웹 인덱싱을 위한 통합 전처리 시스템의 개발[1)]

심 준혁, 차 정원, 이 근배

포항공과대학교 컴퓨터공학과 자연어 처리 연구실

경북 포항시 남구 효자동 산 31번지

Integrated Sentence Preprocessing System for Web Indexing

Junhyuk Shim, Jongwon Cha, Geunbae Lee

Natural Language Processing Lab.

Dept. of Computer Science & Engineering, POSTECH

{ nikkier, jwcha, gblee } @nlp.postech.ac.kr

요 약

웹 문서는 일반 문서들과 달리 자유로운 형식으로 기술되어 있고, 원문에 태그나 코드 등 불필요한 내용들을 많이 포함하고 있어 언어 처리에 바로 사용하기에 적합하지 못하다. 본 논문은 인덱싱 대상 문서로 사용되는 웹 문서를 자동으로 수집하여, 문장 단위로 정렬된 문서로 제작, 관리하는 통합 전처리 시스템인 Web Tagger의 구조와 전처리 방법을 소개한다. Web Tagger는 문서 정제, 문장 분할, 띄어쓰기의 과정을 거쳐 웹 문서에서 표준화된 정보를 추출하고, 형태소 분석기를 포함한 응용 시스템의 목적에 맞게 XML 형식의 원문 코퍼스를 자동으로 생성하고 관리한다. '정규문법(Regexp)', '휴리스틱', '품사 인덱스 참조', 'C4.5를 사용한 학습 규칙' 등의 다양한 전처리 기법은 형태소 분석 정확도 향상과 시스템 안정성 보장에 기여한다.

1. 서 론

최근 들어 인터넷이 확산되면서 WWW(World Wide Web)에 공유된 많은 양의 웹 문서는 점차 자연 언어 처리 작업을 위한 중요한 기본 자료로써 부각되고 있다. 웹 문서는 기존 여러 종류의 전자 문서와는 달리 가공하기 쉬운 동일한 문서 형식으로 작성되어 있다. 또, 웹 브라우저나 웹 로봇을 사용하여 자동으로 수집할 수 있으므로, 다량의 원시 코퍼스(Raw Corpus)를 구축하기 용이하다는 장점을 지니고 있다. 하지만, 정의되지 않은 HTML 태그가 증가하고 있고, 문서 내에 자바 스크립트 코드나 문서 서식을 위한 테그 정보 등의 많은 양의 부가 정보가 추가되고 있으며, 자유롭게 기술된 언어 표현이 증가하면서, 웹 문서는 점차 표준화되지 않은 문서가 증가하고 있는 추세이다. HOTBOT, HTDIC, WEBZIP, LIBWWW 등으로 대표되는 기존의 웹 로봇을 사용해 수집된 문서들에서 HTML 태그만을 제거하거나, 중복된 공백 문자열을 제거하는 등의 단순한 휴리스틱만으로 언어 처리 응용 시스템의 입력 데이터를 만들기에는 근본적인 한계가 있다. 본 연구에서는 다량의 웹 문서를 인덱싱하는 정보 검색 시스템의 입력 문서를 일괄적으로 제작하여 관리하는 웹 전처리 통합 솔루션으로 Web Tagger를 제작하였다. 본 시스템은 인터넷 상에 존재하는 웹 문서를 입력으로 받아 문서 웹 문서 수집, 문서 정제, 문장 분할, 띄어쓰기의 과정을 거쳐 웹 문서에서 표준화된 정보를 추출한 후 일괄적으로 구성하고, 형태소 분석기를 포함한 응용 시스템의 목

적에 맞게 XML 형식의 원문 코퍼스를 체계적으로 구축하고 관리한다.

2. 기존의 연구 분석

2.1. 관련 연구

문서 내에서, 문장은 형태소 분석, 정보 추출 기계 번역, 문서 요약, TTS(Text-to-Speech)를 위한 문서 정열 등, 모든 응용 시스템의 기본 처리 단위로 사용되고 있다. 언어 처리를 시작하는데 있어 입력 문서를 정제하고 일반화 시키는 작업은 필수적이지만 대부분 수작업으로 이루어져 왔다. 이 작업은 작게는 문서의 문단이나 문서의 문장을 분리하고, 제목과 키워드를 명시해 주어서 형식을 일관되게 해주는 작업부터, 크게는 문서 내에 존재하는 오타를 수정하고, 축약 복원하며, 띄어쓰기 오류 및 문장 분할 오류를 수정하는 작업이 포함되어 있다. 일반적으로 문장 분할은 마침표(.), 따옴표(!), 따옴표(?) 등의 문장 부호를 통해 수행될 수 있다. 하지만, (예제 1)과 같이 문장 부호가 축약점으로 쓰이거나 축약점 자체가 문장 부호로도 쓰여서 (Full Stop) 문장 부호와 구분이 애매한 경우가 발생하기도 한다.

(예제 1) 그는 Dr. Lee를 알고 있다.

John Mackenzie Jr. lives in Dallas, Tex. This is a fact.

이러한 애매성을 해결하기 위해, 정규문법을 이용하는 방법([4],[24])과 결정 트리를 이용하는 방법([8]), 신경망에 의한 학습 데이터를 이용하는 방법([8]), Maximum Entropy를 이용하는 방법([5],[11]). 그리고, 문서 내 어휘 중심의 접근 방법([2]) 등이 사용되고 있다. 특히, [2]는 미등록 축약어에 대해 대소문자의 위치 정보나 축약 여부 판단하고, 문장 분할 정보로 사용하여 99.13%의 정확도를 나타내었다. 한국어의 경우, [24]는 신경망에 의한 학습 데이터를 이용한 방법을 시도하였는데, 영어에 비해서 한국어의 축약어 사용이 적기 때문에 일반 전자문서에 대해 휴리스틱 정보 만을 사용해도 99.07%의 성능을 나타냄을 보였다.

(예제 2) 자주움직이게되므로몸에배달이나기쉽습니다.

"대 학 생 선 교 회"에 다녀오겠습니다.

한편 (예제 2)와 같이, 문장 분할 후에도 입력 문장 내에 띄어쓰기 오류가 존재할 경우, 언어 처리 응용 시스템들의 성능을 저하시킨다. 특히, 정보 검색 시스템에서 사용자의 질문 문장의 띄어쓰기가 잘못됐거나 인덱싱 문서의 띄어쓰기가 잘못 되었을 경우, 분석 결과에 직접적인 영향을 미쳐 그릇된 결과를 만들게 된다. 띄어쓰기 오류는 띄붙오류와 붙띄오류가 구분되며, 두 가지 오류를 모두 해결하기 어렵고, 처리 과정에 일관성이 떨어지기 때문에 모든 음절을 붙여서 해결하려는 경향이 많다. 한글 문장의 자동 띄어쓰기는 어휘 지식과 휴리스틱을 이용한 방법([17],[18], [19], [22])과 통계 정보를 이용한 방법([20],[21],[25])으로 크게 나눌 수 있다. 어휘 지식과 휴리스틱을 이용한 방법의 경우, 어휘 사전 규칙 정보에 의해 음절 단위 공기 정보를 이용하는 방법, 순방향 역방향 알고리즘, 양방향 최장일치법 등의 방법이 있다. 음절 단위 공기 정보와 휴리스틱을 이용할 경우, 공백 재현율이 97.3%, 어절 재현율이 93.2%에 이르렀다 한편, 후자의 경우, 어절 경계를 고려한 품사 태깅, 음절간 상호 정보를 이용한 어절 경계 인식의 방법, 음절규칙 적용 후 품사 태깅을 하는 방법이 있다

2.2. 문제 정의

기존의 연구들은 대체로 일반 전자 문서에서 각 문장이 문장 부호와 공백문자로 경계가 존재하는 경우와 한 문장 내에서의 띄어쓰기를 접근한 경우로 문서 내에서 각각의 문제가 상호 연관성을 가지고 발생하는 경우를 배제하고 있어 전처리 범위가 매우 지협적이다([22],[25]). (예제 3) 같이 띄어쓰기 오류에 의해서, 문장 부호와 축약점이 구분이 안가는 경우 새로운 문장 분할에 관한 새로운 애매성 문제를 발생하게 된다.

(예제 3) 우리 다.음.에서 만남시다.의 광고는 이번 달 최우수 카피라이터 상을 받았다.

이 경우, 항상 문장 부호 뒤에 공백 문자가 존재하여야 애매성을 해소하는 (예제1)의 접근 방법을 사용해서는 해결하기 힘들다. 한국어 문장에서는 대체로 축약이를 사용할 때, 음절과 음절 사이에 축약점을 사용하고 어절 끝에는 축약점을 사용하지 않은 경우가 많아서, 이러한 애매성을 해결하기가 쉽지 않다. 또, 때로는 문장과 문장간의 구분 과정에서 사용자의 실수나 의도적인 표현에 의해서 문장 부호가 생략되어 사용될 수 있다. 즉, (예제 4)와 같이 문장 부호를 가지지 않는 문장을 분할하는 경우, 문장 부호가 있는 문장을 분할하는 경우보다 더 많은 주변 정보를

반영하여 분석해야 하며 이는 문서 내에 존재하는 웹 문서의 서식과도 밀접한 관계가 있음을 알 수 있다. 특히, (예제 5)와 같이 문서 내에 모든 음절이 붙어 있는 경우 문장 부호가 없는 문장의 분할과 함께 띄불오류를 해결하는 띄어쓰기가 함께 고려 되어야 한다.

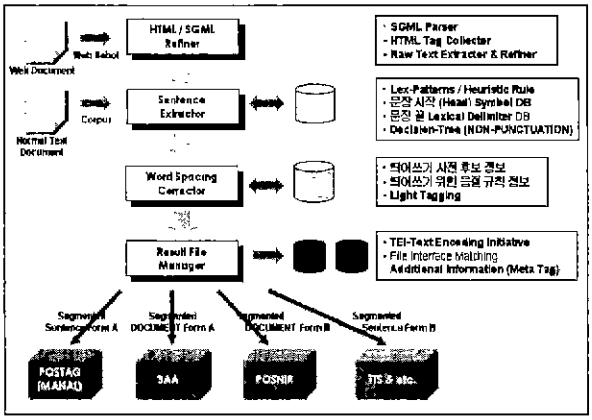
(예제 4) .... 라 말할 수 있다 본 논문의 구성은 .....
- 음절 단위의 띄어쓰기 정확도

(예제 5) ... 추정하여말할수있다이사실을바탕으로우리는또다른 가정을세워야한다 ...

이렇듯, 기존의 연구들은 문장 부호가 없는 문장에 대한 문장 분할을 해결하지 못하고 있다. 대부분의 접근 방법은 긴 학습 시간을 가지며, 학습 데이터의 성격에 따라 결과가 영향을 받는다. 한편, 웹 문서나 기타 문서에 존재하는 HTML 태그나 서식 기호 등의 정보를 분석 정보로 사용하지 않고 있다. 웹 인덱싱을 위한 전처리 시스템은 웹 문서를 손쉽게 자동으로 수집하여, 문서 내 철자오류를 수정하고, 축약어를 일정하게 복원하며, 문장 내에 존재하는 불필요한 문자열을 제거하고 정제하는 과정이 필요하다. 한편, 문서를 단락과 문장 단위로 분리하여 응용시스템의 입력으로 일괄되게 사용될 수 있도록 해야 한다. 또, 문장 내에 띄어쓰기 오류를 수정하고, 다양하게 존재하는 문서 내의 애매성 문제들을 해결함으로써 형태소 분석기 분석 성능을 보장해주는 역할을 할 수 있어야 한다 마지막으로 웹 문서로부터 추출된 원문을 구성하는 과정에서 습득 될 수 있는 문서의 속성 정보와 제목, 키워드, 의미 단어 집합(Multi Word Unit) 등의 의미 있는 문서 정보를 자동으로 재구성할 필요가 있다

3. WEB TAGGER : 웹 인덱싱을 위한 통합 전처리 시스템

포항공대 자연언어 처리 연구실에서 개발한 형태소 분석기인 POSTAG의 입력 Part에 해당하는 본 전처리기는 웹 로봇(nROBOT)의 다양한 옵션을 통해 사용자의 목적에 맞게 웹 문서를 수집한다 수집된 웹 문서는 문서 정제기 를 거치면서, HTML 태그가 제거되고, 문장 내에 불필요한 문자열이 제거된다. 형태소 분석기의 문장 분할기는 입력 문서를 문장 단위로 분리하는 과정에서 기존의 문장 분할 시스템과는 달리 문장 마침표가 있는 문장 뿐만 아니라, 문장 마침표가 없는 문장에 대해서도 정확하게 문장을 분할한다. 자동 띄어쓰기 시스템은 문장 내 띄어쓰기 오류를 수정하는 역할을 한다. (그림 1)은 WEB TAGGER의 전체적인 구성도이다.



(그림 1) WEB TAGGER 전체 구성도

3.1. nROBOT

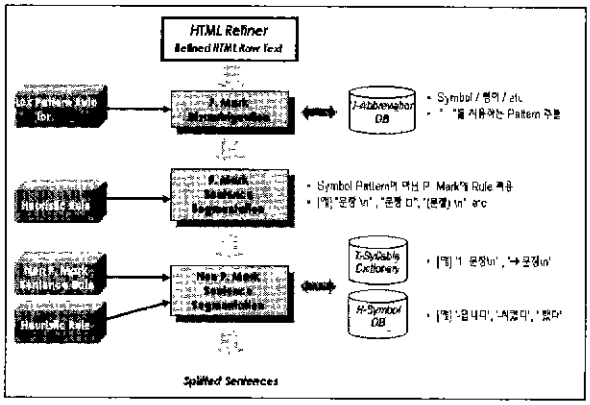
nROBOT은 getwww1.4, htdic, libwww, wget 등을 참조하여 개발한 웹 로봇이다. nROBOT은 20여 가지의 사용자 옵션을 통해 문서 수집의 강건함과 정확성 및 효율성을 유지하고 있다. 일반적인 하이퍼 링크에서 뿐만 아니라, JavaScript 코드와 Shock wave 파일에서도 URL을 추출한다 또, Virtual Host에 대해서도 웹 문서를 수집하며, Cookie를 지원하여 서버와 세션을 유지하면서 웹 문서를 수집하도록 제작하였다 특히, 웹 DB를 별도로 일관되게 관리하고, 웹 상에 존재하는 파일의 저장구조를 그대로 유지하면서, 특정 도메인 문서 모두를 자동으로 수집하는 장점을 가진다.

3.2. HTML Refiner

수집된 웹 문서는 HTML Refiner를 거치면서 원문을 복원하게 된다. 웹 문서는 SGML Parser에서 HTML 태그 오류를 수정하고 태그 별로 원문 내용을 정제한 후, 문서의 원문을 복원한다. 이때, HTML 태그의 속성에 따라서 제목(Title), 키워드, 문서설명(Description), 정보를 함께 추출한다 또, <li>, <td> 외 다수의 HTML 태그 정보만을 이용하여 1차적인 문장을 구분한다. 특히 POSTAG의 안정성을 보장하기 위하여, 문장의 길이를 1024 byte로 제한하고, 어절의 길이를 256byte로 제한하며, 의미 없는 2바이트 코드 문자를 비롯한 불필요한 문자열들을 제거한다

3.3. Sentence Extractor

문장 마침표가 있는 문장을 분리하는 과정에서는 문장 마침표와 더불어 나타나는 Lexical Pattern에 대한 정규 규칙을 사용하여 축약점(Abbreviation Period)과 문장 마침표의 모호성을 해소한다. (그림 3)과 같이 문장 분할기는 총 4개의 단계로 이루어져 있다.



(그림 3) Sentence Extractor

3.3.1. 정규 규칙을 이용한 문장 부호 있는 문장의 분할

정규 규칙은 모호성이 있는 문장 부호(.,?,!)의 사용에 대해서 Date-Time Pattern (9), Web Info Pattern (11), Head Symbol List (17), Abbreviation (4), Alpha numeric Reference (3), Numeric Expression (2) 등 총 46개의 정규 규칙과 예외 축약 DB를 사용한다. 그 결과, 기존 웹 전처리기에 비해서 축약점과 문장 부호의 애매성을 해소함과 동시에 문장 내에 존재하는 의미 있는 단어(Meaning Word Unit)를 더욱 다양하게 구별해내는 장점을 지니고 있다.

- **Abbr1 = [A-Za-z][^]*\.([,?;][a-z0-9]\.)[ ]**
- **Abbr2 = [A-Za-z]\.([A-Za-z]\.)+|[A-Z][bcdfghj-np-tvxz]+\.)[ ]**
- **Num1 = ([0-9]+,?)+(\.[0-9]+|[0-9]+)*[ ]**
- **Num2 = ([0-9]+[,])*[0-9]([.][0-9]+)?[ ]**

(예제 7) Regular Expression for P. mark Sentence

위의 정규문법에서 첫 규칙은 'L-3c509c." 등의 인덱스 형식의 축약을 추출하며, 둘째 규칙은 "I.C.C.P.O.L " 등의 축약어를 찾는다. 한편, 예외 축약 DB에는 총 532 개의 정규규칙에서 예외가 되는 빈도수가 높은 축약어들이 등록되어 있다 정규문법에서 처리하지 못하는 축약어에 대해 미등록 예외 축약어가 발견될 때마다 DB에 추가하거나 정규 규칙으로 반영하여 약 2%의 성능 향상에 기여하였다.

3.3.2. 휴리스틱을 이용한 문장 부호 있는 문장의 분할

한국어의 문장 분할에 대해서 대부분 문장 부호와 공백 문자열이 함께 나타날 경우 문장으로 인식하는 단순한 휴리스틱을 사용하고 있다. 이 경우, 축약이나 그 외의 마침표의 사용에 대해 많은 오류를 만들기 쉽다. 문장 부호와 축약점의 애매성 문제가 해결된 상태에서 한국어 문장 분할을 위한 휴리스틱을 적용할 경우, 휴리스틱의 정확도 더욱 향상된다. 이러한 휴리스틱을 적용함으로써 문장 부호가 존재하는 문장에 대한 문장 분할이 끝나게 된다 WEB Tagger에서는 기존의 휴리스틱보다 더 세밀하고 다양한 휴리스틱을 적용함으로써 문장 분할 정확도를 향상시키고자 한다. WEB Tagger에는 총 9가지의 휴리스틱 규칙이 있고, 대표적인 예는 (예제 8)과 같다 각각의 마침표(.)와 느낌표(!)와 물음표(?)는 문장 부호를 의미한다.

- **(.|!|?) + 공백문자열**
- **(.|!|?) + ("|') + 공백 문자열**
- **(.|!|?) + ('|"|{|[|(|<)**
- **({|[|(|<) + (.|!|?) + 공백 문자열**

(예제 8) Heuristic for P. mark Sentence

3.3.3. 휴리스틱을 이용한 문장 부호 없는 문장의 분할

웹 문서는 일반 전자 문서와는 달리 문장 부호 없는 문장의 빈도수가 매우 높다 이는 웹 문서가 기존의 전자 문서보다 다양한 문서를 포함하고 있고, 각 문서는 문서 작성자의 의도에 따라 자유롭게 기술되기 때문이다. 가령, 중앙일보 신문기사, 재몽 웹사전, 경영관련 강의 등의 사이트들은 일반 전자 문서와 유사한 문장 기호를 포함한 문장으로 구성되어 있는 반면, 컴퓨터 용어사전, 도량형단위 관련 사이트, PPT Slide에 의한 사이트, 행정 및 법률 사이트 등은 Symbol과 ASCII Code를 많이 포함하고 있었고, 홈쇼핑상품정보, 공공기관(포항공대) 홈페이지, 기업 홈페이지 등의 사이트 들은 테이블과 리스트 형식으로 문서 내의 문장들이 구성되어 있었다. 이러한 웹 문서들에서 임의로 200개의 문서에서 3500여 문장을 수작업으로 분류한 결과 대략

35.5%의 문장이 문장 부호를 가지지 않았으며, 특히 이중 가전제품 소개 문서와 PPT 파일에 대한 웹 문서 41개 950 문장에서는 71.3% 이상의 문장이 문장 부호 대신에 공백문자열을 사용하고 있었다 특히, 영어 문장에서는 Title. Subsection등을 제외하고는 문장부호를 정확히 사용하는데 비해서 한국어 문장에서는 문장 부호를 생략하는 경우가 빈번하였음을 알 수 있었다. 이렇듯, 한국어 웹 문서에서의 문장 분할은 문장 기호가 있는 문장 뿐만 아니라 문장 기호가 없는 문장에 대해서 매우 의미가 있다.

WEB TAGGER에서 문장 마침표가 없는 문장을 분리하는 과정에서는 문장 시작 부분에 발현하는 시작 기호와 문장 끝 부분에 발현하는 용언과 종결어미의 2-3음절 패턴을 사용하여서 문장 끝을 확인하는 휴리스틱을 먼저 적용 한다. 문장 시작 기호에 대한 휴리스틱은 1바이트(혹은 2바이트)로 이루어진 리스트 기호(* , • , ◆ , ⇒ , ㅁ 등 71개)와 순차적 번호(A. . 가. . 1. 등 106개)가 문장 시작과 함께 나타날 경우, 최초의 New Line 문자를 문장 부호로 기준 하여 한 문장으로 인식하도록 하였다. 또, 문장 끝 2-3음절 패턴(Space+한다, Space+된다, 했었다, 고한다, 되었다, 것이다, 주었다 등 79개)과 공백 문자열이 연속되어 나타날 경우, 이 공백 문자열을 기준으로 문장 분할을 하는 방법을 적용하였다

3.3.4. C4.5의 학습데이터를 이용한 문장 부호 없는 문장의 분할

하지만, 휴리스틱을 사용하더라도 대부분의 문장 부호 없는 문장의 분할에 적용하기 힘들다. 이에 품사 인덱스 정보와 어절 내 어휘 정보, 공백 문자 정보 등을 결정 트리(Decision Tree)를 구성하는 자질로 사용하여 C4.5의 학습 데이터를 이용한 문장 부호 없는 문장에 분할을 실험하였다. 본 실험에서는 결정 트리를 위크학습기(weak learner)를 사용하여 학습한 데이터를 절지(prunning)하는 C4.5의 대표적인 학습 알고리즘을 문장 분할에 적용하였다. 위크학습기를 사용할 경우, 빠른 시간동안 학습 데이터 자질들간의 다양한 규칙을 뽑아 대상 데이터의 속성을 분류할 수 있다 실험에 필요한 자질로는 문서 내에 공백 문자가 나타날 경우, 공백 문자를 기준으로 앞 어절과 뒷 어절의 음절 정보, 품사 정보 등을 사용하였다 문장 끝에는 대체로 명사, 어미의 이형태 등의 나타나고, 문장의 시작에는 대체로, 명사, 부사, 대명사 등이 나타므로, 품사 정보와 조사, 어미의 음절 정보는 공백문자가 문장 분할 후보가 될 수 있는가를 판별하는 좋은 자질이 된다.

실험 순서로는 먼저 대상문서를 공백 문자들을 기준으로 각 공백 문자의 좌우 정보를 나타내는 문서로 변환 한 후, 변환된 문서에서 각각의 공백 문자로 문장 분할 기준이 되는지 여부를 파악하고, 이를 원 문서에서 반영하는 방식을 취하였다.

- **F어절 last1s : 다(a), 니(b), 오(c), 까(d), 어(e), 음(f), 만(g), 며(h), 는(i), ...**
- **F어절 last2s : 니(a), 었(b), 했(c), 갔(d), 였(e), 시(f), 되(g), 된(h), ... ?.**
- **F어절 last3s : 습(a), 되(b), 합(c), 랍(d), 어(e), .. ?.**
- **B어절 last1s : 다(a), 니(b), 오(c), 까(d), 어(e), 음(f), 만(g), 여(h), 는(i), ...**
- **B어절 first1s : 그(a), 한(b), 오(c), ... , ?.**
- **F어절 Full tag : ?, B, M, T.**
- ***B어절 Full tag* : ?, B, M, T.**
- **F어절 2음절 Part tag : ?, T, B, M.**
- **F어절 3음절 Part tag : ?, T, B, M.**
- **F어절 Pattern : y, n.**

(예제 9) 문장 분할을 학습을 Attributes Name과 Value

3.4. Automated Word Spacing Corrector

자동 띄어쓰기 시스템에서는 문장 내에 존재하는 6음절 이상의 어절에 대해서 띄어쓰기 오류를 수정한다. 먼저 띄어쓰기가 가능한 이형태를 최장일치로 확인하고 바로 띄어쓰기를 적용하는 모듈(Entry Checker)과 띄어쓰기 규칙을 가지는 음절간 형태 정보를 찾아서 띄어쓰기를 바로 적용하는 모듈(Pattern Checker), 그리고, 간단한 품사 정보를 이용하여 띄어쓰기를 적용하는 모듈(Light Tagger) 등 3가지 구조로 이루어져 있다

3.4.1. Entry Checking

조사, 어미, 관형사, 수사, 부사, 대명사 등 6개의 품사 내에서 이형태의 중의성 없어 바로 띄어쓰기가 가능한 5500여 Entry에 의한 띄어쓰기를 시도한다. 각 Entry들은 -1, 1, 2 중의 하나의 값을 가지게 되며 각각은 Entry의 앞, 뒤, 혹은 앞뒤에 공백문자를 삽입하도록 하고 있다

3.4.2. Pattern Checking

보조용언, Collocation 사전, 일반적인 붙띄오류 등 880개의 패턴에 대해서 정답의 패턴으로 치환을 시도한다. 사전 구성도 아래의 (예제 11)과 같이 해당 문자열에 대한 정답 문자열로 이루어져 있으며, 문장 시작의 # 은 주석문을 @는 동록 패턴을 의미한다.

(예제 11) Pattern Checking Dic

- @ /그사람/ /그 사람/
- @ /기쉽/ /기 쉽/
- @ /려하/ /려 하/
- @ /만큼더/ /만큼 더/

3.4.3. Light Tagging

문서의 띄어쓰기 오류를 수정하는 작업은 형태소 분석 결과의 정확도를 보장하기 위한 과정이지만, 한국어의 음절간 규칙만을 사용하여 분석할 경우, 정확도가 떨어지기 때문에 대부분의 시스템에서는 형태소 분석기의 결과를 바탕으로 다시 띄어쓰기를 하는 방법을 취하고 있다. 이는 닭과 달걀의 문제로 실제 응용 시스템에서 형태소 분석을 2번 해야 하는 부담을 안고 있다 이러한 문제에 대해, 품사에 대한 존재 여부를 POSTAG의 트라이 사전에서 파악함으로써 해결하고자 한다.

띄붙오류를 가지는 문자열 내에서 최장일치로 매치된 어미, 조사, 조용사의 이형태를 기준으로 앞뒤 4음절 이내의 음절열에 대해서 명사, 용언 사전을 찾거나 ASCII 문자열인을 확인함으로써, (예제 11)과 같은 품사열 패턴인지 여부를 파악할 수 있다. 이러한 품사열 패턴이 확인된 후에는 해당된 어미, 조사, 조용사의 이형태의 바로 뒤에서 띄어쓰기가 가능하게 된다.

(예제 11) Light Tagging Pattern

- 명사 + 조사 + 명사
- 명사 + 접미사 + 조사 + 명사
- 명사 + 조용사 이형태 + 명사 혹은 용언 lexical
- 명사 + 기호(숫자,영어)+ 조사 + 명사 혹은 용언Lexical
- 명사 + 기호(숫자,명어)+ 조용사 + 명사 혹은 용언Lexical
- 명사 + 연결사-어미 lexical + 명사 혹은 용언Lexical

이러한 접근 방법은 시스템에 적은 부하를 걸면서 띄어쓰기 결정에 중요한 정보인 품사 정보를 효과적으로 사용하여 1음절 조사나 어미에서 발생하는 중의성을 해소한다

3.5. Result File Manager

웹 문서가 정제 과정, 문장 분할 및 문장 단위 정렬 과정, 문장 내 띄어쓰기 오류 교정 과정을 거치고 나면, 응용 프로그램에 맞는 초기 문서를 구성해 줄 필요가 있다 이 과정에서는 각 응용프로그램 별로 부가적인 전처리 과정이 들어가게 된다. TTS 시스템을 위해서 영문이나 한자를 한글음으로 바꾸거나, 정보 추출기를 위해 웹 문서 원문을 형태소 분석한 결과를 다시 웹 문서와 결합하는 등의 과정이 이에 포함된다 웹 문서를 인덱싱 하기 위한 초기 문서에서 웹문서의 URL. 제목, 키워드, 본문 내용, 문서 제작 날짜 등을 XML형식으로 구성하였다(그림 4).

```
(DOC
(TITLE
.... Html의 <title> </title> 태그내용
)TITLE
(KEYWORD
.....Html의 Keyword <meta> 태그내용
)KEYWORD
(BODYTEXT
.... Html의 <body> </body> 태그내용
)BODYTEXT
(URL
http://..... Html의 URL
)URL
)DOC
```

4. 실험 및 평가 결과

본 논문에서 제안한 알고리즘을 구현하여 각 시스템별로 성능을 검증해 보았다 (표 1)은 중앙일보 사이트의 게몽 웹 백과 사전 사이트의 3799 문장에 존재하는 마침표(.) 3967개에 대해 수작업으로 분류해 본 결과이다. 즉, 만일 모든 마침표(.)를 문장 기호로 할 경우, 91.85%의 기본 정확도를 얻을 수 있다. 이 기준으로 정규문법에 의한 문장 분할과 축약 DB에 의한 문장 분할 그리고, 한국어 문장 분할 Heuristic에 의한 문장 분할 결과를 살피보면, (표 2)에서와 같이 축약 규칙과 축약 DB와 문장 Heuristic을 모두 적용하였을 때, 98 84%의 정확도를 얻을 수 있었다. 한편, 일반 전자문서에 대해서 99% 이상을 나타내었던 한국이 문장 휴리스

턱의 경우, 웹 문서에 대해서 적용한 결과 93.85%로 기본 정확도에 근접한 결과를 나타내었음을 알 수 있었다.

	문장 수	마침표 개수
Punctuation Mark	3512	3668
Non-Terminal Punctuation Mark	281	299
Total	3799 문장	3967 개

(표 3)실험 1 : 문장 내 마침표의 분류

실험	Combination	정확도
Base	All P. Mark	91.85%
RegExp 1	축약규칙 적용 후 Base	95.51% (178)
RegExp 2	축약규칙, 축약DB 적용 후 Base	96.57% (136)
Heuristic 1	문장 Heuristic 적용 후	93.85% (244)
Heuristic 2	축약규칙, 축약DB 적용 후 문장Heuristic	98.84% (46)

(표 2) 실험 2 : Heuristic Rule 적용

(괄호 안의 숫자는 3799개 중에서 오류 개수)

(표 3)은 KTSET95에서 뽑은 문장 기호가 없는 1700문장에 대한 문장 분할 실험이다. KTSET95에서 실험의 Base Line으로 문서 내에서 "다" 음절 뒤에 공백 문자가 나올 경우를 모두 문장의 끝이라고 고려하고 실험하였다. 이 경우, 1700개의 해당 공백 문자 위치 중에서 1087개에 해당하는 63.94%의 정확도를 나타냈고, 이와는 별도로, 147개의 과부하 오류를 발생시켰다. (이는 "갖다 놓은" 이나 "데이다 정확도" 등의 어휘에서 주로 발생되었다). 이를 기준으로 문장 끝 2-3음절 패턴과 공백 문자열이 함께 나타날 경우에 대한 실험과 C4.5에 의해서 1700개 문장을 학습시킨 데이터를 기준으로 학습 데이터 내에서 300문장을 뽑아서 테스트한 실험에 대한 결과와 학습 데이터와 다른 문서에서 뽑은 임의의 300문장을 뽑아서 테스트한 실험에 대한 결과를 살펴보면 (표 3)과 같다. 즉, C4.5에 의해 학습된 결과에서는 94.23%의 정확도를 나타내었지만, 문장끝 음절 패턴보도 67개나 더 많은 과오류를 발생시켰으며, 이를 학습데이터 외부에서 데이터를 추출할 경우 과오류수가 25.3%나 증가하였으며, 반면에 정확도는 1.47% 감소하였다

		정확도	과오류수
Base	"다" + 공백문자	63.94%	147
Heuristic	문장끝 음절패턴	79.35%	22
C4.5 (1)	1700외부 300개	92.76%	114
C4.5 (2)	1700내부 300개	94.23%	91

(표 3) 실험 3 : 문장 기호가 없는 문장 분할 실험

(C4.5의 경우 300개 결과를 1700개로 확장)

5. 결론 및 향후 일정

본 논문에서는 인덱싱 대상 문서로 사용되는 웹 문서를 자동으로 수집하여, 정제된 문장을 갖춘 문서로 제작, 관리하는 통합 전처리 시스템인 Web Tagger의 구조와 전처리 방법을 소개하였다. 이제는 언어 처리의 기본이 되는 문서를 다양하고 체계적인 접근 방법에 의해 통합적이면서 전문적으로 구축해야 할 때이다. 이 과정에서 해결해야 할 다양한 문제는 기존과 같이 한가지를 대상으로 직접적으로 해결하는 것이 아니라 다양한 단계를 거치면서 유기적으로 해결해야 할 필요가 있다. 본 논문에서 제안한 방법으로 웹 문서 내에 존재하는 문장 기호가 있는 문장의 분할 정확도가 98.84%이고, 문장 기호가 없는 문장의 분할 정확도가 94.23%였다.

키워드

Text Normalization, Document Tokenization, Sentence Boundary Segmentation, Automated Spacing, Preprocessing, Sentence Split

참고 문헌

[1] A Mikheev, C. Grover and C Matheson. 1998(b) TTT Text Tokenisation Tool. Language Technology Group, University of Edinburgh http://www.ltg.ed.ac.uk/software/ttt/index.html

[2] A. Mikheev. 2000 Document Centered Approach to Text Normalization. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2000), pp. 136-143. Athens. Greece

[3] A Mikheev. 1998. Feature Lattices for Maximum Entropy Modeling. In Proceedings of the 36th Conference of the Association for Computational Linguistics (ACL/COLING'98), pp. 848-854. Montreal, Quebec.

[4] A Mikheev. A knowledge-free method for capitalized word disambiguation In Proceedings of the 37th Conference of the

Association for Computational Linguistics (ACL'99), pages 159-168. University of Maryland, 1999

[5] A. Ratnaparkhi. 1996 A maximum entropy model for part-of-speech tagging. In Proceedings of Conference on Empirical Methods in Natural Language Processing, pp 133-142 University of Pennsylvania

[6] Berkeley University. 1999. Dictionary of Abbreviations and Acronyms in Geographic Information Systems, Cartography, and Remote Sensing
http://www.lib.berkeley.edu/EART/abbrev.html

[7] Bong-Rae Park, Hae-Chang Rim. "A Korean Corpus Refining System based on Automatic Analysis of Corpus", Proceedings of Natural Language Processing Pacific Rim Symposium'95, pp 89-94, 1995

[8] Collins. Michael. 1996. A new statistical parser based on bigram lexical dependenices. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, June.

[9] D. D. Parmer and M. A. Hearst. Adaptive multilingual sentence boundary disambiguation Computational Linguistics, 1997.

[10] G. Grefenstette. 1994. What is a word, What is a sentence, Problems of Tokenization In the Proceedings of the Conference on Computational Lexicography and Text Research COMPLEX'94, Budapest.

[11] K Church. 1995 One Term Or Two? In Proceedings of the 18th Annual Internationals ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95), Seattle

[12] J. C. Reynar and A. Ratnaparkhi. 1997. A Maximum Entropy Approach to Identifying Sentence Boundaries. In Proceedings of the Fifth ACL Conference on Applied Natural Language Processing (ANLP'97), Washington D.C., ACL

[13] Liberman, Mark Y. and Kenneth W. Church. 1992. Text analysis and word pronunciation in text-to-speech synthesis. In Sadaoki Furui and M Mohan Sondi, editors, Advances in Speech Signal Processing. Marcel Dekker, Incorporated, New York.

[14] M.E Lesk and E. Schmidt. 1975. LEX – a lexical analyzer generator Computing Science Technical Report 39 AT&T Bell Laboratories. Murray Hill, N.J

[15] Palmer, David D. and Marti A Hearst. To appear. Adaptive multilingual sentence boundary disambiguation. Computational Linguistics.

[16] Ratnaparkhi, Adwait 1996. A maximum entropy model for part-of-speech tagging In Conference on Empirical Methods in Natural Language Processing, page 133-142, University of Pennsylvania, May 17-18.

[17] 강승식, "음절 정보와 복수어 단위 보를 이용한 한국어 형태소 분석", 서울대학교 컴퓨터공학과 박사학위 논문, 1993

[18] 강승식, "한글 문장의 자동 띄어쓰기", 제 10회 한글 및 한국어 정보처리 학술대회 발표논문집, pp. 137-142, 1998

[19] 김계성, 이현주, 이상조, "음절 정보를 이용한 한국어 띄어쓰기의 구현". 정보과학회 추계 학술발표 논문집, 제 24권 2호, 1997

[20] 김진동, 이상주, 임해창, "어절 띄어쓰기를 고려한 형태소 단위 품사 태깅 모델". 제 10회 한글 및 한국어 정보처리 학술대회 발표논문집, pp. 3-8, 1998

[21] 심광섭, "음절간 상호 정보를 이용한 한국어 자동 띄어쓰기", 정보과학회 논문지(B), 23권 9호, pp. 991-1000, 1996

[22] 안동언, "웹용 영한 기계번역을 위한 문서 전처리기의 설계 및 구현", 한글 및 한국어 정보 처리 학술 발표 논문지, pp. 249-254, 1997

[23] 원영섭, 띄어쓰기,맞춤법 용례, 세창, 1993

[24] 정경미, "규칙을 기반으로 한 한국어 텍스트에서의 문장 분할", 포항공과 대학교 정보통신대학원 석사학위 논문, 1996

[25] 최재혁, "양방향 최장일치법을 이용한 한국어 띄어쓰기 자동교정 시스템", 한글 및 한국어 정보 처리 학술 발표 논문지, pp. 145-151, 1997

1) 본 연구는 과학재단 특정기초(1997 9-2000.8 #97-0102-03-01-3) 와 BK21(교육부 두뇌 한국사업)의 연구비 지원으로 수행되었습니다