CRF를 활용한 한글로 적힌 외래어 및 외국어 받아쓰기 인식

제이미

"하나무라로 떠납니다"라는 문장을 봤을때 우린 "하나무라"가 일본 지명을 한글로 받아 적은 단어라는 것을 바로 알 수 있다. 비슷하게 "토도케테 세츠나사니와 나마에오 츠케요 오카 스노우 할레이션"을 봤을때도 "스노우 할레이션"은 영어 단어를, 나머지는 일본어 문장을 그대로 받아 적은 한글이란 것을 바로 알 수 있다. 이 글에선 이러한 작업을 딥러닝을 활용하여 자동으로 인식 할 수 있게 해주는 방법을 제안한다. 주 목적은 한국어로 된 문장 속 외래어와 한국어 문장이 아닌 외국어 받아쓰기를 알아내는 것이다. 이는 추후에 개체명 인식이나 말뭉치 정제, 한본어와 같은 추가적인 외래어 분석 등에 활용 될 수 있을 것으로 보인다.

I. 정의

A. 외래어

외래어란 보통 외국에서 빌려서 한국어처럼 사용하는 단어 혹은 외국어의 고유명사를 뜻한다[1]. 빌려왔다는 점에서 '차용어'라고도 한다. 전자의 예를 들자면 일반명사 '텔레비전'의 경우, 그것을 지칭할 한국어가 기존에 없었기 때문에 영어 'Television'에서 빌려와서 외래어 표기법으로 적은 후 한국어 단어처럼 사용한다. 후자의 경우 '러브라이브'가 대표적으로, 고유명사는 특정한 사람이나 기관, 작품 등을 구체적으로 지칭하기 위해 사용되기 때문에[2] 마찬가지로 외래어 표기법으로 적어서 사용한다.

B. 외국어 받아쓰기

외국어 받아쓰기dictation의 경우 외래어와는 다르게 목적 자체가 의사소통이나 정보 전달에 있지 않고, 주로 외국어를 읽는데 도움을 줄 수 있도록 들리는대로 한글로 적는 것을 뜻한다. 이를테면 "아루코오 하데나이미치"라는 구절은 노래를 쉽게 부를 수 있도록 일본어 가사를 한글로 적은 것이다.

II. 알고리즘

A. 품사 태깅

자연어처리(이하 NLP) 분야에서 가장 기본적으로 시행되는 프로세스는 품사 태깅[3]이다. 각 단어별로 품사가 무엇인지 구분해서 `태그'를 달아두는 방식인데 이렇게 해두면 추후에 통계적인 방법을 통해 컴퓨터가 자동으로 여러가지 일을 처리 할 수 있게끔 해주는 일종의 방법론이다[4]. 가령 'lazenca save us'라는 문장이 있다면 'lazenca/명사', 'save/동사', 'us/대명사' 처럼 각 단어의 품사를 기록해주는 것이다.

품사 태그를 은닉 마르코프 모델(이하 HMM)과 같은 방법으로 학습 하면 태그가 달리지 않은 단어에 대해서도 컴퓨터로 하여금 추론하게 할 수 있고[5], 여기서 명사만 뽑아서 주요 키워드로 선정한다거나 하는 응용이 가능해진다[7].

B. 음절 단위 태깅

한국어의 경우 영어와는 달리 조사가 어미로 붙는 경우가 많기 때문에 단어가 띄어쓰기로 명확하게 구분되지 않는다. 따라서 대부분의 형태소 분석기는 한국어 맞춤법 규칙들을 활용해서 어느정도 사전에 분석을 해두고[6], 그 후에 HMM이나 조건부 랜덤 필드(이하 CRF) 등의 방법을 사용하여 품사를 유추해낸다. 그러나 딥러닝이 나오면서 최근 NLP의 트렌드는 언어학적인 이론을 사용하지 않고 가능한 모든걸 학습으로 해결하는 방향으로 바뀌게 되었는데, 그 예로써 음절 단위로 끊어서 어미 규칙 등에 해당하는 부분까지 학습을 해보려는 시도가 등장하게 되었다[8,9]. 실제로 이 연구의 목적 중 하나는 한글로 적을 수 있는 모든 일본어 발음의 수가 백여가지를 넘는데, 그것들을 어떤 규칙으로써 미리 입력해두지 않아도 자동으로 알아내게끔 하는 것이다.

C. 로케일 태깅

이 글에서 하고자하는 외래어 및 외국어 받아쓰기 인식의 경우, 각 음절의 발음이 중요한 특성feature이 될 것이라고 예상했기에 음절 단위 태깅을 사용했다. 태깅(혹은 레이블링) 문제에 있어서 탁월한 결과를 낸다고 알려진 CRF를 사용했으며, 실제로 동일한 방식으로 띄어쓰기를 자동으로 해주는 알고리즘이 연구된 적이 있다[10].

각 음절별로 '로케일 태그'라는 것을 달았는데, 이 로케일 태그는 ISO 639-1 코드[13]를 사용하며 가령 영어라면 en, 한국어는 ko, 일본어는 ja가 된다. 예를 들어 "우리 팀 한조 어디갔어"에 로케일 태그를 달게 되면 "우/ko 리/ko 팀/en 한/ja 조/ja 어/ko 디/ko 갔/ko 어/ko"가 된다.

입력되는 문자열을 x, 그 문자열의 로케일 태그들을 y라 할때 CRF는 다음 조건부 확률로 정의된다[10,11]:

$$p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_j \sum_{i=1}^{n} \lambda_j t_j(y_{i-1}, y_i, x, i) + \sum_k \sum_{i=1}^{n} \mu_k s_k(y_i, x, i)\right)$$

그리고 실제 정답인 y^*에 대한 정확도가 높도록 각 λ_j, μ_k를 학습하게 된다. 위 식에서 Z(x)는 정규화 상수이며 전이 함수 t_j와 상태 함수 s_k는 직접 설정 할 수 있는 특성이다. CRF에 관한 자세한 설명은 [12]를 참고하기 바란다.

III. 실험

A. CRF 설정

실제 구현은 python-crfsuite를 이용하였으며, 학습용 파라미터는 기본값을 사용했다. 특성으로는 현재 글자와 좌우 두 글자를 사용하였다. 실제로 실험에 쓰인 코드는 아래 페이지에서 확인할 수 있다[14]:

https://github.com/theeluwin/sscc-1st

여담으로, 이 코드는 굳이 로케일 태그가 아니더라도 그대로 적용할 수 있다. 가령 띄어쓰기를 할지 말지로 태그를 달았다면 [10]에서 소개한 띄어쓰기를 자동으로 해주는 알고리즘이 구현된다.

B. 말뭉치

'날개를 주세요' 등의 일본어 및 영어로 된 노래 가사를 발음대로 한글로 적은것과 나무위키의 '니시키노 마키' 항목과 같은 문서를 활용하여 직접 ko, ja, en 태깅을 해서 말뭉치를 만들어 냈다. 수작업으로 했기 때문에 큰 편이 아니며, 딥러닝 특성상 학습 데이터가 많아진다면 더 좋은 결과를 낼 수 있을 것으로 예상되었기에 말뭉치의 크기를 변화시켜가며 실험했다. 학습 데이터는 크기를 각각 1배수, 2배수, 4배수 정도가 되게끔 설정했으며 (이하 학습-1, 학습-2, 학습-3), 로케일 태그의 비율은 약 en : ja : ko = 1 : 3 : 16 정도가 되도록 비슷하게 설정했다.

C. 결과

테스트 데이터에 있던 "생각해보니 츠바사랑 마키 커플링이 없네"의 경우 "생/ko 각/ko 해/ko 보/ko 니/**ja** 츠/ja 바/ja 사/ja 랑/**ja** 마/ja 키/ja 커/en 플/en 링/en 이/ko 없/ko 네/ko"가 나왔다. 이는 '니코' 등에서 학습된 '니'가 ja로 인식된 것으로 보이며, '랑'의 경우 좌우 2글자씩이 전부 ja여서 그렇게 된 것으로 보이는데, 학습 데이터에 '랑'이 한국어 접속사로 쓰인 경우가 많지 않아서 학습하지 못한 것으로 예상된다. 반면 "토도케테 세츠나사니와"의 경우는 모두 ja로 잘 나왔다.

학습-3 데이터의 경우 세부적인 성능은 아래 표(Table 1)와 같다. 조건부 확률

로케일 태그	정밀도	재현율	F_1 점수	테스트 글자수
en	0.88	0.83	0.86	2,296
ja	0.94	0.94	0.94	5,602
ko	0.98	0.98	0.98	37,257
평균/합	0.97	0.97	0.97	45,155

TABLE 1. 학습-3 데이터의 성능 평가

이 높은 순으로 여러개를 태그하여 정밀도(precision)와 재현율(recall), 그리고 다음과 같이 정의되는 F_1 점수를 계산해봤다.

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

전체적인 정확도는 약 97% 정도가 되며, 학습-3 데이터에선 ko에 해당하는 글자가 총 77%나 되기 때문에 전부 다 ko로 태그해주는 방법을 기준선으로 놓고 비교했을 때도 훨씬 더 좋은 성능을 나타난다고 할 수 있다.

그리고 아래는 학습 데이터의 크기를 변화시켜가면서 성능을 비교해본 표(Table 2)다.

말뭉치	전체 글자수	F_1-en	F_1-ja	F_1-kr	F_1 평균
학습-1	24,470	0.78	0.89	0.97	0.95
학습-2	50,208	0.81	0.92	0.98	0.96
학습-3	100,936	0.86	0.94	0.98	0.97

TABLE 2. 학습 데이터 사이즈에 따른 성능 비교

확실히 학습 데이터의 크기가 커짐에 따라 전반적인 정확도가 올라가는 것을 볼 수 있다. 그리고 놀랍게도 학습-1 데이터의 경우, 전체 글자수가 테스트의 글자수보다 작은데도 상당히 높은 성능을 보였다.

IV. 결론

A. 의의

이 글에선 로케일 태깅을 CRF로 학습하여 한글 음절에 대해 자동으로 태깅 해주는 방법을 제시했다. 그동안 발전해온 한국어 형태소 분석기들 덕분에 한국어 NLP 처리도 많이 수월해졌지만 아직 외래어, 특히 외국어 받아쓰기들에 대해서 인식을 해내는 알고리즘이 없었으므로 새로운 제안이 된다. 외래어 인식은 개체명 인식(NER) 정확도를 올리는데 쓰일 수 있으며 외국어 받아쓰기의 경우 Word2Vec 등에 쓰일 말뭉치에서 제거하는 방식으로 활용이 가능하다. 한본어 분석이나 새로운 외래어를 알아내는 데에도 사용될 수 있을 것이다.

B. 향후 연구 방향

CRF와 같은 딥러닝 알고리즘을 사용하지 않고 좀 더 고전적인 방법들이나 순수한 규칙 기반으로도 구현을 해서 기준선을 잡고 그것들과 정확도를 비교해보면 좋을 것이다. 또한 CRF의 특성을 어떻게 정의하냐에 따라, 혹은 말뭉치에서 외래어와 외국어 받아쓰기 비율에 따라 성능이 어떻게 변하는지를 살펴보면 좀 더 높은 정확도를 지닌 알고리즘을 만들 수 있을 것이다.

References

[1] 정희원, 새국어생활 **14**.2, 5-22 (2004)

[2] 강경인, 영어영문학연구 **23**.1, 1-26 (1997)

[3] K. W. Church, in *Proceedings of the second conference on Applied natural language processing*, (Association for Computational Linguistics, Stroudsburg, 1988), p. 136-143.

[4] E. Charniak, AI magazine **18**.4, 33 (1997)

[5] L. E. Baum and T. Petrie, The annals of mathematical statistics **37**.6, 1554-1563 (1966)

[6] 심광섭 and 양재형, 정보과학회논문지: 소프트웨어 및 응용 **31**.1, 89-99 (2004)

[7] 명재석, 이동주, and 이상구, 정보과학회논문지: 소프트웨어 및 응용 **35**.6, 392-403 (2008)

[8] 심광섭, 인지과학 **22**.3, 327-345 (2011)

[9] 권오욱, *et al*., in *1999 년도 제 11 회 한글 및 한국어 정보처리 학술대회 및 제 1 회 형태소 분석기 및 품사태킹 평가 워크숍* (한국정보과학회 언어공학연구회, 1999), p. 76-87.

[10] 심광섭, 인지과학 **22**.2, 217-233 (2011)

[11] J. Lafferty, A. McCallum, and F. C.N. Pereira, in *Proceedings of the eighteenth international conference on machine learning, ICML*. (2001), Vol. 1, p. 282-289.

[12] C. Sutton and A. McCallum, arXiv:**1011**.4088

[13] "ISO 639-2 Language Code List - Codes for the Representation of Names of Languages," *Library of Congress*. (18 Mar. 2014), https://www.loc.gov/standards/iso639-2/php/code_list.php, Web. 18 Jul. 2016.

[14] theeluwin, "sscc-1st," *GitHub*. (18 Jul. 2016), https://github.com/theeluwin/sscc-1st, Web. 18 Jul. 2016.