

문장 임베딩과 데이터 증강을 활용한

문장 유사도 판별 성능 향상

**Using sentence embedding and data
augmentation**

**Improved sentence similarity detection
performance**

김태혁, 배홍식

지도교수 김희동

한국외국어대학교

공과대학 정보통신공학과

2022年 11月 18日

문장 임베딩과 데이터 증강을 활용한

문장 유사도 판별 성능 향상


Using sentence embedding and data augmentation


Improved sentence similarity detection
performance

소 속:	공 과 대 학	전 공: 정보통신공학
학 번:	201701059	이 름: 김 태 혁
학 번:	201701691	이 름: 배 홍 식

이 논문을 정보통신공학과 졸업논문으로 제출하오니 승인하여 주십시오.


2022年 11月 18日

성 명 : 김 태 혁(인) 

성 명 : 배 홍 식(인) 

위 학생의 논문 제출을 승인함.

2022年 11月 18日

지 도 교 수 : 김 희 동(인) 

목차

1. 서론	5
1.1 연구 배경 및 주제	5
2. 관련 기술동향 조사 및 분석	6
2.1 임베딩	6
2.1.1 문맥정보를 이용한 단어임베딩 BERT	6
2.1.2 RoBERTa를 이용한 문장임베딩	7
2.2 대조학습을 이용한 모델 구조	8
2.2.1 대조학습	8
2.2.2 Sentence BERT(SBERT)	9
2.2.3 DeCLUTR	11
2.2.4 CLEAR (Contrastive Learning for Sentence Representation)	12
2.2.5 SimCSE (Simple Contrastive Sentence Embedding)	13
2.3 데이터 증강	15
2.3.1 Back Translation	15
2.3.2 EDA (Easy Data Augmentation)	16
3. 본론	18
3.1 연구 제안 방안	18
3.2 연구 순서	18
3.3 연구 진행 과정	19
3.3.1 Load Dataset	19
3.3.2 Preprocessing	20
3.3.3 Load Pretrained Model	20

3.3.4 STS를 이용한 훈련	21
3.4 Evaluation.....	21
3.4.1 상관계수	21
3.4.2 STS를 이용한 성능 평가.....	22
3.5 데이터 증강	23
3.5.1 3어절 데이터셋 증강	23
3.5.2 4어절 이상 데이터셋 증강.....	24
3.5.3 데이터셋 예시.....	24
3.5.4 NLI와 STS 데이터를 활용한 추가 학습.....	24
4. 결론	27
5. 출처	29
6. 그림 출처	30

1. 서론

1.1 연구 배경 및 주제

사람은 문장에서 단어가 어떤 의미로 사용되었는지 문맥을 통해 바로 구분할 수 있지만 기계는 그렇게 할 수 없다. 그렇기 때문에 자연어 처리에서는 단어를 수치(벡터)로 표현해서 기계가 이해할 수 있도록 해야 한다. 이를 단어 임베딩(word Embedding), 문장 임베딩(Sentence Embedding)이라 한다.

단어임베딩 방법으로는 단어 벡터가 단어의 의미를 나타낼 수 있도록 신경망을 이용하여 대용량 코퍼스로 훈련한 Word2Vec, 단어보다 더 작은 단위인 서브워드(subword)로 분리하고 표현한 fastText, 문맥을 고려한 단어 임베딩 BERT(Bidirectional Transformers for Language Understanding) 등이 있다. 기존에 자주 사용되던 word2vec 모델은 대용량 코퍼스 학습을 위해 모델을 단순화하여 단어의 등장 비율을 주로 학습하게 되어 단어 간의 상대적 위치정보를 이용하지 않는다는 단점이 있다. 특히 단어 임베딩 BERT는 사전학습 언어 모델로 하류 태스크(downstream)로 전이 학습하고 세부조정하여 우수한 성능을 나타내지만, 문장 단위로 사용할 경우 계산량이 많아 효율적이지 못하다.

그렇기 때문에 개별 단어 임베딩을 계산하지 않고도 문장 의미를 파악할 수 있고, 문장들의 유사성을 판별할 수 있는 문장 임베딩 과정이 필요하다. 문장 임베딩을 사용하는 대표적인 예로는 FAQ(자주하는 질문), Q&A(질의응답 시스템) 등이 있다. FAQ에서는 어떤 질문 문장이 입력되면 FAQ에 있는 질문 문장들과 유사한지 확인하고 FAQ를 참조하게 한다. Q&A에서도 질문문장에 대해서 정확히 파악하고 응답을 해야 하는데 필요하다.

이러한 문장 임베딩 과정을 수행하기 위해 대조학습 방법을 사용한다. 대조학습 방법이란 임베딩 공간에서 유사한 문장(Positive Sample)들은 가까이 위치시키고, 유사하지 않은 문장(Negative Sample)은 멀리 떨어지게 하는 학습 방법이다. 대조학습은 데이터 간의 유사성 평가를 근간으로 한 프리텍스트 태스크를 수행함으로써 일반화가 용이하며 새롭게 데이터를 생성할 필요도 없기 때문에, 전통적인 자기지도학습의 한계를 극복했다고 평가된다.

이러한 문장임베딩의 한 방법인 대조학습을 수행하기 위해서는 많은 양의 데이터로 훈련을 해야 하지만, 데이터셋이 부족한 상황이다. 데이터셋이 부족한 상황을 해결하기 위해서 데이터 증강(Data Augmentation) 기술을 활용하여 데이터셋을 많이 만드는 것이 필요하다. 데이터 증강 방법에는 Back Translation, EDA(Easy Data Augmentation) 등이 있다.

본 논문에서는 문장 간의 의미적 유사도를 구하는 STS(Semantic Textual Similarity) 데이터셋에 "어순을 변경하여도 의미가 변하지 않는다는 한국어의 특성"을 이용하여 데이터를 증강시켰다. 이렇게 증강시킨 데이터셋으로 문장 임베딩 모델인 SBERT(Sentence BERT)를 학습시켜, 문장 간 유

사도의 정확도를 향상시킬 수 있도록 만드는 것이 본 논문의 최종 목표이다.

어순을 변경하여도 의미가 변하지 않는다는 한국어의 특성을 사용하여 데이터셋을 증강시켜 학습했을 경우, 기존 데이터셋들로 학습했을 시보다 문장 간 유사도 판별 성능 지표인 Cosine-Similarity를 Spearman 상관계수로 측정된 결과값들이 더 높게 도출되는 결과를 얻었다.

2. 관련 기술동향 분석 및 활용

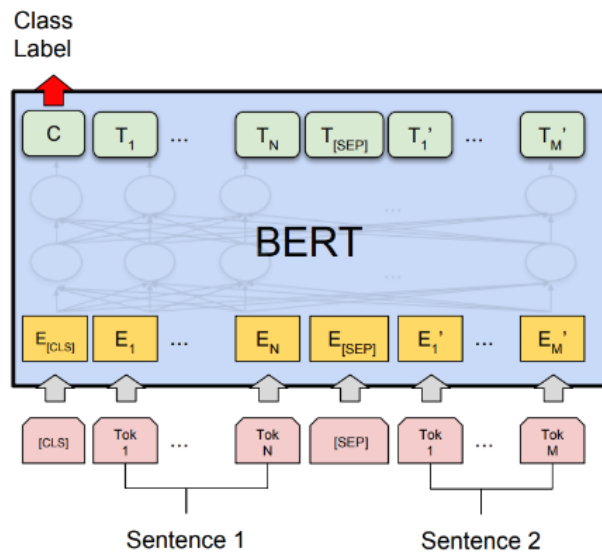
2.1 임베딩

컴퓨터는 사람들이 일상생활 속에서 사용하는 언어(자연어)를 직접 그 형태로 처리할 수 없다. 컴퓨터는 수치 연산만 가능하기 때문에 자연어를 숫자나 벡터 형태로 변환해야 한다. 이러한 과정을 자연어 처리 분야에서는 임베딩(Embedding)이라고 부른다. 즉, 임베딩(Embedding)은 단어나 문장을 수치화하여 벡터 공간으로 표현하는 과정을 의미한다. 따라서 임베딩 품질이 좋다면, 단순한 인공지능 모델로도 훌륭한 결과를 얻을 수 있다.

또한 자연어처리에서 임베딩이라고 하면 크게 단어 임베딩과 문장 임베딩이 있다. 단어 임베딩이란 개별 단어를 벡터로 표현하는 방법이다. 또한 단어 임베딩의 경우 동음이의어에 대한 구분을 하지 않기 때문에 의미가 다르더라도 단어의 형태가 같다면 동일한 벡터 값으로 표현되는 단점이 있다. 문장 임베딩은 전체 문장의 전체 문장의 흐름을 파악해 개별 단어를 벡터로 표현하는 방법이다. 따라서 문맥적 의미를 지닌다는 장점이 있고, 유사성을 판별할 수 있으며, 문장벡터의 예측이 가능하다.

2.1.1 문맥정보를 이용한 단어임베딩 BERT

BERT(Bidirectional Encoder Representation from Transformer)는 구글(Google)이 개발한 사전훈련 언어 모델이며 특정 과제를 수행하기 전 사전 훈련 임베딩 (Embedding)을 통해 특정 과제의 성능을 더 좋게 하는 언어 모델이다. BERT 모델은 양방향성을 가진 [그림 1]트랜스포머(Transformer)의 구조에서 인코더를 적층 시켜 만든 모델이다.

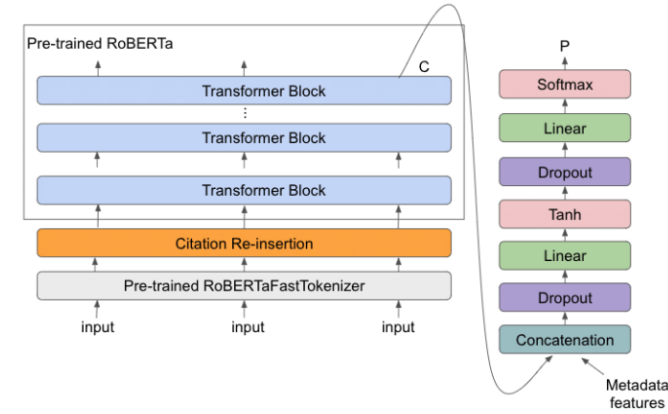


[그림 1] BERT 구조

또한 BERT는 다음 문장을 예측하는 NSP(Next Sentence Prediction)와 문장에서 가려진 단어를 예측하는 MLM (Masked Language Model) 두 가지 방식을 사용한다. MLM은 문장에서 몇 개의 임의의 단어를 마스킹(masking)하고, 문장의 문맥 정보를 파악하여 가려진(masked) 해당 단어를 예측하여 단어 분산 표현에 맥락 정보를 획득한다. NSP는 두 개의 입력문이 연속 문장인지, 관련이 없는 임의 조합 문장인지를 분석하고 판단하도록 훈련함으로써 문장 표현을 학습한다.

2.1.2 RoBERTa를 이용한 문장임베딩

RoBERTa는 자체 지도 방식으로 대규모 영어 데이터 코퍼스에 대해 사전 훈련된 변환기 모델이다. 이것은 사람이 어떤 식으로든 레이블을 지정하지 않고, 해당 텍스트에서 입력 및 레이블을 생성하는 자동 프로세스를 통해 원시 텍스트에 대해서만 사전 훈련되었음을 의미한다. 보다 정확하게는 MLM(Masked Language Modeling)을 목표로 사전 훈련되었다. 문장을 취하면 모델은 입력에 있는 단어의 15%를 무작위로 마스킹한 다음 마스킹 된 전체 문장을 모델을 통해 실행하고 마스킹 된 단어를 예측해야 한다. 이것은 일반적으로 단어를 차례로 보는 기존의 순환 신경망(RNN)이나 미래 토큰을 내부적으로 마스킹 하는 GPT와 같은 자동 회귀 모델과 다르다. 이를 통해 모델은 문장의 양방향 표현을 학습할 수 있다. 또한 RoBERTa는 BERT의 NSP를 없앤 언어모델인데도 하류 태스크의 성능이 높게 나와 NSP가 문장 임베딩에 기여하지 못한다는 것을 의미한다. 문장에 사용된 단어가 조금 변경되어도 완전히 다른 의미로 바뀌는 경우가 있어 문장 벡터를 구하는 것이 도전적 과제이다.



The RoBERTa model architecture.

[그림 2] RoBERTa 모델 구조

2.2 대조학습을 이용한 모델 구조

2.2.1 대조학습

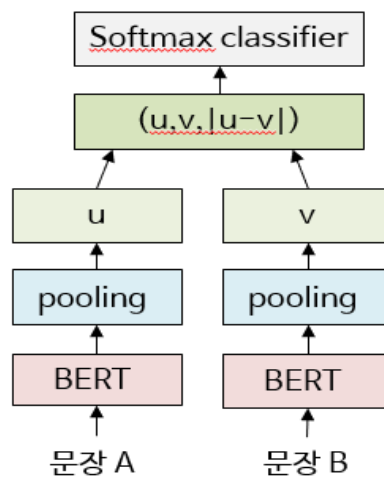
대조 학습이란 기준점이 되는 Anchor와 Positive Sample, Negative Sample들이 주어지면 임베딩 된 벡터공간에서 Anchor와 Positive는 가까이 위치하고, Negative는 멀리 위치시키는 임베딩 방법을 학습시키는 것이다. 또한 주어진 데이터들의 어느 부분이 "유사"한지 혹은 "다른"지를 학습함으로써 데이터가 갖고 있는 고차원의 특징을 배울 수 있다. 대조학습의 주안점으로는 훈련 sample에 대해서 데이터를 증강하여 훈련 sample에 약간의 변형을 준 Positive Sample과 Negative Sample을 만드는 방법이 필요하다. 데이터 증강 방법으로는 유의어 대체, 어순 변경, 단어 삭제, 단어 삽입이 있다. 이러한 증강은 컴퓨터가 자동으로 처리할 수 있어야 한다.

대조학습은 대상들의 차이를 좀 더 명확하게 보여줄 수 있도록 학습함으로써 문장 간 유사도 판별 성능의 정확도를 높이는 좋은 방법이다. 이 점을 활용하여 본 논문에서는 대조학습을 이용한 문장 임베딩 모델인 SBERT의 구조를 이용하였다. 대조학습에 들어가는 데이터의 증강을 위해 기존 STS 데이터에 어순을 바꾸어도 의미가 변하지 않는 한국어의 특성을 활용하여 Positive Sample과 Negative Sample을 제작하여 활용하였다.

2.2.2 Sentence BERT(SBERT)

BERT는 단어 임베딩을 위한 MLM과 두 문장의 인접 여부를 판단하는 NSP를 사용하여 문장의미를 얻으려 하였다. 그러나, NSP는 문장 임베딩에 영향을 주지 못한다는 것이 여러 연구들의 일치된 결론이다. 이를 개선하기 위해서 사전 학습된 BERT를 이용하여 하류태스크에서 세부조정하여 문장을 처리하는 것이 SBERT이다.

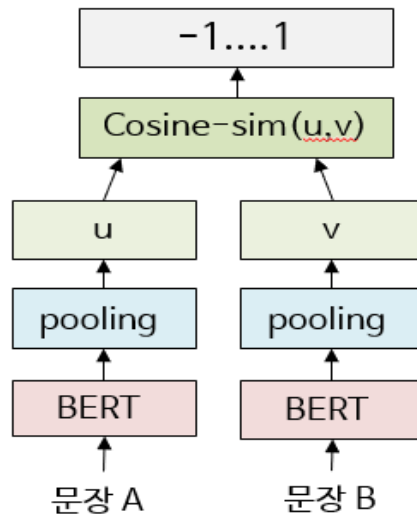
SBERT를 학습하는 방법으로는 2가지가 있다. 첫 번째 방법은 문장 쌍 분류 태스크로 파인 튜닝하는 것이다. 대표적으로는 NLI(Natural Language Inferencing) 문제를 푸는 것이다. NLI는 2개의 문장이 주어지면 수반관계인지, 모순 관계인지, 중립관계인지를 맞추는 문제이다.



[그림 3] 문장 분류 SBERT구조

[그림 4]처럼 우선 문장 A와 문장 B 각각을 BERT의 입력으로 넣고, 앞서 BERT의 문장 임베딩을 얻기 위한 방식이라고 언급했던 평균 풀링 또는 맥스 풀링을 통해서 각각에 대한 문장 임베딩 벡터를 얻었다. 여기서는 이를 각각 u 와 v 라고 칭한다. 그리고 나서 u 벡터와 v 벡터의 차이 벡터를 구한다. 이 벡터는 수식으로 표현하면 $|u-v|$ 이다. 그리고 이 세 가지 벡터를 연결(concatenation)한다. 세미콜론(;)을 연결 기호로 한다면 연결된 벡터의 수식은 다음과 같다.

$$h = (u; v; |u - v|)$$



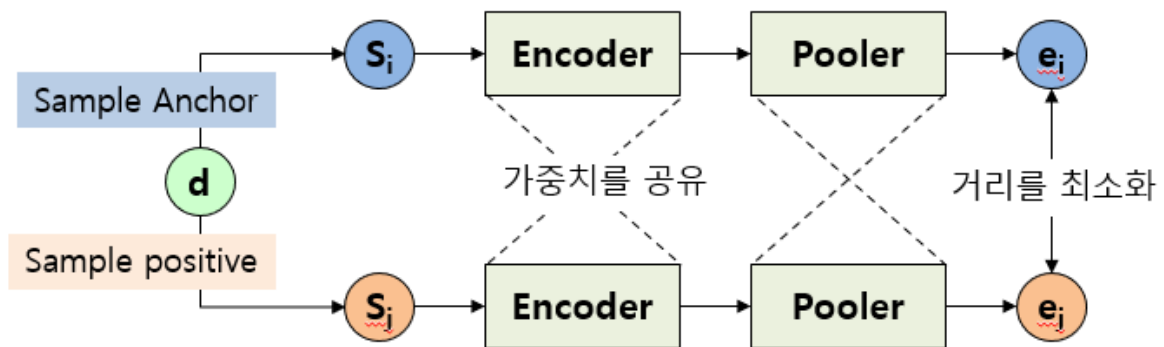
[그림 4] 두 문장의 유사도 측정하는 SBERT구조

2번째 방법으로는 문장 쌍 회귀 태스크로 파인 튜닝 하는 것이다. 대표적으로 STS(Semantic Textual Similarity) 문제를 푸는 것이다. STS란 2개의 문장으로부터 의미적 유사성을 구하는 문제를 말한다. SBERT는 STS 데이터를 학습하기 위해 위의 [그림 5]와 같은 구조를 가진다.

문장 A와 문장 B 각각을 BERT의 입력으로 넣고, 평균 풀링 또는 맥스 풀링을 통해서 각각에 대한 문장 임베딩 벡터를 얻는다. 이를 각각 u 와 v 라고 하였을 때, 이 두 벡터의 코사인 유사도를 구한다. 그리고 해당 유사도와 레이블 유사도와의 평균 제곱 오차(Mean Squared Error, MSE)를 최소화하는 방식으로 학습한다. 코사인 유사도의 값의 범위는 -1과 1사이므로 위 데이터와 같이 레이블 스코어의 범위가 0~5점이라면 학습 전 해당 레이블들의 값들을 5로 나누어 값의 범위를 줄인 후 학습할 수 있다.

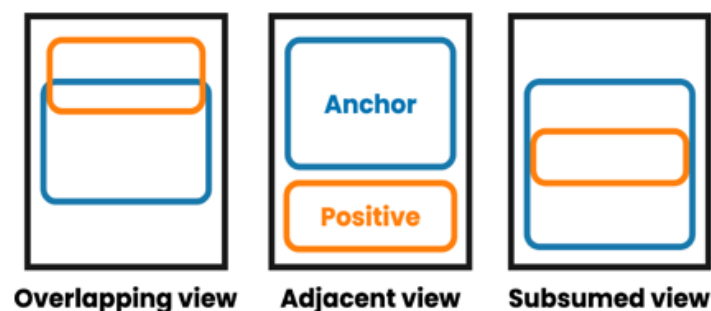
2.2.3 DeCLUTR (Deep Contrastive Learning for Unsupervised Textual Representations)

DeCLUTR는 사전학습 언어모델에 최초로 대조학습을 결합한 방식을 제안하였다. [그림 6]에 제안구조를 나타내었는데, 입력으로 2개의 문장이 2개의 동일한 encoder에 들어가고 동일한 pooler가 임베딩 결과를 출력한다. 어떠한 encoder에 대한 제약은 없으나 실험에는 BERT 기반의 RoBERTa를 사용하였다.



[그림 5] DeCLUTR 구조

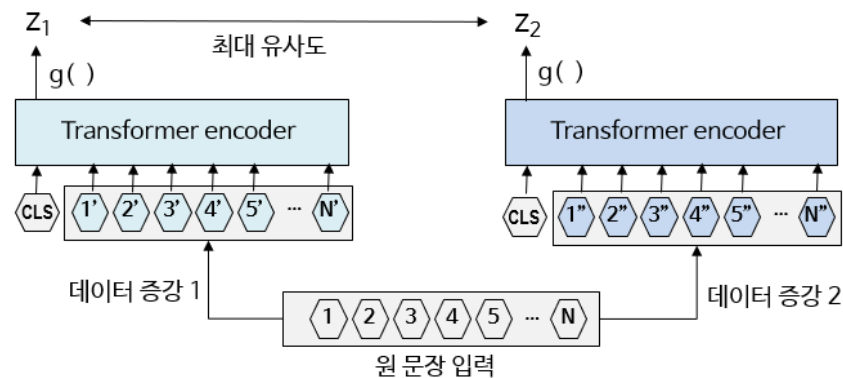
DeCLUTR에서는 표현 벡터가 Positive pair는 가까이 위치하도록 하고, Negative pair의 벡터는 멀리 위치하도록 한다. 샘플이 같은 Segment에서 온 것이면 Positive로, 다른 Segment나 다른 문서에서 왔으면 Negative로 표현한다.



[그림 6] DeCLUTR에서 사용한 데이터증강방식

2.2.4 CLEAR (Contrastive Learning for Sentence Representation)

CLEAR는 BERT의 사전학습 태스크를 변경하여 성능이 개선된 문장 임베딩을 얻고자 하였다. 사전학습 언어모델에 단어 임베딩을 위한 MLM(마스크 LM)과 문장 임베딩을 위한 대조학습을 결합한 것이다. MLM에 의해 단어 레벨의 특징을 추출하는데 도움을 주고 대조학습은 이를 지원한다. 동일 문장을 다르게 변형하여도 임베딩한 값이 최소가 되도록 훈련함으로써 유사 문장 인식율을 높인다. CLEAR의 구성은 아래 [그림 8]과 같다.



[그림 7] CLEAR 구조

[그림 9]와 같이 CLEAR는 단어 삭제, 단어 뭉치 삭제, 어순 변경, 비슷한 말로 대체 등 4가지의 증강 기법을 사용한다. 본 논문에서는 어순을 변경하여도 의미가 변하지 않는다는 한국어의 특성을 이용하여 어순 변경 방법을 채택하여 연구하였다. 하지만 어순 변경의 경우 어절 수가 많아질 경우 오류가 많이 발생할 수 있으므로 주의가 필요하다. 어순 변경 방법을 제외하고, 단어 삭제나 유의어 대체 방법으로도 많은 양의 Positive Sample들을 만들어 문장 간 유사도 판별 성능을 향상시켜보는 것이 향후 과제이다.



[그림 8] CLEAR 문장 증강 기법

2.2.5 SimCSE (Simple Contrastive Sentence Embedding)

SimCSE는 사전학습 언어모델을 Contrastive Learning으로 Fine-Tuning 하는 문장임베딩 기법이다. CLEAR에서는 입력 문장에 대해서 데이터 증강에 의해 서로 다른 문장을 생성하고 부호화 한다. 한편 simCSE에서는 입력 문장에 대해서 mask dropout이 다르게 두 번 부호화 하여 Positive Sample을 2개의 쌍으로 취한다. 그리고 서로 다른 문장을 BERT로 입력한 때에는 Negative Sample로 취한다. 또한 CLEAR는 단어레벨에서 증강한 것이어서 이산적(discrete)이라면, SimSCE는 BERT에서 dropout 한 것으로 연속적(continuous)인 변환인 점이 다르다. SimSCE는 Positive Sample의 쌍은 문장길이가 같고 Negative Sample 쌍들은 다른 문장으로 만들기 문장 길이가 다르게 된다. Positive sample을 만드는 방법으로 Unsupervised SimCSE와 Supervised SimCSE의 두 가지 방법을 사용한다.

1. Unsupervised SimCSE

같은 문장에 대해서 다른 dropout mask를 적용한 2개의 문장 임베딩을 Positive pair로 한다. 그리고 같은 문장을 2번 모델에 통과시켜 최소한의 data augmentation으로서 dropout 사용한다.

Unsupervised 방식의 SimCSE는 Positive instance를 만드는 방법으로써 drop out을 data augmentation으로 활용한다. 예를 들어 masked language model(MLM)로 학습된 RoBERTa를 encoder로 활용할 때, 같은 input x_i 에 서로 다른 drop out mask z_i, z'_i 를 적용하여 두 번 forward를 수행함으로써 Positive instance를 만들어내고, 이를 식으로 정리하면 아래와 같다.

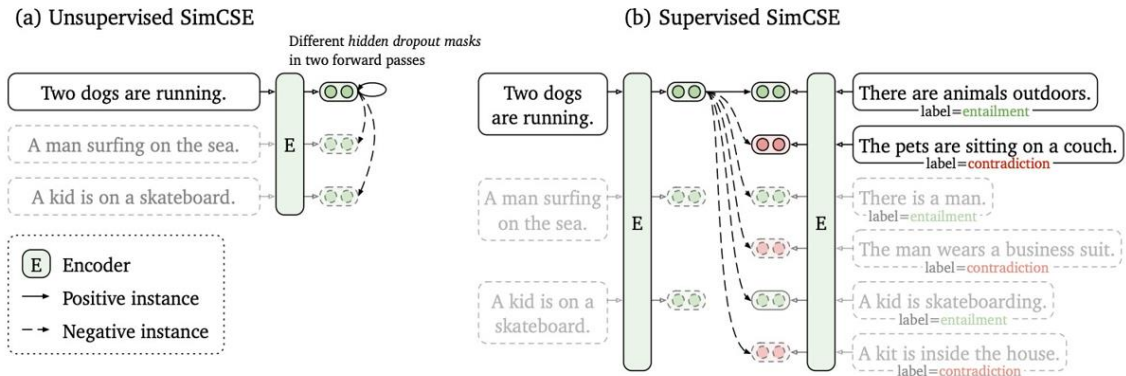
$$l_i = -\log \frac{e^{\text{sim}(h_i^{z_i}, h_i^{z'_i})/\tau}}{\sum_{j=1}^N e^{\text{sim}(h_i^{z_i}, h_j^{z'_j})/\tau}}$$

2. Supervised SimCSE

NLI 데이터 세트의 함의(entailment) 쌍을 Positive pair로 사용하고, 의미적으로 함의관계에 있는 문장 임베딩을 공간상에 인접하게 분포하도록 학습하는 방법이다. Supervised 방식의 SimCSE는 NLI task의 dataset를 활용 시, example triple에서 entailment label이 붙어있는 example pair를 Positive instance로 사용하고, 그 외의 모든 경우를 모두 Negative instance로 취급한다. 특히 하나의 example triple에서 contradiction label이 붙어있는 example pair는 hard Negative로 사용된다. (neutral label의 경우는 활용하지 않는다.) 이를 식으로 정리하면 아래와 같다.

$$-\log \frac{e^{\text{sim}(h_i, h_i^+)/\tau}}{\sum_{j=1}^N (e^{\text{sim}(h_i, h_i^+)/\tau})}$$

아래 [그림 10]과 같이 Unsupervised SimCSE에서는 동일문장을 2번 임베딩하여 대조학습 한다. Supervised SimCSE의 경우 함의관계에 있는 문장을 Positive 쌍으로 하여 대조학습한다.



[그림 9] Unsupervised SimCSE와 Supervised SimCSE

2.3 데이터 증강

문장임베딩의 한 방법인 대조학습을 수행하기 위해서는 많은 양의 데이터로 훈련을 해야 하지만, 데이터셋이 부족한 상황이다. 데이터셋이 부족한 상황을 해결하기 위해서 데이터 증강(Data Augmentation) 기술을 활용하여 데이터셋을 많이 만드는 것이 필요하다. 이러한 데이터 증강을 통해 본 논문의 핵심 목적인 문장 유사도 판별 성능 향상을 기대해 볼 수 있다. 아래에는 데이터 증강 기법의 예시들이다.

2.3.1 Back Translation

Back Translation은 데이터 증강의 한 방법으로서, 대상 언어의 콘텐츠를 문자 그대로 원본 언어로 다시 번역하는 프로세스이다. Transformer와 같은 성능이 좋은 모델들은 대량의 병렬 코퍼스로 학습을 한다. 하지만 이는 시간과 비용이 너무 많이 들기 때문에 합성 코퍼스를 이용하여 만드는 방법을 고안하였다. 즉, 기존의 훈련된 반대 방향 번역기를 사용해 단일 언어 코퍼스에 대한 번역을 진행하며 합성 synthetic 병렬 코퍼스를 만든 후, 이것을 기존 양방향 병렬 코퍼스에 추가하여 훈련에 사용하는 방식이다. 예를 들어 콘텐츠를 한국어에서 영어로 번역하는 경우 번역가는 번역된 옵션의 의도를 쉽게 이해할 수 있도록 역 번역 한국어도 작성한다. Back Translation은 번역 메모리나 번역가가 사용하는 용어집과 같은 기타 리소스에 영향을 미치지 않는다. Back Translation을 이용할 시 단일 언어 데이터를 가상 병렬 데이터로 변환하여 학습데이터의 양을 증가시킨다. Back Translation 같은 경우 NMT(Neural Machine Translation) 즉, 신경망 기계 번역의 장점인 번역기를 만들 때 하나의 병렬 코퍼스로 두 개의 번역 모델을 만들 수 있다는 특성에 기반한다. 반대 방향의 번역기를 사용하여 단일 언어 코퍼스를 학습시켜 합성 Synthetic 병렬 코퍼스를 제작하고, 이를 양방향 병렬 코퍼스에 추가하여 훈련한다.

비슷한 Translation 방법으로 Copied Translation이 존재하는데 이는, 반대 방향 번역기의 활용 없이 단일 언어 코퍼스 활용하는 방법론이다. 즉 소스 쪽과 타겟 쪽에 똑같은 데이터를 넣어 훈련시키는 방법이다. 하지만 소스 언어의 어휘(vocabulary)에 타겟 언어의 어휘가 포함되는 불필요함을 감수해야 한다는 단점이 존재한다.

Back Translation을 사용하여 인공데이터를 만드는 방법에는 5가지가 존재한다.

1. Greedy Search: 디코더 단계마다 가장 확률이 높은 단어만을 선택
2. Beam Search: 디코더 단계에서 가장 확률이 높은 n개를 선택한다. 다음 step의 input으로 n을 넣어 문장 전체의 확률을 고려한다.

3. Random Sampling: 디코더 step에서 나온 각 단어의 확률에 기반으로 Sampling을 통해 단어를 선택하고 번역한다.
4. Top 10 Sampling: 디코더 Step에서의 단어의 확률이 높은 10개만을 선택하여 Sampling 한다.
5. Beam + Noise: Beam Search에서 나온 문장을 0.1% 삭제하고 빈칸으로 변경한 후, 2개의 단어로 변경한다.

위와 같은 방법들 중에서 어려움지표(PPL)로 비교 시, Greedy Search방법이 제일 쉽고 Sampling이 제일 어렵다. 또한 PPL이 가장 높은 Beam+Noise 방법이 성능 향상의 폭이 가장 크다. 데이터 수가 작을 시에는 Beam Search 방법을 사용한 것이 성능 향상이 좋은 것을 확인할 수 있다. 위와 같은 5가지 방법을 통해 만든 인공 데이터를 추가하여 학습하면 모두의 성능이 증가한다. 하지만 Back Translation의 방법은 문장의 구조가 정확하지 못한 경우와 주어가 생략된 문장의 역번역은 구조뿐 아니라 의미상으로도 정확하지 못한 문장을 생성해 내기 때문에 문장 유사도 판별 성능 향상에 좋지 않은 결과를 야기할 수 있다.

2.3.2 EDA (Easy Data Augmentation)

EDA(Easy Data Augmentation)는 말 그대로 쉽게 데이터를 증강하는 방법이다. 데이터를 변형시켜 양을 강제로 늘리는 방법을 사용한다. EDA는 텍스트 분류 작업 성능을 부스팅 할 수 있다. 긴 문장은 짧은 문장보다 단어가 많으므로, 원래의 라벨을 유지하여 노이즈에 상대적으로 영향을 덜 받는다. EDA 방법을 사용하여 데이터셋을 늘릴 시, 가장 중요한 점은 라벨이 훼손되지 않도록 하는 것이 가장 중요하다. EDA의 단점은 BERT와 같은 사전 훈련된 모델을 사용할 때에는 무시할 수 있을 정도의 효율이다.

EDA에서 Training set에서 데이터가 주어졌을 때, 아래와 같은 4가지 방법 중 랜덤으로 선택하여 적용한다.

$$n = a * l$$

n : 바뀐 단어 수, a : 백분율 변경, l : 문장 길이

1. Synonym Replacement (SR): 동의어 대체

: 주어진 문장에서 stop words가 아닌 n 개의 단어를 선택하여 synonyms 중 랜덤으로 치환한

다. α 값이 크면 오히려 성능이 떨어질 수 있다. 너무 단어를 많이 바꾸기 때문에 문장의 의미가 바뀌어 버릴 수 있다.

2. Random Insertion (RI): 임의 삽입

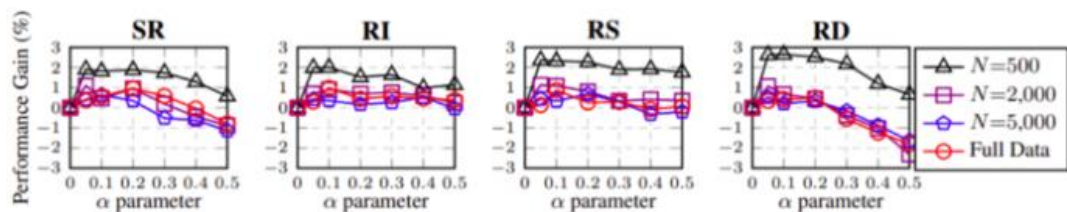
: 문장에서 stop word가 아닌 단어의 동의어를 찾고 해당 동의어를 문장의 임의의 위치에 삽입하는 것을 n 번 반복한다. α 값이 커질수록 성능이 더욱 안전하다. 문장 순서가 원래의 단어와 연산에서 유지되기 때문이다.

3. Random Swap (RS): 랜덤 스왑

: 랜덤으로 2개의 단어를 선택하여 위치를 바꾸고 n 번 반복한다. α 값이 커지면 많은 교환이 일어나 전체 문장이 바뀐 것과 동일해지기 때문에 문장의 의미가 바뀌어 버릴 수 있다.

4. Random Deletion (RD): 무작위 삭제

: 문장 내에서 각각의 단어에 대해 α 확률로 지운다. α 값이 작으면 높은 성능을 보이고, α 값이 높으면 성능이 저하된다. 너무 많이 삭제되어 버리면 문장에 대한 이해를 할 수 없기 때문이다.



[그림 10] EDA 4가지 기법

성능이 저하된다. 위와 같은 4가지 방법 중 본 논문에서는 어순을 바꿔도 의미가 변하지 않는 한국어의 특성을 활용하여 랜덤 스왑(RS) 방법을 채택하여 연구하였다. 랜덤 스왑 방법을 제외하고, 무작위 삭제나 동의어 대체 방법으로도 많은 양의 Positive Sample들을 만들어 문장 간 유사도 판별 성능을 향상시켜보는 것이 향후 과제이다.

3. 본론

3.1 연구 제안 방안

본 논문에서는 자연어 처리에 특화된 사전학습 모델인 BERT로부터 문장 임베딩을 얻을 수 있는 Sentence BERT(SBERT)를 활용하였다. SBERT는 BERT를 Fine-Tuning하여 문장 임베딩 성능을 우수하게 개선한 모델이다. SBERT는 2개의 BERT에 각각 입력이 가해지기 때문에 문장 간 유사도를 측정하는데 다른 언어 모델보다 더 정확하고 빠른 문장 임베딩 능력을 보이기 때문에 문장 임베딩 사용 모델로 SBERT를 채택하였다. 본 논문에서 사용한 SBERT는 서울대학교 전산언어학 연구실에서 개발한 사전 훈련된 한국어 특정 Sentence BERT(KoSBERT)를 사용한다.

본 논문의 데이터셋은 텍스트의 의미적 유사도를 측정하는 STS 데이터와 한국어 자연어 추론 데이터인 NLI 데이터를 활용하여 어순이 자유로운 한국어 특성으로 문장들의 어순들을 바꿔 데이터셋을 증강시켜 문장 간의 유사도 정확도를 향상시킨다.

본 논문의 성능 평가는 EmbeddingSimilarityEvaluator 함수를 통해 학습 시 사용할 Validation 검증기와 모델 평가 시 사용할 test 검증기를 만들어 Cosine, Euclidean, Manhattan 그리고 Dot Product 각각의 Pearson correlation과 Spearman's rank correlation의 평균을 구하여 모델의 선형성과 단조성을 측정한다.

3.2 연구 순서

1. Sentence-Transformers 라이브러리를 활용한 SBERT를 Fine-Tuning 한다.
2. Fine-Tuning을 하기 위한 데이터셋으로 기존의 데이터 셋인 KLUE-STS를 이용한다.
3. 기존의 데이터셋을 사용한 언어 모델들을 성능 평가한다.
4. 기존의 KLUE-STS 데이터셋을 증강시킨다.
 - 4.a 2,3어절의 문장들은 어순 전체를 바꾸어 주는 코드를 코딩하여 사용한다. (Positive)
 - 4.b 4어절 이상의 문장들은 어순을 선택해서 바꾸어 주는 코드를 사용한다. (Positive)
 - 4.c 기존의 KLUE-STS 데이터셋에서 의미가 완전히 다른 문장들을 이용하여 제작한다. (Negative)
5. 증강시킨 데이터셋을 활용하여 SBERT를 다시 Fine-Tuning 시켜본다.
6. Fine-Tuning한 언어모델들의 성능 평가를 진행한다.

7. 더 좋은 성능의 언어 모델을 찾기 위해 Continue Learning(NLI+STS(AUG))를 다시 Fine-Tuning 시켜본다.

8. 단지 데이터셋의 양이 증가해서 정확도가 높아진 것인지(KLUE-STS + KorNLU-STS), 데이터를 증강시킨 데이터셋을 이용하여 정확도가 높아진 것인지(KLUE-STS(AUG Version))를 비교 분석을 한다.

3.3 연구 진행 과정

3.3.1 Load Dataset

STS는 텍스트의 의미적 유사도를 측정하는 문제이다. 모델이 의미상 두 문장의 친밀도를 얼마나 잘 잡아내는지 또는 문장의 의미적 표현을 얼마나 잘 구현하는지 평가하는데 일반적으로 사용된다.

STS 데이터는 KorNLU STS와 KLUE(Korea Language Understanding Evaluation)-STS 2종류가 있다. KorNLU 데이터의 경우 KLUE에 비해 데이터의 수는 월등히 많지만 데이터의 복잡도가 굉장히 낮은 데이터이기 때문에 sentence embedding이 목적이라면 KLUE 데이터를 활용하는 것이 더 좋은 품질의 임베딩이 가능하고, 성능도 더 좋다.

NLI(Natural Language Inference)는 두 문장이 비슷한지, 반대되는지, 관계가 없는지 분류하는 task로서 전제, 가설 label(entailment, contradiction, neutral)로 구성되는 자연어 추론 데이터셋이다.

실습에 사용할 데이터는 KLUE-STS 데이터셋이다. 해당 데이터셋은 'train', 'validation'으로만 구성되어 있지만, 'train' 세트의 10%를 샘플링 하여 validation으로 사용하고 기존 'validation'을 test 용도로 사용하였다.

Datasets: kimtaehyuk11/**klue** like 0

[Dataset card](#) [Files and versions](#)

Dataset Preview

Subset: sts Split: train

guid (string)	source (string)	sentence1 (string)	sentence2 (string)	labels (json)
klue-sts-v1_train_00000	airbnb-rtt	숙소 위치는 찾기 쉽고 일반적인 한국의 반지하 숙소입니다.	숙박시설의 위치는 쉽게 찾을 수 있...	{ "label": 3.7, "real-label": ...
klue-sts-v1_train_00001	policy-sampled	위반행위 조사 등을 거부·방해·기피한 자는 500만원 이하 과...	시민들 스스로 자발적인 예방 노력...	{ "label": 0, "real-label": 0, "binary-...
klue-sts-v1_train_00002	paraKQC-sampled	회사가 보낸 메일은 이 지메일이 아니라 다른 지메일 계정으...	사람들이 주로 네이버 메일을 쓰는...	{ "label": 0.3, "real-label": ...
klue-sts-v1_train_00003	policy-sampled	간접 고용안전지원금은 지역고용대응 등 특별지원금, 지자체...	고용보험이 1차 고용안전망이라면, ...	{ "label": 0.6, "real-label": ...
klue-sts-v1_train_00004	airbnb-rtt	호스트의 답장이 늦으나, 개선될 것으로 보입니다.	호스트 응답이 늦었지만 개선될 것...	{ "label": 4.7, "real-label": ...
klue-sts-v1_train_00005	policy-sampled	정부가 새로운 일자리를 직접 창출하는 노력도 배가하겠습니...	세계에서 우리만큼 오랜 역사와 문화...	{ "label": 0, "real-label": 0, "binary-...
klue-sts-v1_train_00006	airbnb-rtt	지하철을 타도 30분안에는 이동이 가능합니다!	지하철을 탄다고 해도, 30분이면...	{ "label": 4, "real-label": 4, "binary-...

[그림 11]Huggingface 내에 업로드한 KLUE-STS데이터

3.3.2 Preprocessing

InputExample() 클래스를 통해, 두 개의 문장 쌍과 라벨을 묶어 모델이 학습할 수 있는 형태로 변환해준 후, 학습에 사용할 train 데이터는 배치학습을 위해 DataLoader()로 묶고, EmbeddingSimilarityEvaluator()을 통해 학습 시 사용할 validation 검증기와 모델 평가 시 사용할 test 검증기를 만들었다.

3.3.3 Load Pretrained Model

STS를 Fine-Tuning 하기 위해 사용할 사전학습언어모델을 로드하는 과정이며, 해당 실습에서는 Huggingface model hub에 공개되어 있는 KLUE/Roberta-base 모델을 사용하였다. 이 단계에서는

문장 임베딩을 어떻게 계산할 건지를 결정하는 풀링 연산을 수행해야 하는데, 본 논문에서는 Mean-Pooling을 설정하였다. Mean-Pooling이란 모델이 반환한 모든 토큰 임베딩을 더해진 후, 더해진 토큰 개수만큼 나누어 문장을 계산해주는 임베딩이다.

3.3.4 STS를 이용한 훈련

STS 학습 시 loss function의 경우 CosineSimilarityLoss()를 사용하며, 파라미터로는 4 epochs, learning-rate warm-up의 경우 train의 10%를 설정하였다.

3.4 Evaluation

본 논문의 성능 평가는 EmbeddingSimilarityEvaluator 함수를 통해 학습 시 사용할 Validation 검증기와 모델 평가 시 사용할 test 검증기를 만들어 Cosine, Euclidean, Manhattan 그리고 Dot Product 각각의 Pearson correlation과 Spearman's rank correlation의 평균을 구하여 모델의 선형성과 단조성을 측정한다. 앞으로 나오는 빨간색의 네모 안의 값들은 문장 간 Cosine-Similarity를 Spearman 상관계수로 측정된 값들을 나타낸 것이다.

3.4.1 상관계수

Pearson 상관계수

Pearson 상관계수는 두 변수 간에 선형성(linearity)이 얼마나 강한지를 측정하기 위해 사용된다. 선형성이 강하다는 것은 변수들의 관계가 직선에 의해 잘 모델링 된다는 뜻이다. Pearson 상관계수를 구하는 공식은 아래와 같다.

$$PLCC = \frac{\sum_{i=1}^M (X_i - \bar{X}) (Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^M (X_i - \bar{X})^2 (Y_i - \bar{Y})^2}}$$

X_i, Y_i 는 X, Y 변수들의 i 번째 샘플 값을 의미하고, \bar{X}, \bar{Y} 는 X, Y 변수의 평균값들을 의미한다. 그리고 M 은 샘플의 개수이다. 두 변수가 동시에 일정한 비율로 증가하거나 감소하는 것을 두고 양의 선형 관계가 존재한다고 말한다. 반대로 한 변수는 감소할 때 다른 변수는 증가한다면 음의

선형 관계가 존재하는 것이다. 강한 양의 선형 관계라면 Pearson 상관계수는 1에 가까워지고, 강한 음의 선형 관계라면 -1에 가까워진다. Pearson 상관계수가 0에 가깝다면, 두 변수는 서로 선형 상관성이 거의 없는 것이다.

Spearman 상관계수

Spearman 상관계수는 단조성(monotonicity)을 평가하기 위해 사용된다. 단조성이 좋다는 것은 한 변수의 값의 크기가 커지면(또는 작아지면) 다른 변수의 크기도 커진다(또는 작아진다)는 뜻이다. Spearman 상관계수는 아래와 같이 구한다.

$$SROCC = 1 - \frac{6}{M(M^2 - 1)} \sum_{i=1}^M d_i^2$$

여기서 나머지는 위와 같고, d_i 는 두 변수의 값들을 크기 순으로 정렬한 것에서 i 번째 값들의 차를 의미한다. 크기 순으로 정렬한 두 변수 값들의 차이가 작을수록 Spearman 상관계수는 커진다. 즉, Spearman 상관계수는 한 변수의 값이 커지면 다른 변수의 값도 단조롭게 커지는지를 알아보기 위한 것이다. Spearman 상관계수는 1에 가까울수록 두 변수는 좋은 단조 상관성을 갖고 있는 것이고, 0에 가깝다면 단조 상관성이 거의 없는 것이다.

위에서 만든 Test Evaluator를 활용하여 모델의 문장 임베딩 간 코사인 유사도가 얼마나 Gold Label에 가까운지에 대해 각 상관계수를 구하여 나타낸다.

3.4.2 STS를 이용한 성능 평가

기존의 KLUE_STS 데이터셋을 Test 검증기로 모델 성능을 평가한 결과 약 0.88의 성능을 나타내었다.

```
# evaluation sts-test
test_evaluator(model, output_path=sts_model_save_path)

2022-05-02 06:44:39 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-05-02 06:44:42 - Cosine-Similarity : Pearson: 0.8845 Spearman: 0.8839
2022-05-02 06:44:42 - Manhattan-Distance: Pearson: 0.8852 Spearman: 0.8812
2022-05-02 06:44:42 - Euclidean-Distance: Pearson: 0.8864 Spearman: 0.8825
2022-05-02 06:44:42 - Dot-Product-Similarity: Pearson: 0.8748 Spearman: 0.8705
0.8838723786473407
```

[그림 12] 기존 KLUE-STS Dataset 성능 평가

3.5 데이터 증강

본 논문에서는 크게 3어절 데이터와 4어절 이상의 데이터로 나눠서 데이터 증강을 하였다. 데이터를 증강시키는 문장들은 KorNLU-STS 데이터셋에서 선별하여 사용하였다. 3어절로 되어 있는 문장들은 한국어 특성상 무작위로 어순을 변경하여도 의미가 달라지지 않기 때문에 입력 문장을 랜덤하게 섞어주는 알고리즘을 사용하였다. 하지만 4어절 이상의 문장들은 무작위로 단어 순서를 바꿨을 시, 문장의 의미가 완전히 달라지는 문제점이 생겼다. 그리하여 4어절 이상의 데이터셋들은 어순을 선택해서 바꾸어 주는 알고리즘을 사용하였다. 또한 Negative Sample을 늘리기 위해 기존의 KLUE-STS 데이터셋에서 의미가 완전히 다른 문장들을 이용하여 Label 0인 데이터들을 제작한다.

3.5.1 3어절 데이터셋 증강

3어절로 되어 있는 문장들은 한국어 특성상 무작위로 어순을 변경하여도 의미가 달라지지 않기 때문에 입력 문장을 랜덤하게 섞어주는 알고리즘을 사용하였다. 기존의 KLUE-STS 데이터셋에 추가로 3어절의 문장을 가진 데이터셋만 증강시켜 학습을 시켰을 때, 아래와 같은 결과가 나왔다.

```
In [13]: # evaluation sts-test
test_evaluator(model, output_path=sts_model_save_path)

2022-05-31 04:55:09 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-05-31 04:55:27 - Cosine-Similarity : Pearson: 0.9073 Spearman: 0.9100
2022-05-31 04:55:28 - Manhattan-Distance: Pearson: 0.8991 Spearman: 0.9077
2022-05-31 04:55:28 - Euclidean-Distance: Pearson: 0.8989 Spearman: 0.9077
2022-05-31 04:55:28 - Dot-Product-Similarity: Pearson: 0.8953 Spearman: 0.8855

Out [13]: 0.9100403491201536
```

[그림 13] KLUE-STS(3어절 AUG) Dataset 성능 평가

3.5.2 4어절 이상 데이터셋 증강

4어절 이상의 문장들은 무작위로 단어 순서를 바꿨을 시, 문장의 의미가 완전히 달라지는 문제점이 생겼다. 그리하여 4어절 이상의 데이터셋들은 어순을 선택해서 바꾸어 주는 알고리즘을 사용하였다. 기존의 KLUE-STs 데이터셋에 3어절 이상의 데이터셋을 증강시켜 학습을 시켰을 때, 아래 [그림 20]과 같은 결과가 나왔다.

2022-06-30 08:58:52 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-06-30 08:58:54 - Cosine-Similarity : Pearson: 0.9294 Spearman: 0.9242
2022-06-30 08:58:54 - Manhattan-Distance: Pearson: 0.9178 Spearman: 0.9237
2022-06-30 08:58:54 - Euclidean-Distance: Pearson: 0.9182 Spearman: 0.9239
2022-06-30 08:58:54 - Dot-Product-Similarity: Pearson: 0.9133 Spearman: 0.9116
0.92420274722389

[그림 14] KLUE-STs(4어절 이상의 AUG) Dataset 성능 평가 결과

3.5.3 데이터셋 예시

sentence1	sentence2	label	어절 수
개가 물속으로 뛰어든다	물속으로 뛰어든다 개가	5	3
개가 물속으로 뛰어든다	개가 뛰어든다 물속으로	5	3
개가 물속으로 뛰어든다	뛰어든다 개가 물속으로	5	3
가급적이면 에어컨보단 선풍기를 쓰자	쓰자 가급적이면 에어컨보단 선풍기를	5	4
가급적이면 에어컨보단 선풍기를 쓰자	선풍기를 쓰자 가급적이면 에어컨보단	5	4
저는 친구랑 가족에게 이숙소를 강력추천했습니다	강력추천했습니다 저는 친구랑 가족에게 이숙소를	5	5
평일엔 거실에서 들리는 호프집 소리는 괜찮아요	괜찮아요 평일엔 거실에서 들리는 호프집 소리는	5	6

[그림 15] KLUE-STs(3어절 이상의 AUG) Dataset 예시

3.5.4 NLI와 STS 데이터를 활용한 추가 학습

STS 단일 데이터로 학습하는 방법 외에 NLI로 학습 후, STS로 추가 학습하는 방법을 시도하였다.

학습 방법을 간단하게 설명하면,

1. NLI 데이터를 triplet 형태로 구성 (anchor, Positive, Negative)

2. 사전 학습된 BERT모델에 의해 각 입력 시퀀스를 임베딩 벡터로 변환
3. 변환된 임베딩 벡터들에 대해 Pooling 연산(일반적으로 Mean-pooling)을 수행하여 문장 임베딩 벡터로 변환
4. MNR loss를 objective function으로 NLI 데이터셋 Fine-Tuning
5. NLI를 통해 Fine-Tuning된 모델을 로드하여 STS 데이터로 추가 학습
6. 변환된 두 문장 임베딩 벡터를 cosine similarity를 통해 벡터의 유사도 값(-1 ~ 1) 계산

위와 같은 방법은 특히 BERT cross-encoder 방식의 큰 영향을 줄 수 있다. 본 논문에서는 NLI 데이터를 학습하는 과정에서 Softmax loss보다 MNR loss를 통해 학습시키는 것이 더 좋은 성능을 보여주었기 때문에 NLI 데이터를 학습시키는 과정에서 MNR loss를 사용하였다. 그 결과, 아래 [그림 17]와 같이 기존 KLUE-STS의 데이터에서 3어절만 추가한 데이터셋을 KLUE-NLI 데이터와 함께 사용했을 시, 약 0.916이라는 결과가 나왔다. 또한 기존 KLUE-STS의 데이터에서 3어절 이상의 데이터셋을 추가학습한 데이터를 KLUE-NLI 데이터와 함께 사용했을 시에는 [그림 18]와 같이 0.928이라는 결과가 도출되었다.

```
# evaluation continue-learning by data AUG
test_evaluator(model, output_path=sts_model_save_path)

2022-06-06 16:06:27 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-06-06 16:06:29 - Cosine-Similarity :      Pearson: 0.9124 Spearman: 0.9166
2022-06-06 16:06:29 - Manhattan-Distance:    Pearson: 0.9055 Spearman: 0.9136
2022-06-06 16:06:29 - Euclidean-Distance:    Pearson: 0.9059 Spearman: 0.9142
2022-06-06 16:06:29 - Dot-Product-Similarity: Pearson: 0.9020 Spearman: 0.9064
0.9165671360100026
```

[그림 16] KLUE NLI+STS(3어절 AUG) Dataset 성능 평가

```
# evaluation sts-test
test_evaluator(model, output_path=sts_model_save_path)

2022-06-30 09:23:20 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-06-30 09:23:24 - Cosine-Similarity :      Pearson: 0.9310 Spearman: 0.9287
2022-06-30 09:23:24 - Manhattan-Distance:      Pearson: 0.9196 Spearman: 0.9286
2022-06-30 09:23:24 - Euclidean-Distance:      Pearson: 0.9197 Spearman: 0.9287
2022-06-30 09:23:24 - Dot-Product-Similarity:   Pearson: 0.9073 Spearman: 0.9167
0.9286971155965851
```

[그림 17] KLUE NLI+STS(3어절 이상 AUG) Dataset 성능 평가

하지만 이는 단지 데이터셋의 수가 늘어나 정확도가 올라간 것인지, 데이터 증강을 통해 만들어진 Positive Sample과 Negative Sample의 데이터셋으로 인하여 정확도가 올라간 것인지 확인할 필요가 있다고 생각하였다. 그리하여 KLUE-STS에 KorNLU-STS에 있는 데이터셋들을 추가하여 데이터셋을 늘린 후, 학습을 시켜보았다. 그 결과 아래 [그림 19]와 같이 약 0.87이라는 결과가 나온 것처럼 단순히 데이터셋 수가 증가하여 정확도가 높아진 것이 아니라 오히려 학습에 혼란을 주어 정확도가 떨어진 것을 확인할 수 있었다.

```
In [13]: # evaluation sts-test
test_evaluator(model, output_path=sts_model_save_path)

2022-05-31 02:00:32 - EmbeddingSimilarityEvaluator: Evaluating the model on sts-test dataset:
2022-05-31 02:00:49 - Cosine-Similarity :      Pearson: 0.8718 Spearman: 0.8740
2022-05-31 02:00:49 - Manhattan-Distance:      Pearson: 0.8732 Spearman: 0.8731
2022-05-31 02:00:49 - Euclidean-Distance:      Pearson: 0.8741 Spearman: 0.8736
2022-05-31 02:00:49 - Dot-Product-Similarity:   Pearson: 0.8607 Spearman: 0.8597
Out [13]: 0.8739581319971839
```

[그림 18] KLUE-STS Dataset에 증강하지 않고 데이터 수만 늘린 경우

4. 결론

	Model	Pearson	Spearman
STS 데이터로만 학습	KorSBERT-STs	0.8845	0.8839
STS 데이터 + 3어절만 증강한 데이터 학습	KorSBERT-STs (3어절 AUG)	0.9073	0.9100
NLI 데이터 학습 후 STS데이터	KorSBERT-STs (NLI+STS)	0.8962	0.8964
NLI 데이터 학습 후 STS데이터 + 데이터 증강(3어절만) 추가 학습	KorSBERT-STs (NLI+STS(3어절 AUG))	0.9124	0.9166
STS데이터 + 3어절 이상 증강한 데이터 학습	KorSBERT-STs (3어절 이상 AUG)	0.9294	0.9242
NLI 데이터 학습 후 STS데이터 + 데이터 증강(3어절 이상) 추가 학습	KorSBERT-STs (NLI+STS(3어절 이상 AUG))	0.9310	0.9287

[그림 19] 문장 유사도 성능 평가

본 논문의 성능 평가는 Cosine, Euclidean, Manhattan 그리고 Dot Product 각각의 Pearson correlation과 Spearman's rank correlation의 평균을 구하여 모델의 선형성과 단조성을 측정하였다.

STS task의 기존 SOTA들은 문장 임베딩들을 유사도 점수에 복잡한 regression function을 이용해 mapping하는 방법을 써왔다. 이런 regression function들은 pair-wise(pair를 입력받음)하기에 문장 집합이 커지면 조합적 폭발이 발생했고, 모델이 scalable하지 못했다. 하지만 SBERT는 단순한 코사인 유사도를 사용해 두 문장 임베딩을 비교한다. Negative Manhattan/Negative Euclidean 거리 역시 유사도 측정으로 사용했다. 다만 코사인 유사도와 비슷한 결과를 보였다.

[그림 20]은 각각의 모델별로 위에서 언급한 Pearson과 Spearman 성능 평가를 이용하여 문장 유사도 성능 평가를 종합적으로 나타내었다. [그림 20]와 같이 NLI 데이터로 학습 후, KLUE-STs 데이터에 3어절 이상의 문장들을 증강시킨 데이터셋을 추가하여 학습한 모델이 Pearson은 약 0.9310, Spearman이 약 0.9287로 가장 높게 나온 것을 확인할 수 있다.

이로써 데이터 증강 기법을 사용하여 데이터셋을 증강시켜 학습했을 경우, 기존 데이터셋들로 학습했을 시보다 더 정확한 결과값들이 도출되는 것을 확인할 수 있었다.

한국어의 특징인 어순을 변경하여 데이터셋을 증강시킨 데이터셋을 만들고, 그 데이터셋으로 KLUE-Roberta-Base를 학습시킨 결과, 정확도가 향상되었다는 것을 확인할 수 있었다. 본 논문에서는 데이터 증강 기법 중 하나인 한국어 어순 변경을 선택하여 사용하였다. 추후 연구에서는 어

순변경 뿐만 아니라 단어 삽입, 삭제, 유의어 대체 등의 데이터 증강 방식으로 동일 문장을 다르게 변형하더라도 임베딩한 값이 최소가 되도록 훈련함으로써 유사 문장 인식율을 높일 수 있다.

문장 유사도 판별 성능을 향상시켜, FAQ(자주하는 질문), Q&A 등에 활용할 수 있다. FAQ와 Q&A 시스템에서 문장 유사도가 중요한 이유는 어떠한 문장이 들어왔을 때, 비슷한 문장인지를 확인한 후, FAQ를 참조할 수 있게 해준다. 참조하기 위해서는, 들어온 문장이 FAQ에 있는 유사한 문장인지를 빠르고 정확하게 파악해야 한다. 사실상 의미적으로 중복된 질의들을 기계가 인식할 수 있다면 검색 처리 과정의 많은 비용이 절감될 것으로 예상된다. 따라서 비슷한 의미를 가진 문장을 모아서 일정한 응답을 제공할 수 있다면 서비스의 효율성이 올라갈 것이다.

5. 참고문헌

- [1] 김희동 지인영, "문장임베딩 표현을 위한 대조학습", 봄 한국사회언어학회, 담화인지언어학회 공동 주최 학술대회, 2022.04
- [2] 김희동, 지인영, "대조학습을 사용한 문장 임베딩 기술 연구", 정보산업공학논문집 제 40집, 2022.08
- [3] 김봉민 박성배, "한국어 문장 유사도 측정을 위한 Sentence BERT", 한국정보과학회 학술발표 논문집, 2020.12
- [4] [Basic NLP] sentence-transformers 라이브러리를 활용한 SBERT 학습 방법, intrepidgeeks
- [5] DeCLUTR (Giorgi et al., 2020) is the first work to combine Contrastive Learning (CL) with MLM into pre-training. arXiv:2006.03659v4 [cs.CL] 27 2021.5
- [6] Wu et al.: CLEAR: Contrastive Learning for Sentence Representation 2020.12
- [7] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks", arXiv:1908.10084v1 2019.08
- [8] RoBERTa: A Robustly Optimized BERT Pretraining Approach, Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov, arXiv:1907.11692 [cs.CL], 2019.7

6. 그림 출처

[그림 1]: "BERT-Pre-training of Deep Bidirectional Transformers for Language Understanding", jekyll

<https://greeksharifa.github.io/nlp%28natural%20language%20processing%29%20/%20rnns/2019/08/23/BERT-Pre-training-of-Deep-Bidirectional-Transformers-for-Language-Understanding/>

[그림 2]: "The RoBERTa model architecture.", ResearchGate

<https://wikidocs.net/156176>

[그림 3]. [그림 4]: 김희동, 지인영, "대조학습을 사용한 문장 임베딩 기술 연구", 정보산업공학논문집 제 40집, 2022.08

[그림 5], [그림 6]: "DeCLUTR (Giorgi et al., 2020) is the first work to combine Contrastive Learning (CL) with MLM into pre-training". arXiv:2006.03659v4 [cs.CL] 27 2021.05

[그림 7], [그림 8]: "CLEAR: Contrastive Learning for Sentence Representation", Wu et al 2020.12

[그림 9]: "[논문 리뷰] SimCSE : Simple Contrastive Learning of Sentence Embeddings", lm_minjin

https://velog.io/@lm_minjin/%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-SimCSE-Simple-Contrastive-Learning-of-Sentence-Embeddings

[그림 10]: "[논문 뽐내기] EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks", devsaka

https://catsirup.github.io/ai/2020/04/21/nlp_data_argumentation.html