딥 러닝을 이용한 문맥 의존 철자 오류 교정

황현선, 최경호, 이창기
강원대학교 컴퓨터과학과
{hhs4322, gtraccoon, leeck}@kangwon.ac.kr

Context-sensitive Spelling Error Correction using Deep Learning

Hyunsun Hwang, Kyoungho Choi, Changki Lee
Dept. of Computer Science, Kangwon National University

요 약

문맥 철자 오류란 단어만 봤을 때에는 오류가 아니지만 문맥상으로는 오류인 문제를 말한다. 이 문제를 해결하기 위해서는 문맥 정보를 보아야 하는데 기존의 방법들은 언어학의 전문가가 설계한 규칙을 사용하거나, 통계적인 분석 방법을 사용하였다. 하지만 이 방법들은 많은 시간과 노력을 필요로 하지만 높은 성능을 얻지 못한다. 본 논문에서는 최근 자연언어처리에서 연구되고 있는 딥 러닝을 사용하여 문맥 철자 오류 교정을 시도하였다. 실험 결과 자질 설계 등의 복잡한 작업 없이 워드 임베딩 만을 사용하여 해당 단어들에 대해 F1-measure 91.43 ~ 97.27%의 성능을 보였다.

1. 서론

텍스트를 전산 상에 기록할 때 발생한 오타, 작성자의 어휘적 오인 등으로 발생하는 철자 오류는 자연언어 처리 과정에 있어 지속적으로 오류를 누적 시킨다. 텍스트 내의 철자 오류를 한국어 자연언어 처리 과정의 초반 부에 해당하는 형태소 분석 전후로 처리해 내지 못한다면 해당 텍스트를 처리하는 후위 과정에 있어서 큰 오류를 야기할 수 있다.

이러한 철자 오류는 크게 단순 철자 오류(Non-word Spelling Error)와 문맥 의존 철자 오류(Context-sensitive Spelling Error) 두 가지로 나눌 수 있다[1]. 단순 철자오류는 “병 -이 낫- -다 .” 라는 문장에서 철자 오류로 발생할 수 있는 문장인 “병 -이 낙- -다 .” 라는 예문의 “낙-” 처럼 형태소 사전에 존재하지 않아 어절 단위의 사전 검색 만으로 발견 할 수 있는 철자오류를 뜻하고, 문맥 의존 철자 오류는 “병 -이 낳- -다” 처럼 어절 단위의 분석 만으로 알아낼 수 없고, 전후 문맥을 조회해야 알아 낼 수 있는 오류를 뜻한다.

문맥 의존 철자 오류를 발견하는 기술은 고도의 배경 지식을 필요로 하기 때문에 한국어에서는 규칙 기반의 연구가 많이 수행되었다. 보통 규칙으로 시스템을 만들려면 전산학과 고도의 언어학적 지식을 갖춘 전문가의 많은 시간과 노력이 필요하다[2]. 또 기존에 규칙으로 구축해 놓지 않은, 새로운 형태의 철자 오류가 등장했을 때 해당 시스템으로는 처리 할 수 없으며, 더 많은 오류를 잡아내기 위해 규칙을 추가할수록 시스템의 복잡성은 커지게 된다.

본 논문에서는 최근 자연언어처리 연구에 새롭게 대두 되고 있는 딥 러닝(Deep Learning)을 문맥 의존 철자 오류 교정에 적용한다. 또한 자연언어 처리를 위한 딥 러닝의 사전 학습 방법으로 워드 임베딩(Word Embedding)[3,4]을 사용했고, 과적합(Overfitting)을 방지하기 위한 방법으로 Drop-out[5]을 사용했으며, 활성화 함수(Activation Function)로는 ReLU(Rectified Linear hidden Unit) [6]를 사용하여 성능을 높였다.

2. 관련연구

기존의 문맥의존 철자 오류 교정에 대한 연구는 보통 구문 분석을 이용한 규칙 기반의 방법으로 많이 수행되었다. 해당 방법을 이용한 가장 최근의 연구의 정확도는 Precision 98.51% Recall 27.92%로 나타났다[1]. 또한 N-gram 모델을 사용하여 다른 단어와 해당 단어가 같이 등장할 확률을 통계적으로 분석하여 문맥의존 철자오류를 발견하는 연구에서는 평균 Precision 92.03%과 평균 Recall 85.60%의 성능을 보였다[2].

일반적으로 딥 러닝 시스템은 히든 레이어(hidden layer)를 여러층 쌓은 구조로 인해 다양한 입력된 자질의 조합을 학습하고, 높은 수준의 추상화(Abstraction)를 할 수 있다[7]. 하지만 이로 인한 과적합과 Curse-of-Dimensionality로 인해 자연언어 처리에 있어서 사용된 전례가 많지 않으나, 최근 One-hot-representation으로 표현된 단어를, 연관성을 보존한 N차원 벡터로 차원축소를 하는 방법인 워드 임베딩 등의 자연언어 처리에 특화된 사전훈련과, 과적합을 방지하기 위한 Drop-out 등의 방법들이 등장하면서 자연언어 처리 연구에 응용되고 있다[8].

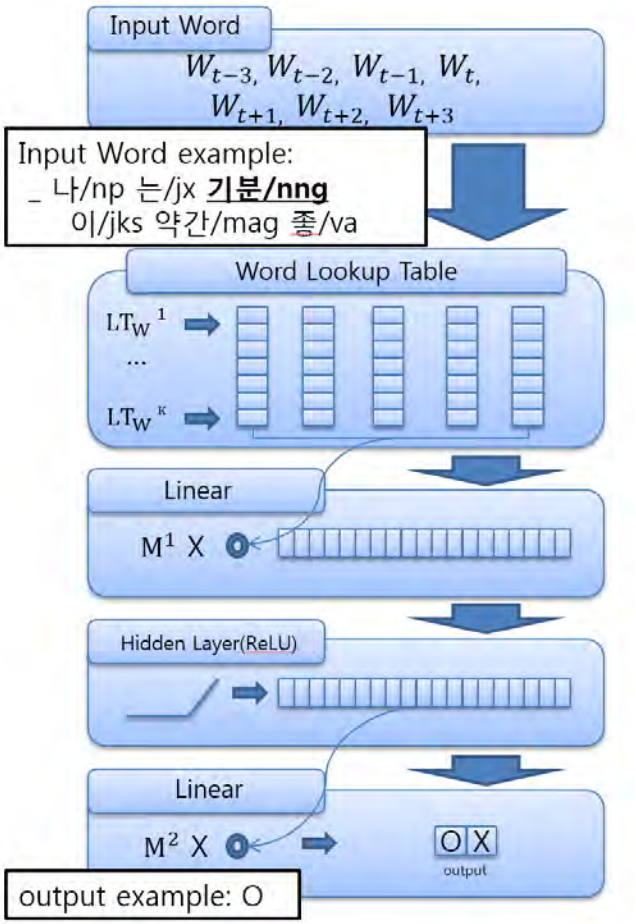
3. 딥 러닝을 기반 문맥 의존 철자 오류 탐지

본 논문에서 사용한 시스템은 아래 (그림 1)과 같이 자연언어 문장에서 문맥의존 철자오류가 발생할 수 있는 형태소를 찾아 그 형태소의 앞 뒤 3개의 형태소와 해당 형태소 총 7개의 형태소를 입력으로 받는다. 입력받은 형태소들을 사전학습인 워드 임베딩으로 구축된 Word Lookop Table에서 조회하여 각 형태소에 맞는 50차원 벡터로 변환한다. 이때 형태소가 Word Lookup Table에 없다면, 없는 형태소를 뜻하는 "UNK"로 치환하여 해당 벡터로 변환한다. 변환된 벡터는 히든 레이어를 거쳐 철자오류의 유무가 결정된다.

애틋/xr 하/xsa ㄴ/etm **기본/nng** 이/jkc 되/vv ㄹ/etm
재래/nng 의/jkg 명절/nng **기본/nng** 을/jko 소롯이/mag 즐기/v
_ 나/np 는/jx **기분/nng** 이/jks 약간/mag 좋/va
"/ss 언니/nng ,/sp **기본/nng** 나쁘/va 아/ec 하/vx

(그림 1) 신경망의 입력 예시

본 논문에서는 한글 및 한국어 정보처리학술대회에서 발표된 딥 러닝 논문에서 사용한 방법으로, 29억 개의 형태소를 원시 말뭉치(raw corpus)로 이용하여 102587개의 단어를 학습한 워드 임베딩을 사용했다[8]. (그림 2)와 같이 한 층의 ReLU 히든 레이어를 사용했으며, 학습과정에서 50%의 확률로 유닛을 비활성화 시키는 Drop-out 기술을 사용했다.



(그림 2) 문맥 철자 오류 수정을 위한 신경망과 입출력 예시

4. 실험

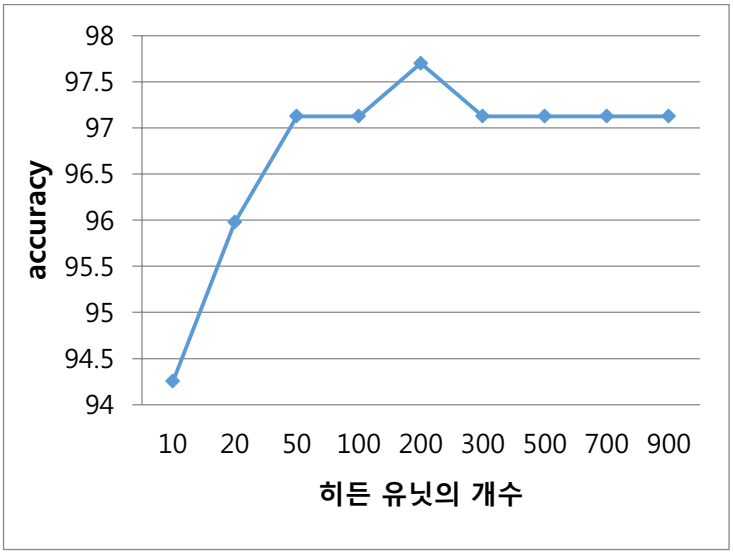
한 음소가 바뀌었을 때 문맥의존 철자오류가 발생 할 수 있는 단어들 중 (배, 베), (짚, 집), (사장, 사정), (주의, 주위), (의지, 의자), (찢, 찧) 6쌍을 교정 대상으로 했고, 세종 품사부착 말뭉치에서 교정 대상이 포함된 문장을 추출해 데이터 셋을 만들었다. 또 시스템의 학습과 평가를 위해 선행 연구[1,2]에서 실험한 바와 같이 데이터 셋에 존재하는 교정대상 단어들 중 50%에 대하여 고의적으로 상대되는 형태소로 바꿔 오류를 만들었다.

구축한 데이터 셋의 양은 <표 1>과 같으며, 추출한 데이터를 4:1의 비율로 나누어 학습 셋과 평가 셋을 구축했다.

단어	문장 수	단어	문장 수
베	458	배	412
짚	406	집	710
사장	2015	사정	1740
주의	2000	주위	1586
의지	1726	의자	889
찢	328	찧	161

<표 1> 실험에 사용한 데이터의 수

각 단어 쌍 별로 히든 유닛의 개수를 아래 (그림 3)과같이 변경 하면서 실험을 수행하여, 각 쌍에 대해 가장 좋은 성능을 보인 히든 유닛의 개수를 시스템에 사용하였다. 그 중 (배, 베) 쌍에 대한 히든 유닛 개수에 따른 accuracy 성능은 (그림3)과 같다.



(그림 3) (베, 배) 쌍에 대한 히든 유닛 개수에 따른 accuracy

(그림 3)에서 보는 보이는 바와 같이 히든 유닛이 일정 개수보다 많아지면 성능 향상이 없다.

시스템을 (베, 배) 단어 쌍에 대해 최적화 시켰던 방법과 같은 방법으로 각각의 단어 쌍들에 대한 최적의 히든 유닛 개수와, 그에 따른 성능을 평가해 아래 <표 2>로 나타냈다.

target1	target2	히든 유닛 개수	Precision	Recall	F1
베	배	200	97.85	97.85	97.85
짚	집	20	97.14	89.47	93.15
사장	사정	300	91.93	94.95	93.42
주의	주위	300	96.61	97.71	97.16
의지	의자	20	97.65	96.89	97.27
찢	찧	50	94.12	88.89	91.43

<표 2> 단어 쌍 별 최적의 히든 유닛 개수와 딥 러닝 성능

평가된 6개의 단어 쌍 모두 300개 초과의 히든 유닛을 필요로 하지 않았다. 또 (찢, 찧) 단어 쌍의 성능이 가장 낮게 나왔는데 이는 데이터의 양이 부족했고, 단어 별로 특성이 다르기 때문인 것으로 보인다.

본 논문에서 제안한 방법과 기존의 규칙 기반의 방법[1]과 N-gram 방법[2]을 <표3>으로 비교해 보았다.

* 기존 연구와 본 논문의 사용한 데이터 셋은 다름

target1	target2	딥 러닝	기존연구[1]	기존연구[2]
베	배	97.85	42.52	-
짚	집	93.15	22.61	-
사장	사정	93.42	-	86.36
주의	주위	97.16	-	90.48
의지	의자	97.27	-	92.43
찢	찧	91.43	46.97	-

<표 3> 기존 연구와의 성능비교[1,2]

실험에서 사용한 모든 단어 쌍들에 대해 기존의 연구보다 우수한 성능을 보였다. 특히 동사로 된 단어 쌍 들에서 더 눈에 띄는 성능 향상을 보였다.

5. 결론

본 논문에서는 딥 러닝을 이용한 문맥기반 철자오류 시스템을 제안하였다. 본 논문에서 제안한 딥 러닝 기반의 시스템은 형태소 단위의 자질만 갖고도 기존의 규칙 기반이나 N-gram 기반의 연구들에 비해 우수한 성능을 보여, 고도의 언어학 지식을 갖춘 전문가의 노력 없이도 딥 러닝을 이용하여 우수한 성능의 시스템을 구축할 수 있음을 보였다.

추후 연구로는 본 논문에서 3으로 고정했던 window-size를 다양하게 실험하고, 다양한 자질을 추가하여 시스템의 성능을 향상시킬 계획이다.

사사

본 연구는 미래창조과학부 및 한국산업기술평가관리원의 산업융합원천기술개발사업(정보통신)의 일환으로 수행하였음[10044577, 휴먼 지식증강서비스를 위한 지능진화형 WiseQA 플랫폼 기술 개발]

참고문헌

[1] Hyunsoo Choi, Aesun Yoon, Hyuk-chul Kwon. “Improving Recall for Context-Sensitive Spelling Correction by Weakening Constraints on Case Markers.” *Journal of KIISE*, 2014, 41.3.

[2] Minho Kim, Hyuk-chul Kwon, Sungki Choi. “Context-sensitive Spelling Error Correction using Eojeol N-gram.” *Journal of KIISE*, 2014, 41.12: 1081-1089.

[3] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, Pavel Kuksa. “Natural language processing (almost) from scratch.” The Journal of Machine Learning Research, 12, 2011

[4] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, Christian Janvin. “A neural probabilistic language model.” In NIPS, 2001.

[5] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, Ruslan R. Salakhutdinov. “improving neural networks by preventing co-adaptation of feature detectors.” CoRR, abs/1207.0580, 2012

[6] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. “Deep sparse rectifier networks.” Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR W&CP Volume. Vol. 15. 2011.

[7] Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh. “A fast learning algorithm for deep belief nets.” Neural computation, 18(7), 2006.

[8] Changki Lee, Junseok Kim, Jeonghee Kim, “Korean Dependency Parsing using Deep Learning.” *Proceedings of 26th Hangul and Korean Information Processing Conference*, 2014.