SNS 채팅 데이터에 적응적인 Self-Attention 기반 문맥의존 철자오류 교정 시스템

최혜원[O*], 장대식[**], 손동철[*], 이승욱[*], 고영중[***]
동아대학교 컴퓨터공학과[*], 동아대학교 수학과[**], 성균관대학교 데이터사이언스융합학과[***]
{hyewon4999, daesik0320, sondongcheol94, seungwooklee76, youngjoong.ko}@gmail.com

Adaptive Context-Sensitive Spelling Error Correction System Based on Self-Attention for Social Network Service Chatting Data

Hyewon Choi[O*], Daesik Jang[**], Dongcheol Son[*], Seungwook Lee[*], Youngjoong Ko[***]
Department of Computer Science and Engineering, Dong-A University[*]
Department of Mathematics, Dong-A University[**]
Department of Applied Data Science, Sungkyunkwan University[***]

요 약

본 논문에서는 Self-Attention을 활용한 딥러닝 기반 문맥의존 철자오류 교정 모델을 제안한다. 문맥의존 철자오류 교정은 최근 철자오류 교정 분야에서 활발히 연구되고 있는 문제 중 하나이다. 기존에는 규칙 기반, 확률 기반, 임베딩을 활용한 철자오류 교정이 연구되었으나, 아직 양질의 교정을 수행해내기에는 많은 문제점이 있다. 따라서 본 논문에서는 기존 교정 모델들의 단점을 보완하기 위해 Self-Attention을 활용한 문맥의존 철자오류 교정 모델을 제안한다. 제안 모델은 Self-Attention을 활용하여 기존의 임베딩 정보에 문맥 의존적 정보가 반영된 더 나은 임베딩을 생성하는 역할을 한다. 전체 문장의 정보가 반영된 새로운 임베딩을 활용하여 동적으로 타겟 단어와의 관련 단어들을 찾아 문맥의존 철자 오류교정을 시행한다. 본 논문에서는 성능평가를 위해 세종 말뭉치를 평가 데이터로 이용하여 제안 모델을 실험하였고, 비정형화된 구어체(Kakao Talk) 말뭉치로도 평가 데이터를 구축해 실험한 결과 비교 모델보다 높은 정확율과 재현율의 성능향상을 보였다.

주제어: Self-Attention, 문맥의존 철자오류, 한국어 맞춤법 검사기

1. 서론

철자 오류는 크게 두 가지로 나눠며, 단순 철자 오류(Non-Word Spelling Error)와 문맥의존 철자오류(Context-Sensitive Spelling Error)로 구분된다. 단순 철자 오류는 잘못된 철자 입력으로 단어의 형태만 보아도 오류 유무를 판단할 수 있다. 문맥의존 철자오류는 옳은 단어 형태이지만 좌우 문맥을 고려했을 때, 의미상 올바르지 않은 단어를 말한다. 이는 좌우 문맥의 의미를 파악해야 해당 단어의 오류 여부를 알 수 있으므로 교정 난도가 매우 높다.

기존의 많은 문맥의존 철자오류 교정 연구들은 주로 규칙 기반, 확률 기반, 단어 임베딩을 활용한 오류교정을 수행해 왔다. 규칙 기반 교정 모델은 규칙을 만들고 검증하는데 심도 있는 언어학지식을 갖춘 전문가가 필요하며 규칙의 수가 늘어날수록 규칙을 만드는 속도가 매우 느려지는 단점이 있다.

확률 기반 모델은 언어 모델 N-Gram을 이용하여 확률값을 구한 뒤, 문장 내에서 타겟 단어 주변 좌우 문맥의 확률값을 활용하여 교정하는 방법이다. 이러한 방법은 고정된 윈도우 사이즈 내에서만 타겟 단어와 관련된 단어를 찾아 확률값을 구하므로, 해당 윈도우 사이즈 내에서 관련된 단어가 나타나지 않으면 확률이 0이 되는 문제가 발생하거나 잘못 교정될 확률이 매우 높다.

[표 1]을 보면 타겟 단어 '사전'은 예시 문장의 좌우 문맥을 고려해 '사전'보다는 '사진'이 문맥적으로 올바른 단어라고 할 수 있다. 하지만 '사전'의 좌우 문맥이 '사진'과 관련된 단어가 없기 때문에 이는 '사진'으로 올바르게 교정될 확률이 매우 낮다.

표 1. 확률 기반 문맥의존 철자오류 예시

타겟 단어 : 사전 교정후보(Distance 1) : 사진, 사전
한 달 전에 **카메라**를 샀는데 동생이 떨어뜨려 **사전이** 다 날아갔다.

최근 단어 임베딩을 활용한 문맥오류 철자오류 교정 방법도 활발하게 연구되고 있다. 하지만 단어 임베딩을 활용한 교정 방법은 전체 문맥에 상관없이 매번 같은 단어 임베딩을 활용해서 교정을 수행한다는 단점이 있다.

예를 들어 '카메라의 사용성과 안정성이 향상되었습니다.'와 '카메라를 사서 사진으로 작품을 만들려고

이 논문은 2019년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00550, 기계학습용 텍스트 데이터 레이블 자동생성 및 검증도구 개발)

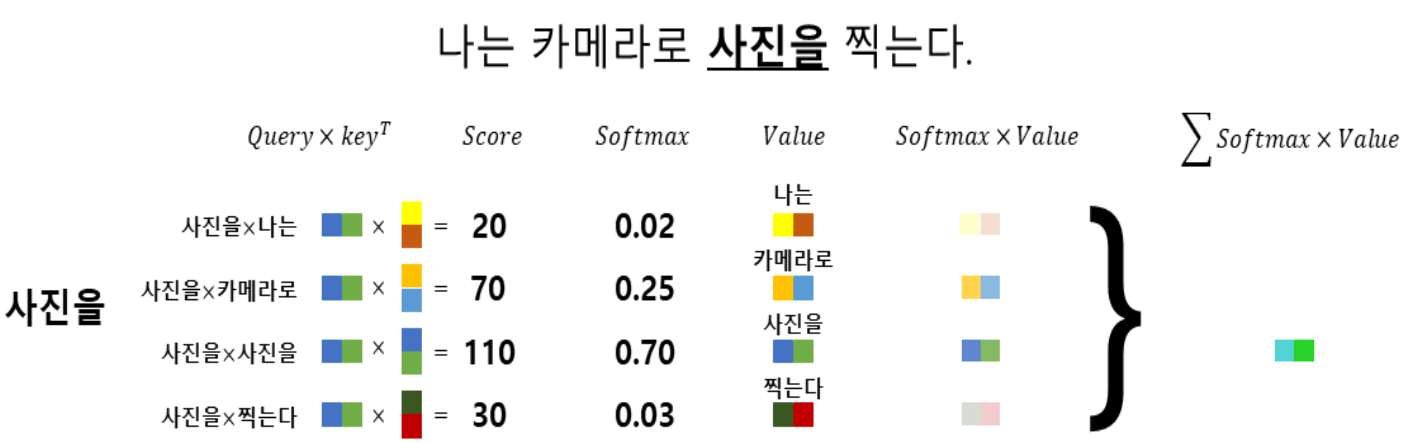

그림1. Self-Attention 과정

합니다.' 이 두 문장에서 단순히 단어 임베딩을 사용하면 앞 문장의 카메라와 뒷 문장의 카메라는 동일한 단어 임베딩일 것이다. 만약 매번 같은 단어 임베딩이 아닌 전체 문맥의존적인 정보를 포함한 임베딩이라면 기존의 단어 임베딩 보다 더 좋은 임베딩의 역할을 할 것이다.

본 연구진들은 이러한 문제점을 Self-Attention을 활용해 해결한다. 기존의 단어 임베딩을 Self-Attention을 통해 매번 동일한 단어 임베딩이 아닌 전체 문장의 문맥의존적인 정보가 포함된 새로운 임베딩을 구한다. 타겟 단어 임베딩과 전체 단어 임베딩들 간의 내적을 통해 전체단어 중에서 타겟 단어와 가장 연관 있는 단어를 동적으로 알아내어 기존 임베딩보다 문맥의존적인 정보를 더욱 풍부하게 반영한 임베딩을 생성하여 철자 오류교정을 시행한다. 실험 결과 확률 기반 모델보다 제안 모델의 성능이 6.6%p 향상 되었다.

논문의 구성은 다음과 같다. 2장에서는 관련 연구를 소개하고, 3장에서는 본 논문에서 사용한 단어 임베딩의 설명과 제안 모델을 설명한다. 4장에서는 제안 모델의 실험과 그 결과를 분석하고, 마지막 5장에서 결론과 향후 계획에 대해 기술한다.

2. 관련 연구

문맥의존 철자오류의 교정 방법은 크게 규칙을 이용하는 방법과 확률 정보를 이용한 방법, 그리고 임베딩을 활용한 방법으로 나눌 수 있다. 현재 가장 많이 사용되는 것은 규칙 기반 방식으로, 이를 활용한 편집거리 알고리즘[1]과 모델의 속도 및 교정 성능을 높이기 위한 다양한 방법이 연구되었다[2, 3]. 규칙 기반 방식은 발생 빈도가 높은 오류 및 정형화된 문맥의존 철자오류에서는 높은 교정 성능을 보이지만, 입력 오류로 발생한 비정형 오류에는 높은 성능을 보여주질 못한다.

확률 정보를 활용한 문맥의존 철자오류 교정은 영어를 대상으로 다양한 연구간 진행되어 왔다[4]. 한글을 대상으로 한 방법에는 확률 모델과 언어 모델을 결합한 채널 모델(Noisy-Channel Model)이 대표적이다[5]. 확률 정보를 활용한 방법은 반복성이 적은 문맥의존 철자오류에도 적용할 수 있고, 확률을 구할 때 사용하는 말뭉치를 바꿈으로 다양한 환경에 적절히 대응할 수 있다는 장점이 있다. 이는 빈도수가 낮고, 정형화되지 않은 문맥의존 철자오류에 낮은 성능을 보여주던 규칙 기반 모델과 대조적이다.

확률 정보를 활용하는 방법은 모델이 참고할 수 있는 정보의 양이 많을수록 교정품질이 좋아지는 장점이 있으나, 고정된 윈도우 사이즈 내에서만 관련 확률을 찾아야 하는 단점과 확률사전이 커질수록 처리 속도가 느려지고 저장 공간의 크기가 급격히 늘어난다는 단점이 있다. 이러한 단점을 개선하거나, 교정 성능을 높이기 위한 다양한 방법이 연구되었다[6, 7].

워드임베딩을 활용한 오류교정도 활발히 진행되었다[8]. 워드임베딩은 단어를 특정 숫자와 매핑 시켜 벡터화하는 기술로, 크게 두 가지 방법으로 분류된다. 이는 각각 빈도 기반 임베딩(빈도 벡터, TF-IDF 벡터, 동시발생 벡터 등)과 신경망 기반 임베딩(Word2Vec, Fasttext, ELMo)이다. 이 방법은 임베딩 정보를 기반으로 교정 단어와 문맥간의 관계를 쉽게 파악하여, 보다 정확한 정보가 반영된 교정을 수행한다.

3. Self-Attention을 활용한 문맥의존 철자오류 교정 모델

3.1 Self-Attention

[그림 1]은 Self-Attention 수행 과정을 도식화한 것이다. 실제로는 행렬 곱으로 문장 내에 있는 모든 어절이 한 번에 연산된다. 이해를 돕기 위하여 '사진을'이라는 단어를 기준으로 문장의 다른 단어와의 상관관계를 구할 때, Query를 다른 단어들의 Key와 곱해준다. Query와 Key를 곱한 값을 Attention-Score라고 한다. Attention-Score 값이 클수록 단어와의 연관성이 높고 낮을수록 연관성이 낮다. Attention-Score 값을 확률개념으로 바꾸기 위해서 Softmax를 적용한다. Softmax의 결과값은 key에 해당하는 단어가 타겟 단어 Query에 어느 정도 연관성이 있는지 나타낸다. 예를 들어 타겟 단어 '사진을'은 자기 자신과 70%, '나는'과 2%, '카메라로'와 25%, '찍는다'와는 3% 연관성이 있다고 볼 수 있다. 각 Softmax 확률값을 Key에 해당하는 Value와 곱해

준다. 이로 인해 타겟 단어와 연관성이 없는 단어들의 임베딩 정보는 희미해진다. 최종 생성된 임베딩은 기존의 '사진을'의 단어 임베딩이 아닌, 문장의 전체적인 문맥 의미가 반영된 임베딩이라고 간주할 수 있다.

3.2 단어 임베딩

본 논문에서는 보단 나은 단어 임베딩 벡터 만들기 위하여 사전 학습을 수행한다. 단어 임베딩을 사전 학습하기 위해 5GB의 뉴스 말뭉치를 사용하였고, GloVe모델을 사용하였다. 단어의 차원은 512차원으로 설정하였고, 학습 단위는 어절 단위로 사용하였다.

3.3 제안 모델 구성

본 논문에서 제안하는 모델은 [그림 1]와 같이 문장의 모든 어절 단위 토큰을 임베딩 벡터로 매핑하고, 매핑된 임베딩 벡터($e \in \mathbb{R}^{512}$)에 Self-Attention을 적용하여 전체문장의 문맥 의존적 정보가 반영된 모든 어절의 임베딩 벡터($\hat{e} \in \mathbb{R}^{512}$)를 출력한다. 문맥정보가 반영된 모든 어절의 임베딩 벡터 $\hat{e}$들을 다시 Transpose 한 후, 내적하여 Attention-Score 행렬을 구한다. Attention-Score 행렬에 가중치 α를 곱한다. 가중치 α에 대한 설명은 [그림 3]에서 자세히 설명한다.

$$\alpha \times \begin{bmatrix} \hat{e_1} \\ \hat{e_2} \\ \vdots \\ \hat{e_j} \\ \vdots \\ \hat{e_n} \end{bmatrix} \times \begin{bmatrix} \hat{e_1} & \hat{e_2} & \cdots & \hat{e_j} & \cdots & \hat{e_n} \end{bmatrix} = \begin{bmatrix} d_{11} & \cdots & d_{1n} \\ \vdots & \ddots & \vdots \\ d_{n1} & \cdots & d_{nn} \end{bmatrix}$$

그림2.Attention-Score 계산과정

Attention-Score 행렬을 구하여, 타겟 단어의 위치인 j행만을 가중치 연산한다. [그림 3]은 타겟 단어의 주변 Attention-Score 가중치 α 연산과정이다. Attention-Score 값에 타겟 단어 좌우 문맥 $j-1$과$j+1$에 위치한 단어들은 가중치 값 z를 곱하고, $j-2$와 $j+2$는 가중치 y, $j-3$와 $j+3$은 가중치 x, $j-4$와 $j+4$은 가중치 w 그리고 $j-5$와 $j+5$은 가중치 v을 곱한다. 나머지 단어들은 1을 곱해 가중치를 부여하지 않았다. 타겟 단어와 가까운 단어일수록 더 높은 가중치 값을 부여했다. 이는 타겟 단어의 주변 문맥은 타겟 단어와 연관되어 있을 단어가 등장할 확률이 높을 것이라는 가설을 세운 뒤 실험을 진행한 결과 실제로 가중치를 주는 모델이 가장 높은 성능을 보였다. Attention-Score에 가중치를 곱한 값을 모든 어절 중에 타겟 단어를 제외한 상위 m개를 뽑아 모두 합하여 최종 타겟 단어의 Score를 구한다. [그림 3]의 '사진을'의 최종 Score는 415 이다. 이 작업을 타겟 단어의 교정후보 갯수만큼 진행하여 그 중 가장 높은 최종 Score를 가진 교정후보가 최종교정후보로 선정된다.

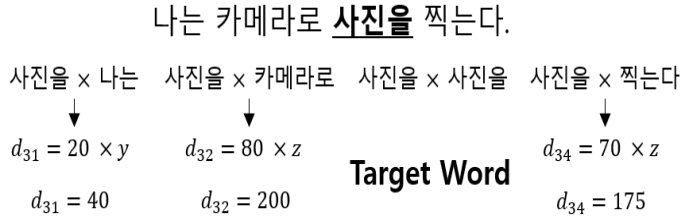

그림 3. 타겟 단어의 주변 Attention-Score 가중치 연산

$$Attention(Q, K, V) = softmax(QK^T)V$$

$$d_{jk} = \alpha\left(\hat{e_j} \bullet \hat{e_k}^T\right) \quad \text{if } 1 \le k \le n,\ j \ne k \qquad \cdots \text{식(1)}$$

$$\alpha = \begin{cases} v & \text{if } k = j-5 \text{ or } k = j+5 \\ w & \text{if } k = j-4 \text{ or } k = j+4 \\ x & \text{if } k = j-3 \text{ or } k = j+3 \\ y & \text{if } k = j-2 \text{ or } k = j+2 \\ z & \text{if } k = j-1 \text{ or } k = j+1 \\ 1 & otherwise \end{cases} \qquad \cdots \text{식(2)}$$

$$D_i = \{d_{jk} \mid 1 \le k \le n,\ j \ne k\} \qquad \cdots \text{식(3)}$$

$$T_i = \{t_1, t_2, t_3, ..., t_m\} \qquad \cdots \text{식(4)}$$

$$S_i = \sum_{p=1}^{m} t_p \qquad \cdots \text{식(5)}$$

$$S* = \underset{i}{\arg max}\ S_i \qquad \cdots \text{식(6)}$$

i : 교정 후보 번호
j : Target Word의 index
n : 문장의 총 어절의 수
d_{jk} : Target Word의 Attention-Score
α : Attention-Score 가중치 값

문맥 의존적 정보가 반영된 새로운 임베딩 벡터를 $\hat{e} \in \mathbb{R}^{512}$라고 할 때, $\hat{e}$을 활용하여 타겟 단어와 전체 문장의 어절 사이의 관계를 알아내기 위하여 (식1)과 같이 타겟 단어와 전체문장 어절과의 내적을 구한다.

(식2)는 내적을 구하여 나온 Attention-Score 값에 타겟 단어 기준으로 왼쪽 문맥 β개, 오른쪽 문맥 β개 어절은 가중치 α를 부여한다. 좌우 문맥 β개 내에서 타겟 단어와 가까운 단어일수록 더 높은 가중치 값을 부여했다. (식3)은 타겟 단어와 문장의 모든 어절의 내적을 활용해 Attention-Score 집합을 만든다. 그 중 Attention-Score가 높은 상위 m개를 선출하여 (식4)의 집합을 만든다. 상위 m개의 Attention-Score를 모두 합하여 최종 타겟 단어의 Score를 구하고(식5), 이 작업을 타겟 단어의 교정후보 갯수만큼 진행하여 그 중 가장 높은 Score를 가진 교정후보가 최종후보로 선정된다(식6). 이러한 방식은 수행시간이 오래 걸릴 것으로 보이지만, Self-Attention 연산이 행렬 곱으로 병렬적으로 수행되기 때문에 수행 시간이 큰 폭으로 증가하지는 않는다.

4. 실험 및 평가

4.1 실험환경

신뢰할 수 있는 결과를 위해, 모든 시스템은 동일한 데이터로 평가를 진행하였다. 확률 모델[5]의 학습 데이터로는 약 20GB의 뉴스 기사 데이터를 수집하여 학습에 이용했다. 평가 데이터는 정형 데이터와 비정형 데이터, 두 가지 말뭉치를 사용하였다. 정형 데이터는 국립국어원이 제공하는 세종 말뭉치 중 10,000문장을 추출하였고, 비정형 데이터는 카카오톡 데이터로 구축해놓은 구어체 말뭉치에서 70,000건의 사용자 메시지를 추출하여 평가 데이터로 활용하였다.

4.2 학습 데이터

학습 데이터는 확률 기반 모델[5]의 학습에 사용하였다. 확률 기반 모델에서는 뉴스 기사 말뭉치를 N-Gram 모델 학습에 사용하였다. [표 2]는 확률 기반 모델의 N-Gram별 데이터 수이다.

표 2. N-Gram별 데이터 구성

Words	uni-gram	bi-gram	tri-gram
1,756,764,975	15,431,221	339,486,862	1,001,929,201

4.3 명사 쌍을 이용한 평가

명사 쌍을 이용한 평가는 [5]에서 사용된 명사 어휘쌍을 활용한다. 해당 명사를 포함한 문장은 세종 말뭉치에서 추출하여 사용하였다. 명사 쌍으로 평가를 진행하는 이유는 용언보다 명사가 규칙으로 문맥철자 오류를 고치기 어렵고, 통계적 실험 결과도 좋지 않게 나오기 때문이다.

표 3. 명사 쌍 평가 데이터 구성

타겟 단어	옳은 문장	오류 문장
기본 *gibon* (basics)	800	200
기분 *gibun* (feelings)	800	200
자식 *jasik* (child)	800	200
지식 *jisik* (knowledge)	800	200
주의 *juui* (care)	800	200
주위 *juwi* (around)	800	200
사정 *sajeong* (reason)	800	200
사장 *sajang* (boss)	800	200
의지 *uiji* (will)	800	200
의자 *uija* (chair)	800	200

명사 쌍의 단어 중 하나의 단어가 대상어가 되면, 나머지 단어는 대치어가 된다. 명사 쌍을 이용한 평가 데이터는 대상어의 문장(실제 대상어를 포함하는 옳은 문장) 800개와 대치어의 문장(대치어를 포함하는 옳은 문장)에 대치어를 대상어로 임의로 변경한 틀린 문장 200개를 추출하였다. 예를 들어, '기본'의 성능을 평가하기 위해, 대상어 '기본'이 사용된 800개의 옳은 문장과 대치어 '기분'이 사용된 옳은 문장에서 '기분'을 모두 '기본'으로 임의로 바꾼 200개의 오류 문장을 만든다. [표 3]은 명사 쌍 평가 데이터에 대한 통계 수치이다. 본 연구진들은 [5]에서 제안하는 확률 기반 모델(결합모델1)을 직접 구현하여 본 논문에서 제안한 모델과 평가 비교하였다.

4.4 구어체 데이터를 활용한 평가

비문자 메시지(Short Message Service; SMS)와 소셜 네트워크 서비스(Social Network Service; SNS) 등, 비정형화된 문맥철자 오류를 다수 포한한 한국어 문서 정제의 요구가 날로 증가하고 있다. 또한, 실제 철자 오류 단어는 '명사 쌍' 만이 아닌 후보가 두 개 이상인 경우가 많고, 특정한 어절만이 아닌 전체 문장의 어절에서 적용할 수 있어야 한다. 따라서 본 연구진들은 실제 어플리케이션 관점에서 미리 구축한 사전에서 편집거리 1 이내의 교정후보들을 추출 후 교정성능을 평가하였다.

표 4. 사전 구축 현황

명사	서술어	신조어	구어체
19,201	11,484	1,389	105,077

모델 동작을 위한 사전은 명사 사전, 서술어 사전, 구어체 사전 그리고 신조어 사전으로 구성하였다. 명사 사전은 국립국어원 표준국어대사전에서 일상 주제로 판단한 28만 6,330개 단어를 추출한 후, 명사 단어 분류 및 현재 사용되지 않는 고어를 정제하여 명사 사전을 구축하였다. 서술어 사전은 28만 6,330개의 단어 중, 품사가 동사 및 형용사, 접속사와 부사인 단어 분류 및 자주 사용되지 않는 표현을 정제하여 서술어 사전을 구축하였다. 구어체 사전과 신조어 사전은 카카오톡 메시지 말뭉치(약 20만 어절)를 정제하여 사전을 구축하였다. 앞선 사전의 정제는 본 연구진들이 수동으로 직접 수행하였다. [표 4]는 사전 구축 현황에 대한 통계이다.

평가 데이터는 수집한 카카오톡 메시지 말뭉치 중, 70,000건의 사용자 메시지를 추출하여 구축하였다. 이때 사용자 메시지는 다량의 노이즈를 포함한 데이터로, 띄어쓰기 및 철자 오류가 함께 적용된 경우가 많다. 따라서 띄어쓰기 오류에 따른 오류전파가 철자 오류 교정 성능에 영향을 줄 가능성이 높으므로, 사전에 띄어쓰기 오류를 교정한 데이터를 평가에 활용하였다. 띄어쓰기 사전 교정과 평가를 위한 철자 오류 교정 정답 생성은 본 연구진들이 수작업으로 수행한 후, 평가 데이터로 활용하였다.

본 연구진들은 [1]과 [5]에서 제안하는 규칙 기반과 확률 기반 모델을 직접 구현하여 본 논문에서 제안한 모델과 평가 비교하였다.

4.5 평가 척도

모든 실험에 대한 성능 평가척도는 다섯 쌍의 모든 단어의 정확률(Precision), 재현율(Recall)을 통해 F1-Score를 구한 뒤 Macro-average F1-Score로 측정한다. 정확률은 시스템의 예측 중에서 올바른 예측인 비율이며, 재현율은 실제 정답 중에서 시스템이 올바른 예측을 한 비율을 나타낸다. F1-Score는 정확률과 재현율 사이의 조화평균(Harmonic Mean) 값이고, F1-Score로 나온 다섯 쌍의 성능을 산술평균(Arithmetic Mean)한 값이 Macro-average F1-Score이다.

$$\text{F1-Score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$\text{Macro-average F1-Score} = \frac{\sum^{N} \text{F1-Score}}{N}$$

4.6 하이퍼 파라미터 실험

명사 쌍의 검증 집합(Validation Set)을 활용하여 하이퍼 파라미터들을 실험한 결과이다. [표 5]의 v, w, x, y, z는 각각 타겟 단어 주변 윈도우 사이즈 내의 위치에 따른 가중치 값이고, m은 전체 어절 중 타겟 단어와 가장 Attention-Score가 높은 상위 m개 단어를 뜻하며 β는 타겟 단어 좌우 문맥 사이즈를 뜻한다.

실험 결과 조합5의 하이퍼 파리미터 조합의 모델이 가장 높은 성능을 보였다. 조합7은 가중치를 주지 않는다는 의미로 v, w, x, y, z 가중치를 1로 설정하였고, 이는 전체 조합 중 가장 낮은 성능을 보인다. 실험 결과를 통해 타겟 단어 주변문맥에 가중치를 설정한 효과를 확인하였다.

표 5. 하이퍼 파라미터 조합 별 성능비교

	v	w	x	y	z	m	β	Macro-average F1-Score(%)
조합1	1.0	1.0	1.0	1.0	1.5	4	1	82.11
조합2	1.0	1.0	1.5	1.5	1.5	4	3	85.29
조합3	1.5	1.5	1.5	1.5	1.5	4	5	84.01
조합4	1.0	1.0	1.5	2.0	2.5	2	3	86.80
조합5	1.0	1.0	1.5	2.0	2.5	4	3	**87.83**
조합6	1.0	1.0	1.5	2.0	2.5	6	3	87.22
조합7	1.0	1.0	1.0	1.0	1.0	4	0	81.73
조합8	1.0	1.0	2.0	2.5	3.0	4	3	87.13

4.7 확률 모델과 성능비교

[표 6]은 확률 모델과 임베딩 모델, 그리고 본 논문에서 제안하는 모델과의 명사 쌍을 이용한 성능 비교 평가표이다. 임베딩 모델은 본 모델에서 제안하는 Self-Attention을 적용하지 않은 모델이다. 확률 모델보다 임베딩을 활용하여 문장의 전체단어와의 유사도를 통해 교정하는 모델의 성능이 5.28%p 향상하여 좋은 성능을 보였고, 확률 모델보다 제안 모델의 성능이 6.6%p 향상되었다. 실험 분석 결과 확률 모델은 타겟 단어 주변의 고정된 윈도우 사이즈 내에 학습한 단어가 나오지 않는다면 OOV(Out Of Vocabulary)로 인한 문제로 확률이 0이 되거나, 제대로 교정하지 못하는 경우가 많았다. 그에 비해 제안 모델은 타겟 단어 주변의 윈도우 사이즈에 가중치를 부여하여, 타겟 단어 주변에 관련 단어가 나오면 관련 단어정보를 중점으로 교정에 필요한 정보를 얻고, 만약 타겟 단어 주변 문맥에서 관련단어가 나오지 않는다면 전체문맥을 보고 관련단어를 찾기 때문에 OOV(Out Of Vocabulary) 문제가 거의 일어나지 않아 확률 모델보다 높은 성능을 보였다. 본 실험의 결과로 고정된 사이즈 내에서만 타겟 단어와의 관련 단어를 찾는 것 보다 문장의 전체 단어 중에서 관련 단어를 동적으로 찾아 교정하는 방법이 좋은 성능임을 확인하였다. 임베딩만을 사용한 모델보다 Self-Attention을 사용하는 제안 모델의 성능이 1.32%p 향상하는 것을 보아 Self-Attention의 기여를 확인할 수 있다.

표 6. 확률 모델, 임베딩 모델, 제안 모델의 성능 비교(%)

타겟 단어	확률 모델	워드임베딩 모델	제안 모델
기본 *gibon* (basics)	Pre:90.37 Re:82.12 F1:86.04	Pre:97.81 Re:66.87 F1:79.43	Pre:98.87 Re:65.87 F1:79.06
기분 *gibun* (feelings)	Pre:94.08 Re:65.62 F1:77.31	Pre:90.14 Re:95.25 F1:92.62	Pre:92.84 Re:95.75 F1:94.27
자식 *jasik* (child)	Pre:91.23 Re:63.75 F1:74.09	Pre:97.76 Re:80.41 F1:88.24	Pre:97.97 Re:78.59 F1:87.22
지식 *jisik* (knowledge)	Pre:88.75 Re:74.00 F1:80.70	Pre:91.12 Re:93.77 F1:92.42	Pre:94.19 Re:95.25 F1:94.71
주의 *juui* (care)	Pre:87.98 Re:81.50 F1:84.61	Pre:95.18 Re:74.11 F1:83.33	Pre:95.19 Re:71.92 F1:81.94
주위 *juwi* (around)	Pre:94.20 Re:58.87 F1:72.45	Pre:90.11 Re:81.67 F1:85.68	Pre:93.84 Re:83.87 F1:88.58
사정 *sajeong* (reason)	Pre:95.61 Re:70.87 F1:81.40	Pre:93.90 Re:58.12 F1:71.79	Pre:96.95 Re:59.62 F1:73.83
사장 *sajang* (boss)	Pre:92.67 Re:85.37 F1:88.87	Pre:88.11 Re:89.33 F1:88.71	Pre:89.80 Re:91.83 F1:90.80
의지 *uiji* (will)	Pre:90.62 Re:89.37 F1:89.99	Pre:93.11 Re:93.15 F1:93.12	Pre:97.20 Re:95.75 F1:96.47
의자 *uija* (chair)	Pre:96.62 Re:68.00 F1:79.82	Pre:95.37 Re:90.11 F1:92.66	Pre:98.37 Re:90.62 F1:94.33
Macro-average F1-Score	81.52	86.80	**88.12**

4.8 비정형화된 데이터를 활용한 성능 비교

[표 7]과 [표 8]은 규칙 기반[1], 확률 기반[5], 제안 모델의 성능 평가표이다. 70,000개의 구어체 데이터(카카오톡 메시지)를 평가에 이용하였고, 에러율은 전체 어절 중 틀린 어절의 비율로 측정하였다. 기존의 평가데이터의 에러율은 6.12%였다. 이를 100%로 환산했을 때, 문맥의존 철자오류는 1.05%밖에 되지 않았고 나머지는 모두 단순 철자오류였다. 그래서 본 연구진들은 실험 결과를 극명하게 보여주기 위해 문맥 의존 철자 오류를 임의로 더 생성하여 에러율을 20.90%까지 늘렸다. 실험은 각 교정 모델을 거치고 난 후 평가데이터의 에러율을 측정하였다.

[표7]은 오류를 생성하기 전 모델 간 성능 비교이다. 규칙 기반 모델은 교정 전 에러율에 비해 10.37%p 에러 증가율을 보였고, 에러율이 더 높아지는 경향을 보였다. 이는 옳은 어절을 틀린 어절로 바꾸는 경우가 매우 많았고, 이는 어플리케이션 관점에서 매우 치명적인 오류이다. 확률 기반 모델은 1.21%p, 제안 모델은 1.45%p의 에러 감소율을 보였다.

[표8]은 규칙 기반 모델은 기존 에러율에 비해 18.55%p 에러 증가율을 보였고, 확률 기반 모델은 3.89%p, 제안 모델은 5.50%p의 에러 감소율을 보였다. 이는 확률 기반 모델보다 제안 모델의 에러율이 1.61%p 더 낮은 결과를 보였다. 이러한 실험 결과는 비정형 데이터로 어플리케이션 관점에서도 비교 모델들보다 높은 성능임을 확인 할 수 있다.

표 7. 오류 생성 전 모델 간 성능 비교

모 델	에러율(%)	증감율(%p)
규칙 기반 모델[1]	16.49	+10.37
확률 기반 모델[5]	4.91	-1.21
제안 모델	4.67	-1.45

* 교정 전 평가 데이터 에러율 : 6.12%

표 8. 오류 생성 후 모델 간 성능 비교

모 델	에러율(%)	증감율(%p)
규칙 기반 모델[1]	39.45	+18.55
확률 기반 모델[5]	17.01	-3.89
제안 모델	15.40	-5.50

* 교정 전 평가 데이터 에러율 : 20.90%

4.9 모델 별 평균 수행 시간 비교

[표 9]는 각 모델의 비정형화된 데이터 1000문장을 교정하는데 드는 평균 수행 시간이다. 제안 모델은 전체문맥 정보를 반영하여 관련 어절을 찾으므로, 비교 모델들보다 성능은 높으나 교정하는 데 드는 평균 수행 시간은 높다.

표 9. 모델 별 평균 수행 시간

모 델	평균 수행 시간(초)
규칙 기반 모델[1]	3.44
확률 기반 모델[5]	5.35
제안 모델	8.19

5. 결론 및 향후 연구

본 연구에서는 Self-Attention 기반으로, 문맥 의존적인 정보가 포함된 임베딩을 활용하여 문맥의존 철자오류 교정 모델을 제안하였다. 제안하는 모델을 검증하기 위해 정형화 데이터와 비정형화 데이터 모두 사용하였으며, 비교 모델보다 우수한 모델임을 입증하였다. 또한, 모델에서 Self-Attention을 제거하여 임베딩만 사용함으로써 Self-Attention의 기여를 확인하였다.

본 논문에서 Self-Attention의 기여를 확인하였으므로, 향후 연구로는 Multi-Haed-Attention을 활용하여 Self-Attention의 능력을 더 향상해 문맥 의존적인 정보를 더 풍부하게 활용하여 문맥의존 철자오류 교정을 수행할 것이다.

참고문헌

[1] 노강호, 박근수, 조환규, 장소원, “음소의 1차원 배열을 이용한 한글 유사도 및 편집거리 알고리즘”, 정보과학회논문지 : 컴퓨팅의 실제 및 레터 제17권 제10호, pp.519-526, 2011.

[2] 김성환, 조환규, “다중서열정렬을 이용한 변형 문자열 집합의 유사도 계산 기법”, 정보과학회논문지 : 소프트웨어 및 응용 제40권 제1호, pp.53-60, 2013.

[3] 옹윤지, 강승식, “터치스크린 환경에서 쿼티 자판 오타 교정을 위한 N-gram 언어 모델”, 스마트미디어저널, pp.54-59, 2018.

[4] Ryo Nagata, Hiroya Takamura, Graham Neubig, “Adaptive Spelling Error Correction Models for Learner English”, Procedia Computer Science, Vol.112, pp.474-483, 2017.

[5] 김민호, 권혁철, 최성기, “어절 N-gram을 이용한 문맥의존 철자오류 교정”, 정보과학회논문지 제41권 제12호, pp.1081-1089, 2014.

[6] 이정훈, 김민호, 권혁철, “문맥의존 철자오류 후보 생성을 위한 통계적 언어모형 개선”, 멀티미디어학회논문지 제20권 제2호, pp.371-381, 2017.

[7] 이정훈, 김민호, 권혁철, “통계적 문맥의존 철자오류 교정 기법의 향상을 위한 지역적 문서 정보의 활용”, 정보과학회 컴퓨팅의 실제 논문지 제23권 제7호, pp.446-451, 2017.

[8] 이정훈, 김민호, 권혁철, “워드임베딩을 이용한 문맥의존 철자오류 교정 기법”, 한국정보과학회 2018 한국컴퓨터종합학술대회 논문집, pp.607-609, 2018.