TnT 태거를 이용한 Head-Tail 품사 태거

서현재, 김정민, 강승식
국민대학교, 컴퓨터공학과
hjworld32@gmail.com, kimjm@kookmin.ac.kr, sskang@kookmin.ac.kr

Korean Head-Tail POS Tagger using TnT Tagger

Hyun-Jae Suh, Jung-Min Kim, Seung-Shik Kang
Kookmin University, Dept. of Computer Science

요 약

한글을 토큰화할 때, 어절을 토큰화하는 기준이 명확하지 않다는 문제점이 있다. 기존 한글 품사 태거는 형태소 단위로 어절을 분해하여 불필요한 형태소까지 추출하기 때문에 문장 내 단어 간의 연관성을 파악하는데 어렵게 만들고 모델의 성능을 떨어뜨릴 수 있다. 따라서 Head-Tail 방식으로 어절을 음절 단위로 Head, Tail 두 개의 토큰으로만 분할하여 어절이 과도하게 분해되는 문제점을 보완하였다. 통계 기반 품사 태거인 TnT tagger을 사용하여 Head-Tail 토큰화된 데이터셋에 대해 품사 태깅을 수행하였다.

1. 서론

한글 토크나이저에는 토큰을 어떻게 정의할 것인지에 대한 문제가 있다. 한글은 어절을 토큰으로 분해할 때의 기준이 다소 모호하다는 특징이 있다. 예를 들어 “맨주먹”이라는 단어를 토큰화할 때, 접두사인 “맨”을 별개의 토큰으로 분해할지, “맨주먹”이라는 하나의 토큰으로 분해할지가 명확하게 정해져 있지 않다.

기존의 한글 품사 태거는 어휘형태소와 문법형태소를 세세하게 토큰화한 후 품사 태깅을 하는 방식으로 작동한다. 이 같은 방식은 의미적으로 불필요한 형태소까지 분해하여 모델의 성능을 떨어뜨릴 수 있다는 단점이 될 수도 있다. 기존의 형태소 단위 토큰화를 음절 단위 토큰화 방법으로 대체함으로써 결과를 단순화할 수 있다. 음절 단위 토큰화 방법인 Head-Tail 토큰화를 통해 토큰화 결과가 과도하게 복잡해지는 문제를 해결하고자 한다.

Head-Tail 토큰화는 어절을 Head, Tail 두 개의 토큰으로 분리하는 토큰화 방법이다. 어절의 어휘형태소 부분을 Head로, 문법형태소 부분을 Tail로 지정한다. 여기서 Tail은 Head를 제외한 나머지 부분이 된다. Head-Tail을 분리할 때는 음절 단위로 토큰화한다. 2개의 음절이 하나의 음절로 표현된 경우에도 음절을 분해하지 않는다. 또한 복합어는 분해하지 않는다.

본 논문에서는 Head-Tail 방식으로 토큰화한 데이터에 대해 품사 태깅을 수행한다. 통계 기반 품사 태거인 TnT tagger를 이용하여 Head-Tail 토큰화한 데이터에 대한 품사 태거 모델을 소개하고자 한다.

2. 관련 연구

2.1 TnT tagger

TnT(Trigrams N Tags) tagger은 2차 마르코프 모델에 기초한 통계 기반 품사 태거이다. 언어와 태그셋 종류에 관계없이 학습이 가능하다. TnT tagger은 영어, 독일어 말뭉치 데이터셋에 대해서 우수한 품사 태깅 정확도를 보이는 것으로 보고되었다[1].

마르코프 모델의 성질에 따라 특정 토큰의 태그는 이전 태그에 의존한다. TnT tagger에서는 이전 두 개의 태그에 의존하여 알맞은 태그를 결정한다. 시퀀스 끝 마커 t에 의존성을 추가하여 문장의 끝을 인식할 수 있도록 한다. 이를 방정식으로 나타내면 수식 1과 같다.

$$arg\max_{t_1^n}\left[\prod_{i=1}^{n} P(W_i|t_i)P(t_i|t_{i-1},t_{i-2})\right]P(t_{n+1}|t_n) \quad (1)$$

TnT tagger은 품사 태깅된 말뭉치로부터 전환, 출력 확률을 예측한다. 트라이그램만을 고려한다면 데이터 희소성의 문제가 발생할 수 있으므로 트라이그램과 함께 유니그램, 바이그램 확률도 활용한다. 첫 번째 단계로 수식 2, 3, 4, 5과 같이 품사 태그 셋의 모든 t_1, t_2, t_3와 단어 사전의 w_3에 대한 최대우도 확률 $\hat{P}$을 구한다. N은 학습 말뭉치의 토큰 수이다.

$$Unigrams : \hat{P}(t_3) = \frac{f(t_3)}{N} \quad (2)$$

$$Bigrams : \hat{P}(t_3|t_2) = \frac{f(t_2,t_3)}{f(t_2)} \quad (3)$$

$$Trigrams : \hat{P}(t_3|t_1,t_2) = \frac{f(t_1,t_2,t_3)}{f(t_1,t_2)} \quad (4)$$

$$Lexical : \hat{P}(w_3|t_3) = \frac{f(w_3,t_3)}{f(t_3)} \quad (5)$$

다음에는 선형 보간법을 이용해 유니그램, 바이그램, 트라이그램 확률의 가중합을 구한다. 가중치 값을 이용해

수식 6과 같이 확률을 구할 수 있다. $\hat{P}$은 최대우도 확률이고, 결과 값이 확률 값임을 보장해야 하므로 $\lambda_1 + \lambda_2 + \lambda_3 = 1$이 된다.

$$P(t_3|t_1,t_2) = \lambda_1\hat{P}(t_3) + \lambda_2\hat{P}(t_3|t_2) + \lambda_3\hat{P}(t_3|t_1,t_2) \quad (6)$$

가중치 λ_1, λ_2, λ_3값은 다음과 같이 구할 수 있다. 우선 가중치 λ_1, λ_2, λ_3값을 모두 0으로 둔다. 모든 트라이그램을 순회하며 유니그램, 바이그램, 트라이그램 중에 가장 큰 값을 가지는 경우가 무엇인지 파악한다. 유니그램이 가장 큰 값을 가지는 경우, λ_1의 값을, 바이그램이 가장 큰 값을 가지는 경우는 λ_2의 값을, 트라이그램이 가장 큰 값을 가지는 경우는 λ_3의 값을 f(t_1, t_2, t_3) 만큼 증가시킨다. 모든 트라이그램을 순회하였으면 λ_1, λ_2, λ_3 값을 정규화한다.

3. 모델

3.1 데이터셋 설명

모델 학습, 테스트에는 KCC 국민대학교 한국어 원시 말뭉치[1]를 부산외대와 전북대에서 형태소 분석한 데이터셋[2]을사용하였다. TnT 태거 모델의 학습데이터로는 KCCq28 데이터셋을 사용하였다. KCCq28 데이터셋은 1050866개의 문장으로 구성되어 있다. 테스트 데이터로는 KCC150 데이터셋에서 30000문장을 추출하여 사용하였다.

학습과 테스트에 사용한 데이터셋은 Head-Tail 형태로 구성되어 있다. 문장을 어절 단위로 분리한 다음, 각 어절을 Head-Tail 방식으로 분할한다. 어절의 처음부터 어휘형태소의 길이만큼을 Head로, 그 이후 부분을 Tail로 간주한다. 이후 각 토큰의 품사 정보를 부착한다. Head, Tail로 분할 했을 때, 여러 형태소가 하나의 토큰에 포함되었을 경우, '_'로 합쳐서 표시해주었다.

(표 1) KCCq28 Head-Tail 데이터 예시

[('잠실', 'NNP'), ('에서', 'JKB'), ('우리', 'NP'), ('가', 'JKS'), ('더', 'MAG'), ('강했', 'VA'), ('다', 'EP_EF'), ('.', 'SF')]
[('앞', 'NNG'), ('으로도', 'JKB_JX'), ('이런', 'MM'), ('얘기', 'NNG'), ('가', 'JKS'), ('계속', 'MAG'), ('나올', 'VV_ETM'), ('것', 'NNB')]

3.2 기존 형태소 분석과 Head-Tail 토큰화된 데이터 비교

기존 형태소 분석은 어절을 언어의 최소 의미 단위인 형태소로 분절한다. 따라서 어절에 존재하는 모든 형태소가 분절되어 나타나게 된다. 반면 Head-Tail 토큰화는 모든 형태소를 분절하지 않고, 어휘형태소 부분을 Head로, 문법형태소 부분을 Tail로 분리하는 단순한 방식을 취한다.

(표 2) 기존 형태소 분석과 Head-Tail 비교

어절	기존 형태소 분석(Kkma)	**Head-Tail**
이것이	이것/NP+이/JKS	**이것/NP+이/JKS**
전반적인	전반적/NNG+이/VCP+ㄴ/ETD	**전반적/NNG+인/VCP+ETM**
추세인지는	추세/NNG+이/VCP+ㄴ지/ECD+는/JX	**추세/NNG+인지는/VCP_EC_JX**
따져봐야	따지/VV+어/ECS+보/VXV+아야/ECD	**따져/VV+봐야/EC_VX_EC**

기존 형태소 분석 방식은 문장을 형태소 단위로 분절하였다. "전반적인"을 "전반적"+"이"+"ㄴ"와 같이 형태소 단위로 세세하게 분절하였다. 반면 Head-Tail은 "전반적"+"인"으로 분절하였다. 어휘형태소인 "전반적"을 Head로, 나머지 부분인 "인"을 Tail로 지정하여 Head, Tail 단 두 개의 요소로 토큰화하였다.

3.3 모델의 입력, 출력 데이터 구조

TnT tagger 모델에는 Head-Tail 방식으로 토큰화된 문장이 입력된다. 토큰화된 문장을 입력받아 각 토큰에 대한 알맞은 품사를 태깅한다. 각 토큰에 대해 (Token, POS tag) 형태로 출력한다.

(표 3) 입력 데이터와 출력 데이터 형식

Input	['세계', '금융시장', '이', '극심한', '공포', '에서', '잠시', '벗어났', '다', '.']
Output	[('세계', 'NNG'), ('금융시장', 'NNG'), ('이', 'JKS'), ('극심한', 'VA_ETM'), ('공포', 'NNG'), ('에서', 'JKB'), ('잠시', 'MAG'), ('벗어났', 'VV'), ('다', 'EF'), ('.', 'SF')]

표 4의 예시에서 입력, 출력 데이터의 형식을 확인할 수 있다. 입력 데이터는 Head-Tail 방식으로 토큰화된 단어들의 리스트이다. 출력 데이터는 각 토큰에 대해 알맞은 품사를 태깅하여 튜플로 묶어 저장한 리스트이다.

1) http://nlp.kookmin.ac.kr/kcc/

2) https://github.com/bufsnlp2030/BUFS-JBNUCorpus2020

3.4 모델 태깅 결과

TnT tagger 모델로 품사 태깅한 결과는 다음과 같다.

(표 4) TnT tagger 모델로 알맞게 품사 태깅된 결과

입력	['응답자', '중', '가장', '많', '은', '의견', '이었다', '.']
결과	[('응답자', 'NNG'), ('중', 'NNB'), ('가장', 'MAG'), ('많', 'VA'), ('은', 'ETM'), ('의견', 'NNG'), ('이었다', 'VCP_EP_EF'), ('.', 'SF')]
입력	['반면', '글로벌', '금융위기', '의', '주범', '이었던', '선진국', '의', '살림살이', '는', '크', '게', '악화됐', '다', '.']
결과	[('반면', 'NNG'), ('글로벌', 'NNG'), ('금융위기', 'NNG'), ('의', 'JKG'), ('주범', 'NNG'), ('이었던', 'VCP_EP_ETM'), ('선진국', 'NNG'), ('의', 'JKG'), ('살림살이', 'NNG'), ('는', 'JX'), ('크', 'VA'), ('게', 'EC'), ('악화됐', 'VV'), ('다', 'EF'), ('.', 'SF')]

(표 5) TnT tagger 모델로 잘못 품사 태깅된 결과

입력	['무균성', '인', '경우세균성', '보다', '증상', '이', '가볍', '고', '후유증', '을', '남기', '는', '경우', '도', '적', '다', '.']
결과	[('무균성', 'Unk'), ('인', 'VCP_ETM'), ('경우세균성', 'Unk'), ('보다', 'JKB'), ('증상', 'NNG'), ('이', 'JKS'), ('가볍', 'VA'), ('고', 'EC'), ('후유증', 'NNG'), ('을', 'JKO'), ('남기', 'VV'), ('는', 'ETM'), ('경우', 'NNG'), ('도', 'JX'), ('적', 'VA'), ('다', 'EF'), ('.', 'SF')]
입력	['꽃', '물결', '에', '발걸음', '이', '한층', '가벼워집', '니다', '.']
결과	[('꽃', 'NNG'), ('물결', 'NNG'), ('에', 'JKB'), ('발걸음', 'NNG'), ('이', 'JKS'), ('한층', 'MAG'), ('가벼워집', 'Unk'), ('니다', 'EF'), ('.', 'SF')]

4. 실험 결과

4.1 테스트 데이터에 대한 정확도

(표 6)

테스트 데이터	정확도
KCC150 30000개 문장	0.9387

Head-Tail 토큰화된 KCCq28 데이터셋으로 TnT tagger 모델을 학습시키고 KCC150 데이터셋에서 30000개 문장을 추출하여 실험한 결과, 93.87%의 정확도를 보였다.

5. 결론

형태소 위주의 한글 토큰화 방식의 단점을 보완한 Head-Tail 토큰화 방식으로 KCCq28 데이터셋을 토큰화한 후, 해당 데이터로 TnT tagger를 학습시켰다. 이후 학습된 TnT tagger로 KCC150 데이터셋에서 추출한 30000개 문장에 대해 품사 태깅을 수행하고 정확도를 측정하였다. 결론적으로 TnT tagger가 Head-Tail 토큰화 데이터셋에 대해 충분히 우수한 품사 태깅 성능을 보인다는 실험 결과를 도출하였다.

Acknowledgement

이 논문은 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2021R1F1A1061433).

참 고 문 헌

[1] Thorsten Brants, TnT - A Statistical Part-of-Speech Tagger, ANLP-2000, April 29-May 3, 2000