**KITE: 한국어 고의 오탈자를 활용한 텍스트 데이터 증강 방법론  
Text Data Augmentation Methods**

**using Korean Intentional Typographical Errors**

| **유승욱**  **Seunguk Yu**  소속기관  소속기관 영문명  e-mail address | **최주환**  **Juhwan Choi**  소속기관  소속기관 영문명  e-mail address | **서희재**  **Heejae Suh**  소속기관  소속기관 영문명  e-mail address | **진교훈**  **Kyohoon Jin**  소속기관  소속기관 영문명  e-mail address | **김영빈**  **Youngbin Kim**  소속기관  소속기관 영문명  e-mail address |
| --- | --- | --- | --- | --- |

**요약문**

최근 자연어 데이터를 더욱 효율적으로 활용하기 위해 다양한 텍스트 증강 기법이 제안되고 있다. 본 연구에서는 온라인 상의 댓글 등에서 나타나는 오탈자에 초점을 맞추어 한국어 고의 오탈자를 활용한 텍스트 데이터 증강 기법을 제안한다. 다른 언어와는 달리 한 글자 내에서 다양한 오탈자가 발생할 수 있는 한국어의 특징을 활용해 고의 오탈자를 활용한 텍스트 증강 규칙을 규정하였다. 본 논문에서 제안하는 기법을 통해 증강한 문장은 포함된 오탈자가 늘어남에 따라 기존 문장과의 코사인 유사도 차이가 커짐을 확인하였다. 제안한 방식으로 실험 결과 NSMC, Korean Hate Speech 데이터셋에서 평균 1.1%의 성능 향상을 이룰 수 있었으며, 기존에 제안된 방법과 비교했을 때 더욱 효율적임을 알 수 있었다.

**주제어**

딥러닝, 자연어처리, 데이터 증강

1. **서론**

최근 텍스트 데이터를 처리하는데 있어 가장 핵심적으로 사용되는 방법은 딥러닝이다. 자연어 처리에 활용되는 모델은 주어진 데이터의 특징을 학습하여 텍스트 분류와 같은 다양한 작업에 활용된다. 그렇기에 딥러닝 기반 모델의 성능을 결정하는 가장 중요한 요소는 데이터의 양과 질이라고 할 수 있다[1]. 그러나 정해진 맞춤법에 따라 정제된 데이터를 통해서만 학습된 모델은 학습이 완료된 이후 오탈자 등의 노이즈가 포함된 실제 데이터에 대해서는 기대한 성능을 온전히 발휘하지 못할 우려가 있다.

텍스트 데이터 중 특히 댓글을 활용한 데이터는 정해진 맞춤법에서 어긋나거나 오타를 포함하는 경우가 빈번하다. 이같은 온라인의 글은 비격식적인 상황에서 작성되었기에 오탈자가 포함되었더라도 보통은 이를 수정하지 않고 방치하는 경우가 대다수이다. 본 연구에서 온라인 상의 텍스트로 구축된 데이터셋을 살펴본 결과 다수의 경우에 오탈자가 포함되어 있음을 확인하였다. 본 연구에서는 이러한 현상에 주목하여 한국어 문장에 고의 오탈자를 포함시키는 방법을 규정하고 이에 기반한 텍스트 데이터 증강 기법을 제안한다.

데이터 증강이란 보유한 데이터의 양을 늘려 모델 학습에 도움을 주는 기법을 말한다. 이는 일반화 성능을 높임으로써 더욱 강건한 모델을 만드는데 목적을 둔다. Easy Data Augmentation (EDA)[2]은 특정 단어의 교체, 삽입, 순서 변경, 삭제를 통한 간단한 방법으로 텍스트 증강을 시도하였다. 한국어 텍스트를 다룬 연구[3]에서도 마찬가지로 EDA와 유사한 규칙 기반의 데이터 증강 기법을 사용했으며, 역번역을 통해 문장을 생성하고 입력 오타를 활용하는 방식을 추가하기도 하였다. 그러나 문장에서 특정 단어를 임의로 교체하는 등의 방법은 실제 데이터에서 찾아보기 힘든 경우로, 그 특성상 새로 생성된 문장의 의미가 기존 문장과 달라질 가능성을 가진다.

반면 한국어의 특성을 통해 생긴 오탈자는 기존 문장 속 단어 의미를 크게 해치지 않는다. 영어 단어에서는 각 글자가 단어 표현에 있어 일정 비중을 차지하기에 ‘dog’라는 단어에서 ‘fog’, ‘dot’과 같이 한 글자만 오탈자가 발생하여도 본래 의미를 보존하지 못한다. 반면 한국어 단어에서는 각 글자가 초성, 중성, 종성으로 구성되기에 ‘강아지’라는 단어에서 ‘겅아지’, ‘강아즈’와 같이 자소 단위로 오탈자가 발생하여도 본래 의미를 일부분 추측할 수 있다.

본 논문에서는 한국어 문장에 고의 오탈자를 포함시켜 텍스트 데이터를 증강하는 방법인 Text Data Augmentation using Korean Intentional Typographical Errors (KITE)를 제안한다. 상단의 예시와 같이 기존 문장 속 단어의 의미를 최대한 해치지 않는 선에서, 실제 데이터에 있음직한 텍스트를 만들어내는 것을 목표로 한다.

글자의 초성에서 오탈자가 발생한다면 이는 영어 단어에서의 예시와 같이 단순한 오탈자가 아니라 의도한 단어 의미를 훼손시킬 가능성이 높기에, 오탈자는 중성 혹은 종성의 위치에만 적용시키고 초성에서는 고려하지 않았다. 그 예로 ‘강아지’라는 단어에서 ‘장아지’, ‘강아기’와 같이 초성에 오탈자가 발생한 경우는 읽는 이가 원래 의미를 추측하기 어렵기에 즉시 수정하는 경우가 많다. 이렇듯 영어와 같은 다른 언어와는 달리 한 글자가 초성, 중성, 종성으로 구성되는 한국어로 오탈자를 만들 수 있는 경우는 수없이 다양하기에 고의 오탈자가 적용되는 방법을 상세히 규정하였다.

제안하는 방법을 통해 기존 데이터셋에서 한국어 고의 오탈자가 포함된 데이터로 증강하고, 이를 모델 학습에 활용하여 성능 향상이 가능한지 확인하였다. 앞서 코사인 유사도를 통해 언어 모델이 오탈자로 증강한 문장을 기존 문장과는 얼마나 다르게 받아들이는지 확인해 보았으며, 기존의 제안되었던 텍스트 증강 방법들과 KITE와의 성능을 비교하기 위한 모델 학습 및 검증을 진행하였다.

1. **고의 오탈자를 활용한 문장 생성 기법**

본 연구에서 데이터 증강 기법의 수단으로서 활용한 한국어 고의 오탈자 생성 규칙을 설명한다. 기존 문장에서 오탈자를 포함한 새로운 문장의 생성은 2.3에서 다루며, 2.1과 2.2에서는 오탈자 단어를 만들기 위한 과정을 설명하였다.

* 1. **오탈자 단어 생성 과정**

오탈자 단어 생성을 위해 한 단어 내에서 자소 자체를 바꾸는 경우 및 자소 순서를 바꾸는 경우로 구분하여 표 1에 예시와 함께 정리하였다. 두 경우 모두 두글자 이상의 단어에서만 오탈자를 생성하였는데, 보통 관형사나 감탄사로 사용되는 한글자의 단어에서 오타가 나는 경우는 극히 드물었기 때문이다.

표 1. 한국어 고의 오탈자를 활용한 단어 생성

|  | 사례 | 예시 |
| --- | --- | --- |
| 자소 자체를 바꾸는 경우 | 일반자음 혹은 일반모음의 변형 | 설마 → 섷마, 좋았다 → 좋았드 |
| 이중자음 혹은 이중모음의 탈락 | 많이 → 맣이, 희생 → 히생 |
| 쌍자음의 탈락 | 그랬다 → 그랫다 |
| 위의 사례들이 혼용된 경우 | 그랬다 → 그랳다 |
| 자소 순서를 바꾸는 경우 | 첫번째 글자는 중성까지, 두번째 글자는 종성까지 구성된 경우 | 지금 → 직므, 대중 → 댖우, 조용히 → 종요히 |
| 첫번째 글자는 종성까지, 두번째 글자는 중성까지 구성된 경우 | 같이 → 가티  둘이서 → 두리서,  좋네요 → 조헤요 |

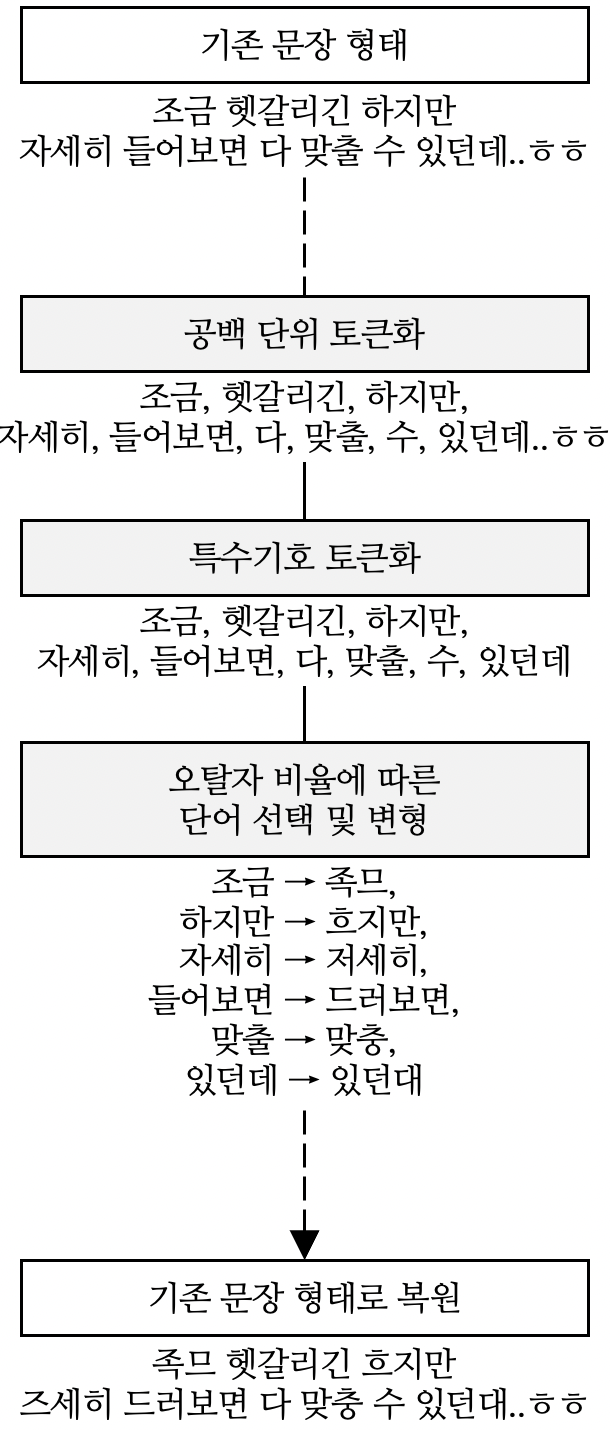
자소 자체를 바꾸는 경우는 키보드 상의 위치에서, 기존 자소와 한 칸 이내에 위치한 임의의 자소로 교체하는 방법이다. 일반자음 혹은 일반모음의 변형 사례 중에서는 ‘설마’라는 단어에서 ‘섷마’라는 단어로 오탈자를 만든 경우이며, 자음과 모음 총 26개의 자소에 대해 적용되도록 하였다. 이중자음 혹은 이중모음의 탈락 사례로는 ‘많이’, ‘희생’에서 두 자소 중 하나가 탈락되어 각각 ‘맣이’, ‘히생’와 같이 일반자음으로만 남는 형태로 적용되도록 하였다. 특히 ‘그랬다’과 같은 쌍자음의 경우는 ‘그랫다’와 같이 일반자음으로 탈락될 수 있으며, ‘그랳다’와 같이 다른 일반자음으로 변형이 일어날 수도 있다.

자소 순서를 바꾸는 경우는 한 단어 안에서 특정 글자의 초성, 중성, 종성 속 특정 자소를 다른 글자의 초성, 중성, 종성 속 특정 자소와 교체하는 방법이다. 입력된 단어의 첫번째 글자와 두번째 글자가 각각 어떻게 구성되었는지 파악하여 정해진 규칙에 맞게 서로의 자소를 바꿔주었다. 그 예로 ‘지금’이라는 단어에서 ‘직므’이라는 단어로 오탈자를 만들거나, ‘둘이서’라는 단어에서 ‘두리서’라는 단어로 오탈자를 만든 경우가 이에 속한다.

* 1. **오탈자 문장 생성 과정**

한국어 고의 오탈자를 활용한 텍스트 데이터 증강을 위해 기존 문장을 몇배의 비율로 증강할 것인지 뜻하는 ‘증강 비율’, 문장 안에서 오탈자의 비율을 얼마로 설정할 것인지 뜻하는 ‘오탈자 비율’을 사용자가 임의로 설정할 수 있다. 이같이 각 비율을 설정하였을 때 입력된 문장에 대해서 그림 1의 과정으로 데이터 증강이 이루어진다.

그림 1. 한국어 고의 오탈자를 활용한 문장 생성



위의 예시는 오탈자 비율을 80%로 설정하였기에 문장 내 대부분의 단어에서 오탈자가 생성되었다. 자소 순서를 바꾸는 경우가 적용되어 ‘조금’이라는 단어에서 ‘족므’라는 단어로 바뀌고, 자소 자체를 바꾸는 경우가 적용되어 ‘있던데’라는 단어에서 ‘있던대’라는 단어로 바뀐 결과를 확인할 수 있다.

1. **실험 설계 및 결과**

앞서 제안한 한국어 고의 오탈자를 활용한 텍스트 데이터 증강 기법의 성능을 검증하고자 설계한 실험 과정과 그 결과를 제시한다. 먼저 고의 오탈자로 증강한 문장이 기존의 문장과 얼마나 다른지 비교하고자 임베딩 상에서의 코사인 유사도 차이를 확인해보았으며, 이후 사전 학습된 언어 모델을 텍스트 분류 작업에 Fine-Tuning하는 과정에 KITE로 증강한 데이터를 사용함으로써 증강 전후의 성능 차이를 비교하였다.

* 1. **실험 데이터셋**

텍스트 분류 데이터셋으로는 Naver Sentiment Movie Corpus (NSMC)[4]와 Korean Hate Speech[5] 데이터셋을 사용하였다. NSMC 데이터셋은 네이버 영화 리뷰에 기반한 데이터셋으로 영화를 관람한 사람들의 반응을 긍, 부정으로 이진 분류한 150,000개의 학습용 데이터로 구성되어 있다. Korean Hate Speech 데이터셋은 인터넷 댓글의 공격성을 3단계로 분류한 데이터셋으로 약 8,000개의 학습용 데이터로 구성된다.

* 1. **오탈자 비율에 따른 텍스트 증강**

본 논문에서 제안하는 KITE를 활용하여 텍스트를 증강하기 위해서는 문장 안에서 오탈자의 비율을 얼마로 둘 것인지 뜻하는 ‘오탈자 비율’을 설정해야 한다. 이때 각 비율에 따라 증강한 문장이 기존 문장과 얼마나 다른지 비교하고자 KR-SBERT[6]를 이용하여 임베딩 상에서의 문장 사이 코사인 유사도를 측정하였다.

표 2. KITE 적용 후 코사인 유사도 차이 예시

| 기존 문장 | 송중기 시대극은 믿고본다. 첫회 신선하고 좋았다. |
| --- | --- |
| 증강 문장 예시 1 | 송중기 시댜극은 미도본다. 처쇠 신선허고 좋았드. (0.735, -0.265) |
| 증강 문장 예시 2 | 솣중기 세대극은 믿고본다. 처쇠 신선하규 좋았다. (0.580, -0.42) |

위의 예시는 오탈자 비율이 80%일 때 증강한 문장이다. 따라서 두 문장 모두 오탈자의 발생 개수는 동일하지만, 오탈자가 발생한 위치에 따라 기존 문장과의 코사인 유사도가 각각 0.735, 0.580으로 차이나는 모습을 보인다. 표 2에서 데이터셋 전체로 비교 대상을 늘려 증강 전후의 코사인 유사도를 비교할 때는 Korean Hate Speech의 테스트 데이터셋을 사용하였고, 증강 비율을 8배로 두어 많은 양의 증강이 이루어진 상황을 만들었다. 이후 기존 문장과 증강한 문장들 사이 코사인 유사도를 구하여 전체 문장에 대해 평균값을 구하였다.

표 3. KITE에서 오탈자 비율에 따른 코사인 유사도 차이

| 오탈자 비율 | 증강 전후 문장 간 코사인 유사도 |
| --- | --- |
| 20% | 0.940, -0.06 |
| 40% | 0.866, -0.134 |
| 60% | 0.801, -0.199 |
| 80% | 0.725, -0.275 |

그 결과 오탈자 비율이 증가함에 따라 증강한 문장과 기존 문장과의 코사인 유사도가 차이가 커지는 현상을 확인했다. 문장에 많은 오탈자를 추가할수록 기존 문장과 임베딩 상 거리가 멀어지는 것을 의미하며, 이후 활용할 언어 모델이 증강 전후의 문장을 서로 다르게 받아들임을 의미한다. 이는 한국어 자소 단위로 오탈자를 만들면 사람 시각에서는 본래의 의미를 일정 부분 유추할 수 있지만, 언어 모델의 시각에서는 다른 의미를 가진 문장이 된다는 사실을 시사한다.

* 1. **증강 방법에 따른 성능 비교**

본 논문에서 제안하는 KITE의 성능을 검증하고자 기존에 제안된 텍스트 증강 방법들과 비교하는 실험을 진행하였다. 사용된 기존 텍스트 증강 방법들 중 Back-Translation (BT)[7]은 가진 텍스트를 외국어로 번역한 뒤 다시 원래의 언어로 재번역함으로써 텍스트를 증강한다. EDA는 특정 단어의 교체, 삽입, 순서 변경, 삭제를 활용하는 간단한 방식의 텍스트 증강 방법이다.

위의 방법으로 증강한 데이터셋을 한국어로 사전 학습된 KoBERT[8]의 학습 데이터로 이용하여 텍스트 분류를 위한 Fine-Tuning을 진행하였다. 모델을 학습시키기 위한 Optimizer로는 AdamW를 사용하였다. Batch의 크기는 16으로, Learning Rate는 3e-6으로 지정하고 Cosine Annealing을 사용하여 20 Epoch 동안 학습을 진행하되 손실값이 4 Epoch 동안 개선되지 않는 경우 Early Stopping을 주었다. 실험 결과는 표 3으로 그 중 Original은 아무런 증강을 거치지 않은 데이터로 실험한 성능을 뜻한다.

표 4. 증강한 데이터셋을 통한 텍스트 분류 성능 비교

|  | NSMC | Korean  Hate Speech | Average |
| --- | --- | --- | --- |
| Original | 88% | 52.8% | 70.4% |
| BT | 89.3% (+1.3%p) | 53.4% (+0.6%p) | 71,35% (+0.95%p) |
| EDA | 88.3% (+0.3%p) | 50.6% (-2.2%p) | 69.45% (-0.95%p) |
| KITE | 88.7% (+0.7%p) | 54.3% (+1.5%p) | 71.5% (+1.1%p) |

사용한 모든 방법에서 증강 비율은 항상 2배로 두었으며, KITE에서 오탈자 비율은 NSMC 데이터셋과 Korean Hate Speech 데이터셋에서 모두 60%으로 설정하였다. 실험 결과 각 데이터 증강 방법들을 사용하였을 때 EDA를 제외하고는 모두 Original보다 성능 향상을 보였으며, 특히 본 논문에서 제안하는 KITE가 각 데이터셋에서 기존과 +0.7%p, +1.5%p으로 평균 +1.1%p의 차이를 보였다.

특정 단어의 교체 및 삽입 등 간단한 방식을 사용하는 EDA의 경우 Korean Hate Speech 데이터셋에서 비교적 큰 성능 하락폭을 보였다. 텍스트의 재번역을 통해 증강하는 BT의 경우 평균 성능의 차이는 KITE보다 약간 낮으나, 번역에 사용하는 모델에 따라서 증강 소요 시간 및 성능이 일정하지 않으며 다른 증강 방법과 달리 GPU 연산이 요구된다는 단점이 있다. 그에 반해 KITE를 통한 증강은 별도의 딥러닝 모델을 사용하지 않으며 증강 시간도 각 데이터당 5분 이내로 신속하게 수행되었다.

1. **결론**

본 논문에서는 사람의 실수로 인해 한국어 문장에서 나타나는 오탈자 종류를 상세히 규정하고 이를 활용한 텍스트 데이터 증강 기법인 KITE를 제안하였다. 이는 별도의 딥러닝 모델을 사용하지 않는 간단한 규칙 기반의 데이터 증강 기법이면서도, 기존 연구에서 주로 사용되었던 특정 단어 교체 및 삽입 등의 방식으로 문장의 본래 의미를 훼손시키지 않는 텍스트 증강 방식이다. 이렇듯 한국어 문장에서의 오탈자가 각 글자의 자소 단위로 발생한다는 점에 기반하여 KITE를 통한 텍스트 증강 방법을 규정하였다.

실험에서는 증강 비율 및 오탈자 비율을 설정함에 따라 기존의 데이터셋을 여러 형태로 증강하였다. 그 결과 증강한 문장은 포함된 오탈자가 늘어남에 따라 기존 문장과 코사인 유사도 차이가 커짐을 보이며 언어 모델이 증강 전후의 문장들을 각각 다르게 받아들임을 확인할 수 있었다. 또한 텍스트 분류 작업으로 Fine-Tuning하는 과정에서 KITE로 증강한 데이터셋을 학습시켰을 때 NSMC 데이터셋과 Korean Hate Speech 데이터셋 모두에서 기존에 제안된 텍스트 증강 방법들보다 높은 성능 향상 결과를 확인하였다.

본 연구는 실제 데이터에 있을법한 형태로 텍스트 증강을 이루어낸 시도로, 그 과정에서 한국어의 특징을 반영함으로써 증강 과정의 설득력을 확보하였다. 온라인 상에서 일정 양의 한국어 텍스트 데이터를 확보하기 어려울 때 KITE를 활용하여 오탈자가 포함된 다양한 형태의 한국어 텍스트를 얻을 수 있기를 기대한다.

**사사의 글**

이 연구는 “A 사업”의 지원을 받아서 수행되었다.

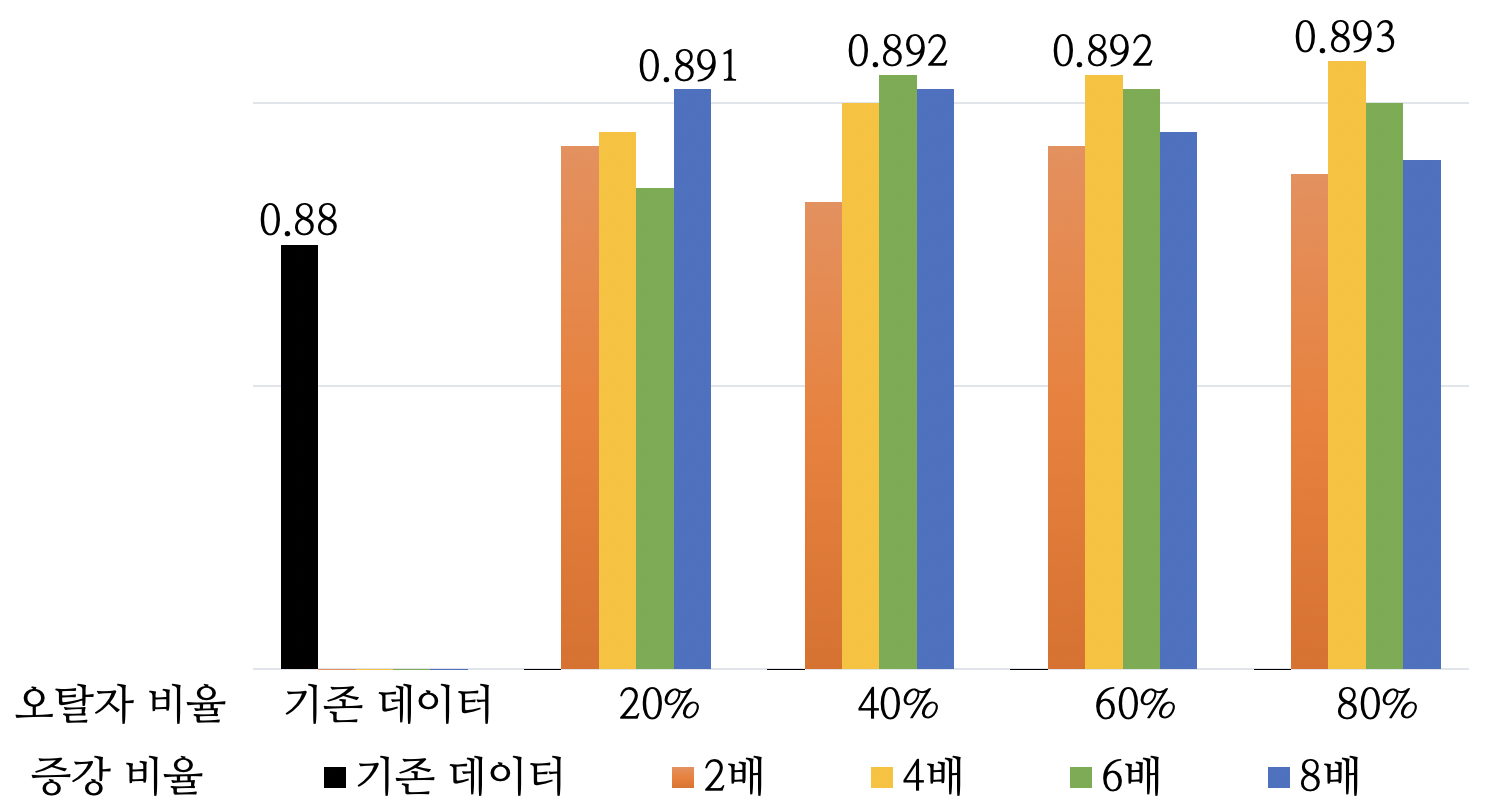
**참고 문헌**

1. Luca, A. R., Ursulenau, T. F., Gheorghe, L., Grigorovici, R., Iancu, S., Hlusneac, M. and Grigorovici, A. Impact of quality, type and volume of data used by deep learning models in the analysis of medical images. Informatics in Medicine Unlocked. 29. Elsevier. 100911. 2022.
2. Wei, J. and Zou, K. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 6383-6389. 2019.
3. 조진욱, 정민수, 이정훈, 정윤경. 한국어 텍스트 데이터를 위한 변형적 데이터 증강 방법론. 한국정보과학회 2020 한국소프트웨어종합학술대회 논문집. 한국정보과학회. pp. 592-594. 2020.
4. NSMC, Accessed: Nov. 2022. [Online]. Available: https://github.com/e9t/nsmc
5. Moon, J., Cho, W. I., and Lee, J. BEEP! Korean Corpus of Online News Comments for Toxic Speech Detection. In Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media. pp. 25-31. 2020.
6. KR-SBERT, Accessed: Nov. 2022. [Online]. Available: https://github.com/snunlp/KR-SBERT
7. Rico, S., Barry, H. and Alexandra, B. Improving Neural Machine Translation Models with Monolingual Data. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 86-96. 2016.
8. KoBERT, Accessed: Nov. 2022. [Online]. Available: https://github.com/SKTBrain/KoBERT

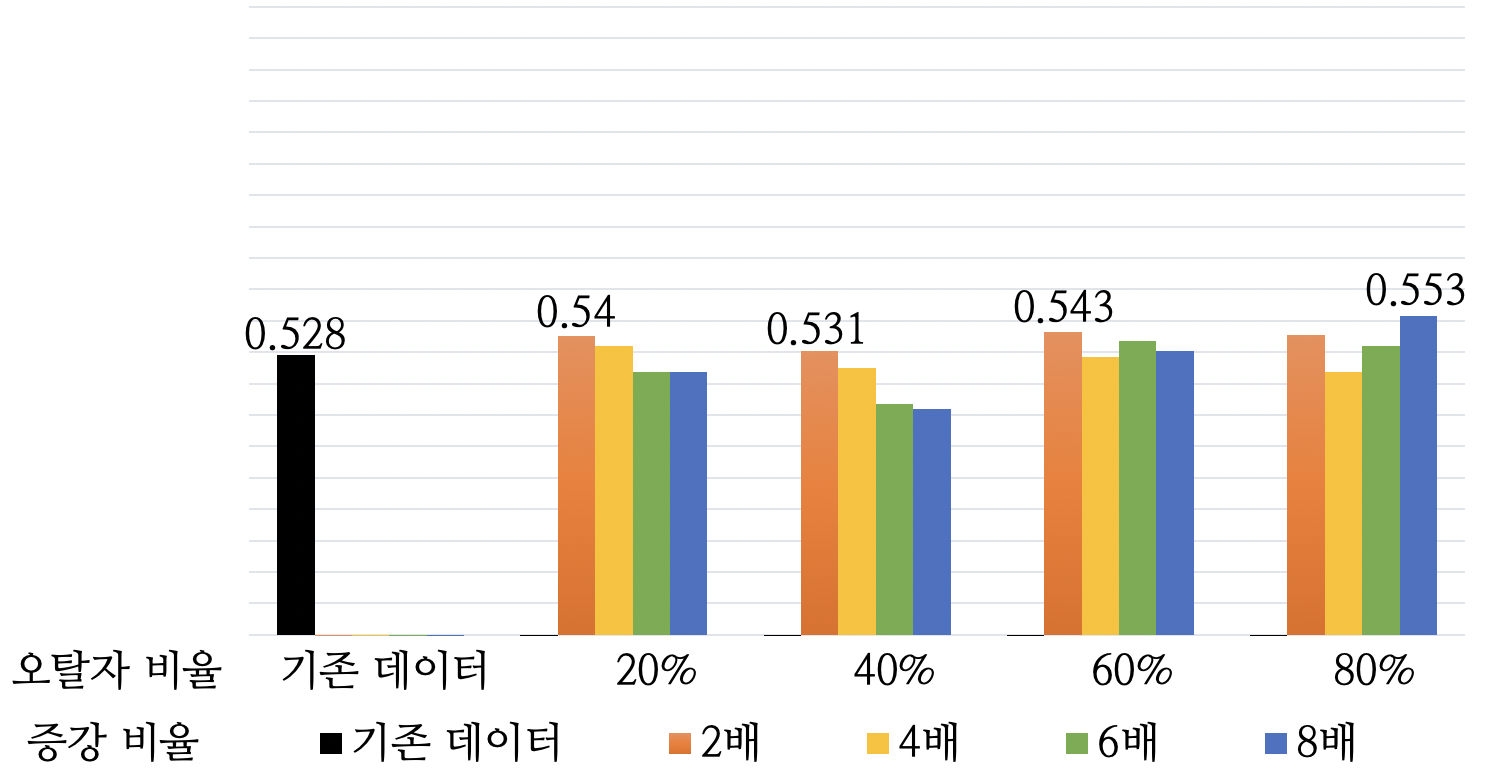
| nsmc | X | 2 | 4 | 6 | 8 |
| --- | --- | --- | --- | --- | --- |
| X | .906/.880(6) |  |  |  |  |
| 0.2 |  | .928/.887(5) | .922/.888(4) | .992/.884(17) | .994/.891(12) |
| 0.4 |  | .939/.883(11) | .938/.890(4) | .927/.892(4) | .996/.891(20) |
| 0.6 |  | .916/.887(7) | .920/.892(5) | .939/.891(4) | .958/.888(7) |
| 0.8 |  | .909/.885(7) | .917/.893(4) | .936/.890(6) | .905/.886(4) |

| hate speech | X | 2 | 4 | 6 | 8 |
| --- | --- | --- | --- | --- | --- |
| X | 52889 |  |  |  |  |
| 0.2 |  | 54037 | 53425 | 5172 | 51771 |
| 0.4 |  | 53154 | 52056 | 49791 | 49403 |
| 0.6 |  | 54394 | 52737 | 53768 | 5315 |
| 0.8 |  | 54157 | 51783 | 53473 | 55355 |

본 논문에서 제안하는 한국어 고의 오탈자를 활용하여 NSMC, Korean Hate Speech 데이터셋에 텍스트 증강을 진행한 후 학습했을 때 모델의 테스트 정확도를 비교하였다.



NSMC 데이터셋에서의 실험 결과 한국어 고의 오탈자를 적용하지 않은 기존 데이터셋에서는 0.88의 정확도를 보였으나, 오탈자 비율과 증강 비율을 달리한 16가지 버전의 모든 증강된 데이터셋에서 정확도가 향상된 모습을 보였다. 특히 오탈자 비율이 20%로 가장 낮은 상황일 때는 증강 비율을 최대인 8배로 설정했을 때, 기존 데이터셋보다 0.011의 정확도 향상을 보였다. 오탈자 비율이 40~80%로 점점 높아짐에 따라 증강 비율을 최대로 맞추는 것보다 4~6배로 적절하게 설정했을 때 각각 0.012, 0.013의 정확도 향상을 보였다.



Korean Hate Speech 데이터셋에서의 실험 결과 한국어 고의 오탈자를 적용하지 않은 기존 데이터셋에서는 0.528의 정확도를 보였다. 오탈자 비율과 증강 비율을 달리한 16가지 버전의 증강된 데이터셋 중 기존 데이터셋보다 정확도가 일부 낮은 경우가 있었으나, 증강된 데이터셋으로 인해 가장 큰 폭으로 정확도가 오른 경우는 0.025로 NSMC 데이터셋에서의 결과보다 큰 폭의 향상을 보였다. 특히 오탈자 비율이 80%로 가장 높은 상황일 때는 증강 비율을 최대인 8배로 설정했을 때, 기존 데이터셋보다 0.025의 정확도 향상을 보였다. 오탈자 비율을 20%~60%로 설정했을 때에서는 모두 증강 비율을 2배로 두는 세팅이 각각 0.012, 0.003, 0.015의 정확도 향상을 보였다.

본 논문에서 제안하는 한국어 고의 오탈자를 활용하여 NSMC, Korean Hate Speech 데이터셋에 텍스트 증강을 진행한 후 KoBERT 모델에 Fine-Tuning하였을 때 모델의 테스트 정확도를 비교하였다.

표 4. 한국어 고의 오탈자로 증강한  
데이터셋에서의 Fine-Tuning 텍스트 분류 성능

|  | NSMC | Korean  Hate Speech |
| --- | --- | --- |
| 기존 데이터셋 | 88% | 52.8% |
| 오탈자 비율 20% | 증강 비율 8배,  89.1% (+1.1%p) | 증강 비율 2배,  54% (+1.2%p) |
| 오탈자 비율 40% | 증강 비율 6배,  89.2% (+1.2%p) | 증강 비율 2배,  53.1% (+0.03%p) |
| 오탈자 비율 60% | 증강 비율 4배,  89.2% (+1.2%p) | 증강 비율 2배,  54.3% (+1.5%p) |
| 오탈자 비율 80% | 증강 비율 4배,  89.3% (+1.3%p) | 증강 비율 8배,  55.3% (+2.5%p) |

NSMC 데이터셋에서의 실험 결과 한국어 고의 오탈자를 적용하지 않은 기존 데이터셋에서는 0.88의 정확도를 보였으나, 오탈자 비율과 증강 비율을 달리한 16가지 버전의 모든 증강된 데이터셋에서 정확도가 향상된 모습을 보였다. 특히 오탈자 비율이 20%로 가장 낮은 상황일 때는 증강 비율을 최대인 8배로 설정했을 때, 기존 데이터셋보다 0.011의 정확도 향상을 보였다. 오탈자 비율이 40~80%로 점점 높아짐에 따라 증강 비율을 최대로 맞추는 것보다 4~6배로 적절하게 설정했을 때 각각 0.012, 0.013의 정확도 향상을 보였다.

Korean Hate Speech 데이터셋에서의 실험 결과 한국어 고의 오탈자를 적용하지 않은 기존 데이터셋에서는 0.528의 정확도를 보였다. 특히 오탈자 비율이 80%로 가장 높은 상황일 때는 증강 비율도 최대인 8배로 설정했을 때, 기존 데이터셋보다 0.025의 정확도 향상을 보였다. 오탈자 비율을 20%~60%로 설정할 때에는 모두 증강 비율을 2배로 두는 세팅이 각각 0.012, 0.003, 0.015의 정확도 향상을 보였다.

두 데이터셋에서 오탈자 비율과 증강 비율을 적절히 설정함에 따라 Fine-Tuning된 모델의 성능 향상을 확인할 수 있었다. NSMC 데이터셋은 모든 오탈자 비율에서 1%p대의 일정한 성능 향상을 보장했지만, Korean Hate Speech 데이터셋은 특정 오탈자 비율에서 성능 향상 폭이 조금 낮거나, 성능 향상 폭이 큰 경우를 보였다. \*\*이는 각 학습 데이터셋의 증강 전 사이즈가 각각 15만 개, 8천 개로 상이했기에 비교적 큰 데이터인 NSMC에서 보장된 성능이 관찰으로 해석된다. 또한 NSMC는 영화 리뷰와 관련한 데이터로 텍스트를 이진분류하지만 Korean Hate Speech는 텍스트 레이블이 3개인 다중분류 작업이었기에 더 어려운 난이도로 증강된 데이터의 영향을 덜 받은 것으로 보인다.

\*\*텍스트 상 오탈자는 본래 사람의 실수에 의해 자연스럽게 발생하지만, 언어 모델이 데이터셋에 포함된 일부 오탈자에 대해서만 편향적으로 학습할 가능성이 존재한다. 본 연구에서는 이러한 현상에 주목하여 한국어 문장에 고의 오탈자를 포함시키는 방법을 규정하고 이에 기반한 텍스트 데이터 증강 기법을 제안한다.

한국어 오탈자를 활용한 Data Augmentation

Target - HCI Korea 2023: 11월 21일 자정까지 제출

논문 양식: [논문요약본 국문 Form download](https://conference.hcikorea.org/pds/2023/HCI2023_paperForm.doc) 참고

# 구현(코딩)

1. Rule-based 한국어 오탈자 Augmentation

* 담당: 유승욱

1. 한국어 Classification

* kobert 사용
* NSMC, Korean Hate Speech에 적용
* 담당: 서희재

1. 오타/맞춤법 차이로 인한 Model Output 차이 표현

* Visualization / Cosine Similarity
* 담당: 서희재

# 논문 Writing 흐름

1. 잘 정제되지 않고 오타가 있는 데이터에 대해서 Task를 진행해야 할 때도 있다.  
   실제로 NSMC나 Korean Hate Speech 등의 Dataset에는 오타 사례가 있음.
   1. 오타 정의
2. 우리가 정의한 오타에 따라서 오탈자가 포함된 문장을 생성하는 Rule을 규정

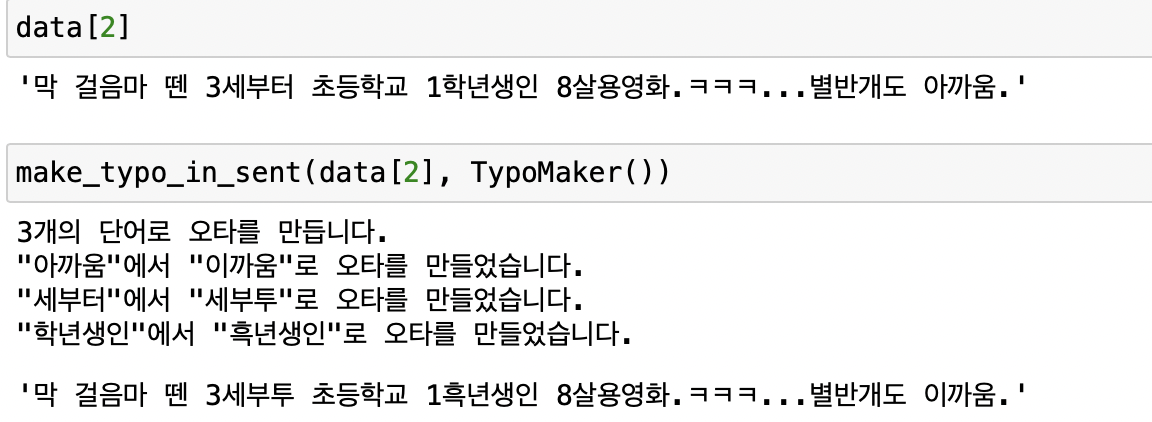
고민해야 할 사항)  
 **한 문장 안에서 얼마나 오타를 낼 것인가 (25% / 50% / 75%)  
 기존 한 문장에서 오타낸 문장을 얼마나 만들어낼 것인가 (2배 / 5배 / 10배)**

**한 단어 안에서 얼마나 오타를 낼 것인가**

* 1. 오탈자 규칙
     1. 키보드에서 비슷한 위치 (**한 칸** / 두 칸 이내)  
        **구현하다가 든 생각으로 한 칸 이내의 자소로 바꾸되,  
        키보드 상에서 같은 라인이거나 아래 라인으로만 바꿀까?  
        ex) ‘ㅇ’에 대한 오타의 경우 한 칸 이내의 자소라면 ‘ㄷ’, ‘ㄴ’, ‘ㄹ’, ‘ㅊ’ 인데 여기서 키보드 상 위 라인에 위치하는 ‘ㄷ’은 제외할까라는 생각.  
        보통 오타를 낼때 키보드 상 위 라인으로 오타를 내지는 않는 것 같아서..**
        + 자리 잡을 **듯** → 자리 잡을**둣, 원**래 익숙한 → **월**래 익숙한, 예**뻐**지고 → 예**뼈**지고, 못된 **놈**들 → 못된 **넘**들, 별**로**더라 → 별**루**더라
     2. 쌍자음 ㄱ ↔ ㄲ, ㄷ ↔ ㄸ  
        쌍자음에서 단일 자음으로의 오타는 있지만, 그 반대로의 오타는 없음
        + 충격적이**었**다 → 충격적이**엇**다, 재**밌**네요 → 재**밋**네요
     3. 1)과 2)의 혼용 케이스
        + 재**밌**당 → 재**밓**당
     4. 풀어쓰기 (실제 예시 발견 못함)
        + 원해 → 우너해
     5. **자음 혹은 모음 탈락 (반영됨)**
        + **관해서 → 과내서, 뭐가 → 머가**
     6. **받침으로 들어가는 경우 (반영됨)**
        + **도적 → 돚거**
  2. 오탈자 문장 생성시 가정
     1. 오타는 **2**글자 이상 단어에서만 남
     2. 자음은 자음끼리, 모음은 모음끼리만 오타남
  3. 오탈자 문장 생성 과정
     1. 문장을 공백 토큰화 -> 그중 한글 단어로만 분리 (특수기호, 영어 제거)  
        **댓글 데이터다보니 기대하는 대로 띄어쓰기, 맞춤법 등이 지켜진게 아님**  
        ex) 막 걸음마 뗀 3세부터 초등학교 1학년생인 8살용영화.ㅋㅋㅋ...별반개도 아까움.  
          
        따라서 우선은 공백을 기준으로 한 단어라 가정하였으며,  
        ex) ['막', '걸음마', '뗀', '3세부터', '초등학교', '1학년생인', '8살용영화.ㅋㅋㅋ...별반개도', '아까움.']

그 사이에 특수기호나 영어 등이 있을때 한번더 분리함.  
ex) ['막', '걸음마', '뗀', '세부터', '초등학교', '학년생인', '살용영화', '별반개도', '아까움']

* + 1. 위 단어 중에서, 문장을 얼마나 망칠 것인가에 따라  
       2글자 이상의 단어들을 랜덤하게 선택함
    2. 위의 조건에 따라 한 단어에서  
        - 오타 낼 한 글자 랜덤하게 선택  
        ex) ‘목소리’ 중 ‘목’  
        - 해당 글자에서 중성/종성 중 랜덤하게 선택  
        ex) ‘목’ 중 ‘ㅗ’를 오타낼 것임  
        - 선택된 자소 중 키보드에서 가까운 자소로 변경  
        ex) ‘ㅗ’를 ‘ㅓ’로 변경, ‘목소리’ -> ‘먹서리’  
        - 초성은 오타 사례가 없기에 대상으로 선택하지 않음  
         
       아무런 인자도 주지 않았을 때  
       (문장에서 1글자 이상인 단어들 중 **절반**을 오타화,  
       한 단어에서 오타 **한번** 냄)



인자 주었을 때  
 (문장에서 1글자 이상인 단어들 중 **80%**를 오타화,  
 한 단어에서 오타 **두번** 냄)



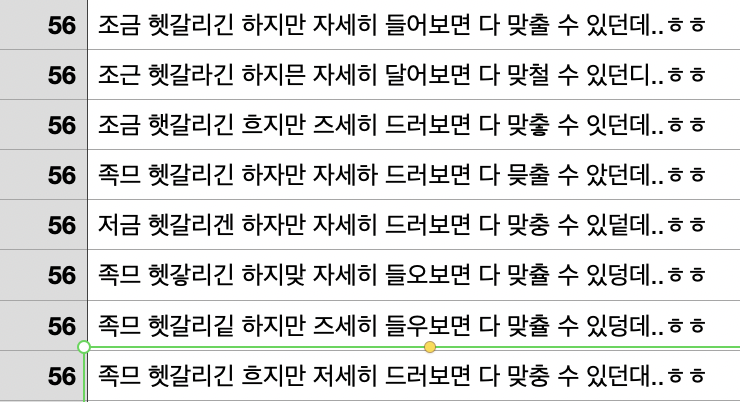
* + 1. 참고: 자/모음 영문 발음  
       [한글 자음과 모음의 영문발음 ㄱ 기역... - Learn and Study Korean](https://www.facebook.com/833865823409139/posts/915471185248602/)

1. 오탈자 Augmentation 진행, 예시를 보여준다
2. 이렇게 오탈자를 포함해 생성시킨 문장은 원래 문장과 Output 차이가 있음을 보여준다 -> 모델이 인지하고 내놓는 Output 차이가 큰 만큼 새로운 데이터로써 유용하게 사용할 수 있음

* Visualization 으로 할것인가 Cosine Similarity 적용할것인가는 결과물 보고 상황에 따라…

1. Augmentation된 데이터를 추가하기 전 후의 성능 비교

밑의 논문 예시(2.1, 2.3)는 모두 dev.hate\_8\_0.8\_default.csv에서 가져옴  
2.1의 경우는 앞부분에 다 있고, 2.3의 경우는 id 56번 글



typo\_aug train 방법

train\_data, test\_data와 같은 디렉토리, 혹은 하위 디렉토리를 두고 코드 위치 기준으로 path 설정

ex)

python main.py --data\_type 'nsmc' --batch\_size 64 --train\_path "/ratings\_train\_6\_0.2\_default.tsv" --test\_path "/ratings\_test.txt" --sentence\_column "document" --sep "\t" --device\_name "cuda:3"

Hyper Parameter

Batch\_size = 16/8

Learning\_rate = 3e-6

Warmup\_ratio = 0.1

max\_length = 64

max\_grad\_norm = 5

num\_epochs(max) = 20

optimizer = AdamW

loss\_fn = nn.CrossEntropyLoss

scheduler = get\_cosine\_schdeule\_with\_warmup

dropout\_rate = 0.5

실험 결과, 각 칸에 train/test accuracy(epoch)

early\_stop\_patience = 4

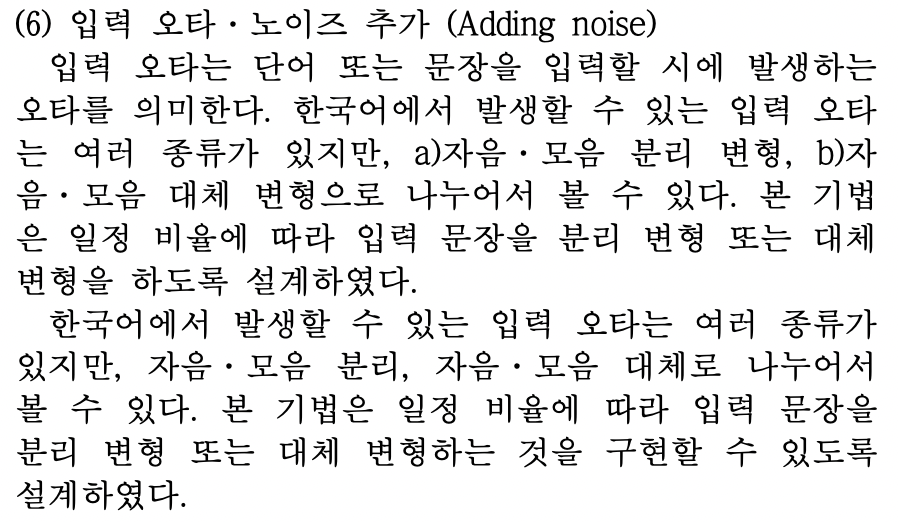
| nsmc | X | 2 | 4 | 6 | 8 |
| --- | --- | --- | --- | --- | --- |
| X | .906/.880(6) |  |  |  |  |
| 0.2 |  | .928/.887(5) | .922/.888(4) | .992/.884(17) | .994/.891(12) |
| 0.4 |  | .939/.883(11) | .938/.890(4) | .927/.892(4) | .996/.891(20) |
| 0.6 |  | .916/.887(7) | .920/.892(5) | .939/.891(4) | .958/.888(7) |
| 0.8 |  | .909/.885(7) | .917/.893(4) | .936/.890(6) | .905/.886(4) |

| hate speech | X | 2 | 4 | 6 | 8 |
| --- | --- | --- | --- | --- | --- |
| X | 52889 |  |  |  |  |
| 0.2 |  | 54037 | 53425 | 5172 | 51771 |
| 0.4 |  | 53154 | 52056 | 49791 | 49403 |
| 0.6 |  | 54394 | 52737 | 53768 | 5315 |
| 0.8 |  | 54157 | 51783 | 53473 | 55355 |

\*기존 연구 사례 정리

# **[1] 한국어 텍스트 데이터를 위한 변형적 데이터 증강 방법론 한국정보과학회(KSC), 2020. 12., 조진욱 외**

* 서론: 텍스트 증강 기법은 변형적(transformational) / 생성적(generative) 증강 기법으로 나뉘며, 본 연구는 지도학습 데이터에 적합한 변형적 텍스트 데이터 증강 방법론을 제안함
* 서론에서는 보다 일반적인 데이터에서 원초적인 증강 기법을 연구한다는 느낌이지만,  
  우리가 작성한다면 ‘댓글’ 데이터라는 것, 현실에 좀더 있을법한 데이터로 증강한다는 어필.
* 방법론: 두 토큰 순서를 임의 변경, 임의 토큰 추가, 임의 토큰 지우기, 특정 토큰을 동의어로 변경, 역변역을 통한 문장 생성, 입력 오타 노이즈 추가  
  이중에서도 ‘입력 오타 노이즈 추가’ 부분에서 우리와 유사하지만, 설명이 간략하게만 서술됨,  
  우리가 작성한다면 한국어 오타가 발생할 수 있는 경우의 수를 디테일하게 고려했다는 어필.  
  문장을 얼마나 망쳤는지와 같은 비율을 명시하지 않았기에, 우리와 같이 파라미터 설정을 통해 문장을 몇배로 늘릴지, 문장을 얼마나 망칠지, 한 단어를 얼마나 망칠지 고려하였다는 어필.



* 실험: NSMC 데이터만 사용, 모델로는 TextCNN, BERT(multi-lingual), KoElectra  
  방법론에서 제시한 모든 기법(총 6가지)를 사용했을 때 2.6~5.5%의 성능 향상을 보임,  
  그중에서도 ‘임의 토큰 추가’가 가장 의미있는 성능 향상을 보였다고 함  
  변형적 데이터 증강기법이 데이터 분포에 존재하는 편향을 최소화하였다고 주장

# **[2] 단어 계층 기반 텍스트 데이터 증강 방법론 한국컴퓨터정보학회(JKSCI), 2021. 01., 김무성 외**

* 서론: text to image 태스크에서 동일한 이미지더라도 텍스트로 설명할 수 있는 방법이 문맥에 따라 다양하기에 텍스트 증강을 진행함. 기존엔 유의어를 사용하는 어휘 대체 기반의 증강 기법을 사용했지만, 이는 텍스트 대체 과정에서 품사에 따라 각 단어의 계층적 관계를 고려하지 못한다고 주장. (text to image 태스크에서 유의어로 텍스트 대체에 따라 증강된 텍스트가 원본 이미지의 의미를 왜곡할 수 있음)  
  시소러스 기반 데이터 증강은 워드넷에서 트리 구조로 정의된 유의어 관계를 사용함. 따라서 상당한 시간과 비용 소모되며 OOV 문제 초래.  
  임베딩 기반 데이터 증강은 사전학습된 word2vec, fasttext, glove에서 단어 벡터와 유사한 벡터를 찾음.  
  이렇듯 기존 증강법은 모두 단어 유의한 관계만 기반하며, 단어 계층적 관계를 고려하지 못함.  
  이 연구는 너무 text to image 태스크 focused라서 언급하기 적절할 지는 모르겠음.
* 방법론: 문장에서 n개 단어들을 선택, 품사 파악, 워드넷을 활용하여 계층적으로 단어 대체  
  (‘계층적’의 의미가 유의어인 단어를 잡음과 동시에 그 단어를 포함하는 의미의 단어를 표현한다고 추정, ex. man → person, surfboard holding → board keeping)
* 실험: 증강한 데이터로 AttnGAN(text to image model) 사용 후 inception score 확인

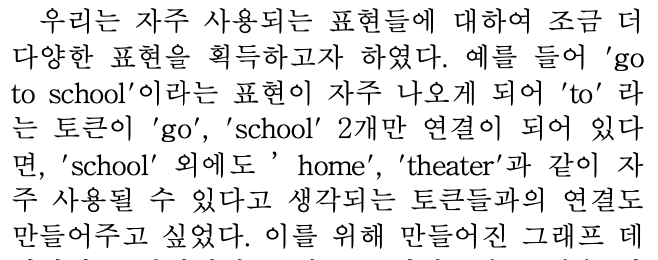
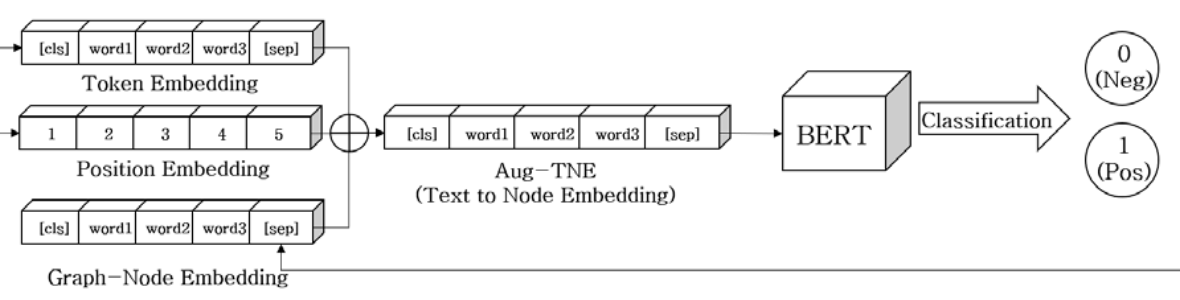
# **[3] 정도 부사를 이용한 텍스트 분류 작업에서의 데이터 증강에 관한 연구 한국통신학회, 2022., 김지환 외**

* 서론: 임의로 단어를 대체하는 등의 방법 말고, 기존 문장 의미를 크게 해치지 않는 정도 부사를 활용하면 더 나은 성능을 보일 것이라 주장.
* 실험: KLUE 중 뉴스 헤드라인 분류 데이터셋에서 진행. 정도 부사 [아주, 매우, 몹시, 무척, 심히, 유난히, 꽤, 상당히] 사용. 모델로 GRU 사용.  
  비교를 위해 EDA 증강 활용. “EDA의 경우 한국어 어휘 의미망의 성능 한계로 지시 관형사가 있는 경우 의미 없이 비슷한 관형사가 추가되는 식의 데이터 학습의 방해될 수 있는 데이터가 생성되는 경우를 발견하였고, 이런 문제로 인하여 성능 저하를 보인 것으로 사료된다.”고 언급.
* 증강한 데이터의 은닉층 출력값을 t-sne로 시각화하여 보여주어, 기존 데이터의 위치와 비교함.

# **[4] 기계 독해 성능 개선을 위한 데이터 증강 기법 한국정보과학회, 2021. 12., 이선경 외**

* 서론: back translation, 언어 모델로 대체 단어 예측 등의 방법은 추가적인 모델 학습이 요구되어 계산 비용이 크다, 우리는 텍스트 편집을 통한 간단한 데이터 증가 기법을 제안함.
* 제안 방법: 단어 단위 텍스트 편집 기법에는 단어 교체, 단어 위치 무작위 변경, 단어 무작위 삽입, 단어 무작위 삭제를 사용. 이를 문장 단위로 확장하여 문장 교체, 문장 위치 무작위 변경… 등의 문장 단위 텍스트 편집 기법도 활용.  
  우리의 오타를 사용한 데이터 증강 방법이 더 reasonable하다고 어필 할 수 있을듯.
* 실험: WebQuestions, SQuAD 데이터셋 이용, 성능 측정에는 Exact Match(EM), F1 이용  
  사용한 모델의 규모에 따라 / 데이터셋의 크기에 따라 성능 향상 정도를 비교하기도 함.

# **[5] 데이터 증강으로 강화된 Aug-TNE을 활용한 감성분석 한국지능시스템학회, 2022. 06., 김호승 외**

* 요약  
  
* 

**오탈자 단어 생성을 위한 가정**

앞서 언급한 오탈자 단어 생성 과정은 아래의 가정을 따른다. 첫째로 자소 자체를 바꾸는 경우에 의해 오탈자를 생성하면 자음은 자음끼리, 모음은 모음끼리만 교체하였다. 영어와 같은 다른 언어와는 달리 한국어의 각 글자는 초성, 중성, 종성의 구조로 구성되기에 각 자리에서 자음이 모음으로 바뀌거나, 모음이 자음으로 바뀌는 경우는 없다. **(당연한 얘기인 것 같아 삭제)**

둘째로 자소 자체를 바꾸는 경우에 의해 오탈자를 생성하면 중성 혹은 종성의 위치에서만 교체하고 초성은 고려하지 않았다. 보통 초성에 오탈자가 발생하면 이는 단순한 오탈자가 아니라 의도한 단어 의미 자체를 훼손하는 일이기에, 실수로 초성의 오탈자가 발생하여도 이는 바로 고치는 일이 허다하기 때문이다. 그 예로 ‘하지 마세요’라는 표현에서 ‘바지 마세요’, ‘하지 하세요’ 등 초성에 오탈자가 난 경우는 의도한 바를 온전히 전달하기 어렵기에 실제 오탈자가 발생하여도 바로 수정하기 마련이다. **(서론 뒷부분으로 이동)**

마지막으로 자소 자체를 바꾸는 경우와 자소 순서를 바꾸는 경우 모두에서 두글자 이상의 단어에서만 오탈자를 생성하였다. 보통 관형사나 감탄사로 사용되는 한글자의 단어에서 오타가 나는 경우는 극히 드물었기 때문이다. **(2.1로 이동)**

…오탈자 문장의 의미가 기존 문장과 달라질 가능성을 가진다….. …

**논문 제목**

1) 입력된 문장을 공백 단위로 토큰화한다.

역번역을 사용하는 방법은 그 특성상 새로 생성된 문장의 의미가 기존 문장과 달라질 가능성을 가진다.

1) 입력된 문장을 공백 단위로 토큰화한다.  
2) 공백 단위로 나뉘어진 토큰에 특수기호 등이 있을 경우 이를 기준으로 한번 더 토큰화한다.  
3) 그렇게 남겨진 토큰 중 하이퍼 파라미터로 입력한 오탈자 비율에 따라 2글자 이상의 단어들을 랜덤하게 선택한다.  
4) 선택된 단어에서 50%의 확률로 자소 자체를 바꾸는 경우 혹은 자소 순서를 바꾸는 경우 중 하나를 선택하여 오탈자로 변형한다.  
5) 오탈자로 생성된 단어를 활용하여 기존 문장 형태로 복원한다.  
6) 하이퍼 파라미터로 입력한 증강 비율에 따라 위 과정을 반복한다.

첫번째로, 사전 학습된 언어 모델 중 하나인 BERT []에 기반한 모델을 통해 일반 문장과 제안한 기법을 통해 생성한 오탈자 문장을 인코딩한 출력값을 비교하여, 그 차이를 확인해본다. 두 출력값의 차이가 크다는 것은 사람이 이해하기에는 같은 의미를 가지는 두 문장이 모델에게는 그렇지 않다는 것으로 해석할 수 있다. 이를 통해, 기존 문장에 오탈자를 포함시켜 새롭게 생성한 문장을 새로운 데이터로 활용할 수 있다는 결론을 얻을 수 있다.

Devlin, J., Chang, M., Lee, K. and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics. pp. 4171-4186. 2019.

영어 단어에서는 각 글자가 단어 표현에 있어 일정 비중을 차지하기에 ‘dog’라는 단어에서 ‘fog’, ‘dot’과 같이 한 글자만 오탈자가 발생하여도 본래의 의미를 보존하지 못한다. 반면 한국어 단어에서는 각 글자가 초성, 중성, 종성으로 구성되기에 ‘강아지’라는 단어에서 ‘겅아지’, ‘강아즈’와 같이 자소 단위로 오탈자가 발생하여도 본래 의미를 일부분 추측할 수 있다. → **서론 반영**

| 기존 데이터셋 정확도 | | | | | 88.0% | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 증강  데이터셋 정확도 | | 2 | | 4 | | 6 | | 8 | |
| 20% | | +0.7%p | | +0.008 | | +0.004 | |  | |
| 40% | | +0.003 | | +0.010 | | +0.012 | |  | |
| 60% | | +0.007 | | +0.012 | | +0.011 | | +0.008 | |
| 80% | | +0.005 | | +0.013 | | +0.010 | | +0.006 | |

(오타가 많은 현상에 대한 레퍼런스 관련) → . ‘비격식’이라는 표현이 괜찮아보여 내용을 수정했습니다.

다른 언어와는 달리 한 글자가 자소 단위로 구성되는 한국어의 특징을 반영한 고의 오탈자 규칙을 제안하여, 사람의 실수로 인한 텍스트를 고의로 생성함으로써 실제 데이터에 있음직한 텍스트를 만들어내고자 하였다.

**→** 다른 언어와는 달리 글자 내 자소의 변경 및 교체에 따라 발생되는 오탈자의 경우가 다양한 한국어의 사례에 초점을 맞추어 고의 오탈자 규칙을 상세히 규정하였다.

실험 목적 및 방향성이 확실히 명시되지 않았던 것 같아 아래에 정리합니다.

1. 제안하는 한국어 고의 오탈자를 활용하였을 때 문장을 망치는 오탈자 비율+문장을 늘리는 증강 비율, 각 비율에 비례하여 기존 데이터셋에 비해 성능이 얼마나 향상하는지 확인해보기 위함.
2. 각 비율에 비례하여 증가하지 않는다면 어떤 세팅에서 최선의 성능인지 확인 후 그에 대한 결과 해석  
   (언어 모델이 오탈자가 많이 포함된 문장을 학습할때 일반화가 잘 되는지,  
   그게 아니라면 오탈자가 어느정도로 포함된 문장을 학습해야 추론 과정에서 성능이 좋을지)

추가 실험 고려 사항

1. 학습에 전체 데이터를 사용하지 않고 샘플링하여 사용하는 방법 → 학습 시간의 비효율성 해결  
   (ex) nsmc 학습 데이터는 15만개인데, 2만개 정도로 샘플링하여 3번 정도 나눠 학습,  
   참고논문 [2]에서도 사용한 방식이기도 함)
2. 데이터 증강 비율에서 사실상 2,4,6,8배의 차이가 크다고 느껴지지 않으며, 기존 실험 결과 해석도 애매하기에 4,8,16배로 설정하는 것을 고려 (실험 횟수는 줄이되 그 차이가 크게 느껴지도록)
3. 오타 비율에서도 같은 이유로 20%,40%,60%,80%를 30%,60% or 40%,80%로 설정하는 것을 고려  
   (아직 확인해봐야 하지만 오타 비율 차이를 크게 두면 그에 따라서 Similarity 차이도 클 것으로 기대)
4. (필수 사항은 아니지만 작성해두겠습니다.) 현재 방식으로 데이터 증강을 진행하면 문장에서 오탈자를 만들지 않는 경우, 예를 들어 문장이 너무 짧거나 오타 비율이 낮은 경우도 이를 고치지 않고 똑같은 문장을 반복적으로 증강함, 이런 경우 반복된 문장을 제거하는 옵션?  
     
   밑에 사진에서 index 0과 같이 고칠 부분이 많으면 오탈자를 많이, 다양하게 만들지만  
   현재는 index 1과 같이 고칠 부분이 없으면 오탈자는 안만들어도 증강 비율만큼 증강해냄.  
   이런 경우 index 1은 한문장만 남도록 하는 옵션



I. item 1  
 A. Level 2 Item 1  
 B. Level 2 Item 2  
 C. Level 2 Item 3  
 1. Level 3 Item 1  
 2. Level 3 Item 2  
 3. Level 3 Item 3  
 4. Level 3 Item 4  
 5. Level 3 Item 5  
II. Level 1 Item 2

● Bullet Level 1 Item 1  
● Level 1 Item2  
 ○ Level 2 Item1  
● Level 1 Item3  
 ○ Level 2 Item1  
 ○ Level 2 Item2  
 ○ Level 2 Item3  
 • Level 3 Item1  
 ◦ Level 4 Item1  
 ◦ Level 4 Item2  
 ○ Level 2 Item4