입하여 97.65%의 재현율과 98.52%의 정확률을 보였다.

구문 분석 전단계에서의 각 분석 모듈들의 현재까지 보고된 최고의 성능을 취하면 구문 분석기의 입력은 표 2.1 과 같이 정확률이 91.56%로 낮아진다.

표 2.1 구문분석기 입력의 정확률

분석 모듈	정확률(State-of Art)	누적 정확률
문장 분리	99.25%[18]	99.25%
품사 태깅	97.13%[19]	96.40%
복합단위인식	98.52%[12]	91.56%
구문분석기의 입력의 정확률		91.56%

[1] 단편(Fragment)에 대한 분석 실패 시 Fail Safe 기능은 많은 시스템들이 가지고 있다

[2] 형태론에서는 이러한 현상을 Haplology 라고 한다.

따라서, 본 논문에서는, 각 단계에서 결정적이고 최종적인 분석 결과를 제공하는 것이 아니라, 구문 분석기로 하여금, 분석의 전 과정에서 제공된 정보를 제공받아, 구문 분석 과정에서 얻어지는 구문 정보를 이용하여 각 분석의 결과를 전역적으로 최종 판단하도록 할 것을 제안한다. 문장 분리, 형태소분석, 품사 태깅, 복합단위 인식기의 분석 결과는 구문분석기의 챠트(Chart)에 기록되며, 구문 제약 정보를 가지는 규칙들은 구문 분석 과정에서 적용된다.

3. 문장 분리의 필요성

자연언어를 위한 구문 분석기는 일반적인 시간 복잡도(Time Complexity)가 $O(n^3)$ (여기서 n 은 입력 토큰의 수)이다. 따라서, 입력의 길이가 길어질수록 분석 시간과 메모리 요구량이 급격히 증가한다. 본 논문에서 제안하는 구문분석기는 한 문장을 입력 단위로 가정하지 않으므로 입력의 길이가 기존의 구문분석기에 비해 길어질 수 있으며, 이전 분석에서의 결과를 최종적인 것으로 받아들이지 않으므로 분석 단계에서 보관해야 할 정보량이 많아지게 된다. 컴퓨터 하드웨어의 비약적인 발전으로 CPU 의 성능과 Memory 의 양은 충분할 수 있지만, 실용적인 시스템을 위해서는 분석 시간과 메모리를 고려하지 않을 수 없다.

따라서, 본 논문에서는 구문 분석 과정 중에 하나의 문장(또는 분리하여도 상관없는 부분 문장)을 인식하며, 이를 하나의 분석 결과로 출력하고 관련된 Active Item 과 Inactive Item 을 Chart 에서 제거한다. 이를 위해 Item List 들은 Best-First Parsing 을 위해 Priority Circular Queue 자료구조에 보관된다.

구문 분석 수행 중에 문장을 분리하는 기능은 본 논문에서 처음으로 제안하는 것으로, 기존의 시스템에서 Memory Fault 가 예견될 때, Chart 에서 Coverage 가 가장 넓은 분석 결과를 출력하는 Fail Safe 와는 다르다. 즉, 본 논문에서 제안하는 방식은, Bottom-Up 과 Top-Down 파싱을 수행하는 도중에 메모리 Fault 가 발생하지 않을 상황에서도 문장 분리가 이루어진다. 이는, 기계번역시스템을 최종 응용시스템으로 염두에 둔 전략이다. 즉, 기존의 모든 영한 번역시스템은 단어의 길이가 길어질수록 번역의 정확률이 급격히 떨어져 15 단어 이상의 장문의 번역률이 40% 미만이라는 실험결과에 바탕을 둔다. 즉, 장문의 입력 전체에 대한 분석은 불필요한 시간 낭비와 더불어 분석 정확률을 떨어뜨리는 주된 요인이다.

기존의 일부 시스템에서는 구분 분석 전단계에서 규칙에 기반하여 문장 분리를 시도하고 있으나 단문 분리의 재현율과 정확률이 각각 96%와 88%로 단문 분리 오류의 부작용(Side Effect)으로 인한 분석 실패율이 매우 높다[20]. 또한, 이들 시도는 구문 분석 전단계에 이루어져 구문 정보를 충분히 이용할 수 없다는 단점을 가지고 있다.

본 논문에서 제안하는 구문분석기는, 구문 분석 과정에서 Bottom-Up 분석을 통한 충분한 구문 정보를 이용하여 문장 분리를 시도함으로써 기존의 방법에 비해 단문 분리 오류를 크게 줄일 수 있다.

4. 분석 결과의 최종 판단자로서의 구문 분석기

본 논문에서 제안하는 영어 구문분석기는 전처리 과정부터 복합단위 인식기까지의 구문 분석 전단계에서 제공된 모든 분석 정보를 결정적이며 최종적인 정보로 사용하지 않으며 Priority 를 가진 분석 후보로 받아들이며, 구문분석을 통해 이전 분석 과정의 결과로 제공된 정보를 검증한다. 이를 통해, 이전 분석과정에서 제공되는 문장 분리, 품사 태깅, 복합단위 인식 결과 등을 구문 분석 과정을 통해 최종적으로 결정한다. 제안된 구문분석기는 통계적인 Best-First 통계적 Earley Parser 를 기반으로 제작된다[15].

4.1 Earley Parsing Algorithm

Earley Parsing 은 Active Chart Parsing 이라고도 하며 하향식(Bottom-Up)과 상향식(Top-down)을 동시에 수행하며, Prediction, Scanning 그리고 Completion 의 3 가지의 Transition 을 반복적용하며 분석한다. **Prediction** 은 현재의 상태가

$$i:_kX \rightarrow \lambda . Y \mu$$

일 때, 비 단말(Non-terminal) Y 로 시작하는 모든 문법 규칙을 찾아 i 번째 Column 에 기록한다. 이때 기록되는 Item

$$i:_iy \rightarrow .v$$

를 Predicted Item 이라고 한다. **Scanning** 은 현재 상태가,

$$i:_kX \rightarrow \lambda . a \mu$$

일 때(즉, Dot 다음 기호가 단말(Terminal 일 때) 현재의 입력 xi 와 일치하면 다음 Column 에

$$i+1:_kX \rightarrow \lambda a . \mu$$

를 추가한다. 이 때 추가되는 Item 을 Scanned Item 이라고 한다. **Completion** 은 다음과 같은 두 가지 Item 이 존재할 때 발생한다.

$$i:_jY \rightarrow v.$$
$$j:_kX \rightarrow \lambda . Y \mu \quad (j <= i)$$

즉, Dot 가 RHS(Right Hand Side)의 끝에 오는 Complete Item 이 존재하고, 이 Item 의 Origin 인 j 번째 Column 에 있는 Item 가운데 Dot 다음 기호가 Complete Item 의 LHS(Left Hand Side)인 경우 다음과 같은 Item 이 추가된다.

$$j:_kX \rightarrow \lambda Y . \mu$$

이 때 추가되는 Item 을 Complete Item 이라고 한다.

4.2 통계적 Earley Parsing

파스트리(Parse Tree) T_i 를 형성하는데 사용된 규칙들을 $r_1, r_2, \ldots, r_n$ 이라고 할 때, 파스트리의 확률 함수는 다음과 같다.

$$(\text{식 1}) \quad p(T_i \mid S) \sim= \prod_{j=1}^{n} P(r_i)$$

(단, S:주어진 입력문장, T_i:출력된 파스트리 중의 하나, r_j:파스트리를 구성하는 구문분석 규칙)
규칙의 확률은 다음과 같이 적용한다.

(식 2) $P(r_i)=P(X \rightarrow \lambda \mid P_{parent}(X \rightarrow \lambda) * \text{Trigram 확률}(a_0 a_1 a_2)$

Trigram 확률은 다음 수식에 의해 구해진다.

(식 3) P(a0a1a2) / P(a0xa2) (단, x 는 임의의 품사)

이 확률 모델은 Pearl 의 확률모델[11]을 단순화한 형태이다.

5. 구현 및 실험

본 논문에서 제안하는, 영어 구문분석기는 Visual C++6.0, 컴파일러 제작 도구인 ParserGenerator[15][16]]를 이용하여 구현되었다. 통계적인 영어 구 구조 규칙은 Penn Treebank Version 2 의 영어 Combined Corpus[3]중 Wall Street Journal(WSJ)의 00 부터 24 로부터 추출하였으며, 규칙의 개수를 줄이기 위해 Function Tag 와 Epsilon 을 제거한 후 추출하였다. 이를 통해 추출된 분석 규칙의 개수는 16,619 개이다. 이중에서 출현 빈도가 10 이하인 규칙을 제거하고 사용한다.

6. 결론 및 향후 연구

본 논문에서 제안하는 영어 구문분석기는 전처리 과정부터 복합단위 인식기까지의 구문 분석 전단계에서 제공된 모든 분석 정보를 결정적이며 최종적인 정보로 사용하지 않으며 Priority 를 가진 제안(Suggestion)으로 받아들이며 구문분석을 통해 이전 과정의 결과로 제공된 정보를 검증한다. 또한, 구문 분석 이전 단계의 분석 모듈에서 구문정보를 중복 처리 없이 제약 정보로 사용할 수 있도록 하며, 이를 통해, 이전 분석과정에서 제공되는 문장 분리, 형태소분석, 품사 태깅, 복합단위 인식 결과 등을 구문 분석 과정을 통해 최종적으로 결정한다. 분석의 첫 단계로서 문장 분리를 수행하는 전처리 과정과 같은 하위 수준에서도 중복처리 없이 구문 수준의 제약정보를 사용할 수 있도록 하여 분석 시스템의 정확률을 크게 향상시킨다.

이러한 특성에 기인하여, 본 논문에서 제안된 구문 분석기는, 문장 분리가 전혀 되어 있지 않거나, 또는 문장 종결 부호가 사용되지 않았거나 여러 문장이 한 문장으로 인식된 경우에도, 기존의 구문분석기와 달리 문장 분리를 겸하면서 구문 분석을 수행한다. 이러한 능력은 자동 번역(Machine Translation), 자연어 질의어 처리(Natural Language Query), 음성인식을 위한 언어 모델(Language Model) 등에서 유용하게 사용될 수 있다.

참고문헌

[1] 여상화, 정한민, 채영숙, 김태완, 박동인, “실용적인 영한 기계번역을 위한 전처리기의 설계 및 구현,” 제 8 회 한글 및 한국어 정보처리 학술대회 발표논문집, pp.313-321, 1996.

[2] 여상화, 정한민, 김태완, 박동인, 서정연, ”영한 기계번역을 위한 하이픈 단어의 전처리,” ‘97 한국정보과학회 가을학술발표논문집(II), pp173-176, 1997

[3] 여상화, 서정연, “구조변환을 겸한 영어 구문분석기”, 2003 년 봄 한국정보과학회 학술발표논문집(B), pp.507-509, 2003.

[4] 여상화, 서정연, “정규표현을 이용한 연속 및 불연속 복합단위 인식기”, 제 20 회 한국정보처리학회 추계학술발표논문집 제 10 권 제 2 호, 2003.

[5] 정한민, 여상화, 김태완, 박동인, “부분 파싱을 이용한 복합단위 인식,” ‘97 한국정보과학회 가을학술발표논문집(II), pp.223-226, 1997

[6] Aho A.V., and Ulman, *The Theory of Parsing, Translation and Computing*, Prentice Hall, 1972

[7] Bumble-Bee Software, http://www.bumblebeesoftware.com/

[8] Christopher D. Manning, Hinrich Schűtze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999

[9] David D. Palmer and Marti A. Hearst, “Adaptive Sentence Boundary Disambiguation,” In Applied Natural Language Processing, 4, pp.78-83, 1994

[10] David D. Palmer and Marti A. Hearst, Adaptive Multilingual Sentence Boundary Disambiguation, *Computational Linguistics*, No.23, pp.241-267

[11] David M. Magerman, Mitchell P. Marcus, “Pearl: A Probabilistic Chart Parser,” in Proceedings of 2nd International Workshop on Parsing Technologies, pp.193-199, 1991

[12] Hanmin Jung, Sanghwa Yuh, Taewan Kim and Dong-In Park, “Compound Unit Recognizer for Pattern-Based Approach to Multilingual Machine Translation,” Proceedings of PCALING’97, pp.167-172, 1997

[13] John R.Levine, Tony Mason, and Doug Brown, *Lex & Yacc*, O’Reilly & Associates, Inc, 1992

[14] Michael D. Riley, “Some Applications of the tree-based Modeling to Speech and Language Indexing,” In Proceedings of the DARPA Speech and Natural Language Workshop, pp.339-352, 1989

[15] Phillippe McLean, R.Nigel Horspool, “A Fast Earley Parser,” In Proceedings of International Conference on Compiler Construction, pp.281-293, 1996

[16] Sanghwa Yuh, et al., “NeuTag: A Hybrid Neural Network English Tagger with Pre-Fail Softener,” ICCPOL’99, 1999.

[17] Yoon, S, “Efficient Parser to find Bilingual Idiomatic Expressions for English-Korean Machine Translation,” In Proceedings of ICCPOL’94, 1994.

[18] Andrei Mikheev, “Feature Lattices for Maximum Entropy Modelling,” in ACL, vol.36, pp.848-854, 1998

[19] Adwait Ratnaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolution*, Ph.D. Dissertation. University of Pennsylvania, 1998.

[20]Sung Dong Kim, et al., “Reducing Parsing Complexity by Intra-Sentence Segmentation based on Maximum Entropy Model ,” *EMNLP200*, 2002

[3] Parsed Corpus 에 품사 Tag 가 부착된 Corpus