**연 구 용 역 보 고 서**

**페이크 뉴스 자동 검출 응용 서비스의 인공지능 표준 시스템 분석**

**수탁기관: 동의대학교 산학협력단**

한 국 전 자 통 신 연 구 원

**연 구 용 역 보 고 서**

**페이크 뉴스 자동검출 응용 서비스의 인공지능 표준 시스템 분석**

**수탁기관: 동의대학교 산학협력단**

한 국 전 자 통 신 연 구 원

**제 출 문**

**한국전자통신연구원장 귀하**

본 보고서를 “페이크 뉴스 자동검출 응용 서비스의 인공지능 표준 시스템 분석”의 최종 연구보고서로 제출합니다.

2020년 11월 07일

수 탁 기 관 : 동의대학교 산학협력단

수 탁 기 관 장 : 이 임건 (인)

연 구 책 임 자 : 김 근형

참 여 연 구 원 : 한 윤진

박 예진

이 승찬

**요 약 문**

1. **제 목**

페이크 뉴스 자동검출 응용 서비스의 인공지능 표준 시스템 분석

1. **연구의 목적 및 중요성**

**1) 연구의 목적**

**2) 연구의 필요성**

1. **연구의 내용 및 범위**
2. **연구 결과**
3. **활용에 대한 건의**



**목차**

**그림 목차**

**표 목차**

# 제 1 장 개요

## 제 1 절 인공지능 서비스 배경 및 현황

1. **서비스 활용 사례**
2. **시장/제품 분석**

## 제 2 절 신뢰성 기술

1. **정의**
2. **중요성, 필요성, 파급효과**
3. **국내외 현황**

# 제 2 장 인공지능 신뢰성 분석

## 제 1 절 인공지능 서비스 현황

(문제점, 해결 방안)

## 제 2 절 표준화 현황

1. **국외 표준화 현황**
2. **국내 표준화 현황**

# 제 3 장 인공지능 신뢰성 프레임워크

1. **서비스 현황 사례**

(참조구조 및 기능, 공통 인터페이스)

1. **인공지능 신뢰성 요구 사항**

(참조구조 및 기능, 공통 인터페이스)

# 제 4 장 설명 가능한 인공지능 기반 서비스

## 제 1 절 설명 가능한 리뷰의 감성 탐지 서비스

1. **단어 임베딩**

일반적으로 컴퓨터는 자연어처리에서 단어의 의미와 상관없이 각각을 개별적인 기호로 취급한다. 컴퓨터가 자연어를 처리하도록 하기 위해선 자연어를 계산 가능한 형식으로 변형시켜야 한다. 문장에서 총 단어의 수만큼을 벡터의 차원으로 두고, 단어마다 해당하는 차원에 1로 표시하여 단어 벡터로 생성하는 것을 one-hot encoding이라고 한다. 각각의 단어 벡터들을 모두 더한 것을 해당 문장의 벡터라고 할 수 있는데, 단어의 의미를 담지 않은 단어 벡터들의 합이므로 문장의 의미 또한 표현하는데 어려움이 있다.

[http://doc.mindscale.kr/km/unstructured/11.html, 한국어 임베딩(책)]

이를 해결하기 위해 단어의 의미를 좀 더 세밀한 차원에 벡터로 표현하려 한 것이 단어 임베딩(word embedding)이다. 단어 임베딩은 one-hot encoding과 다르게 각각의 단어 벡터에 의미를 포함시키므로 비슷한 의미의 단어가 있는 경우 벡터 상에서 가까운 곳에 위치하게 된다. 그러므로 단어끼리 의미 비교를 위해 임베딩된 단어 벡터간에 유사도 계산이 가능하다. 유사도가 높으면 의미가 비슷하다는 것을 알 수 있다. 단어 임베딩의 종류에는 LSA, Word2Vec, GloVe, FastText 등이 있는데 이와 같은 단어 임베딩은 단어마다 벡터 값이 고정되어 있기 때문에 단어가 사용된 문맥은 고려하지 않는다. 즉 같은 단어이지만 다른 맥락에서 사용되었을 때도 같은 벡터 값으로 출력되기 때문에 차이를 알 수 없다.

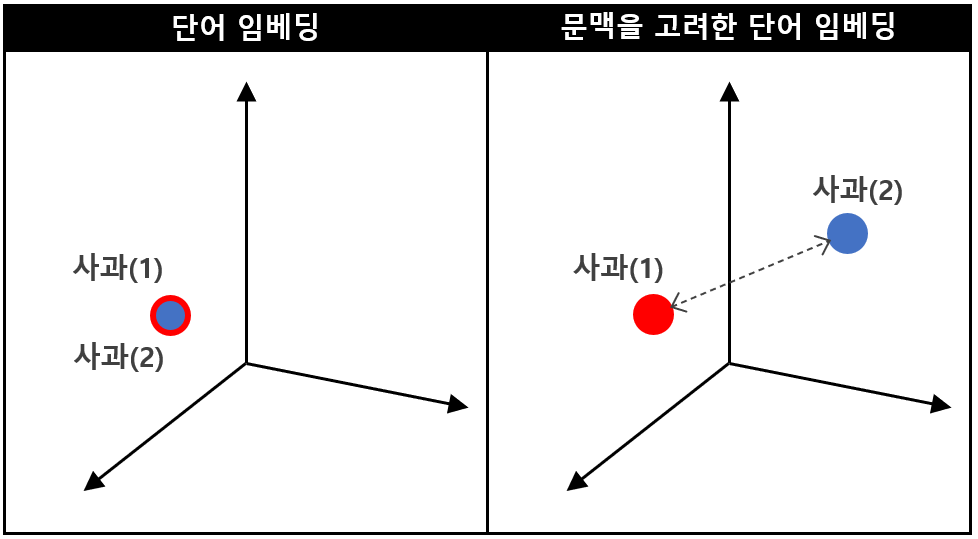


그림 1. 임베딩 방법 비교

단어가 사용된 문장의 맥락을 파악하기 위해 등장한 것이 문맥을 고려한 단어 임베딩(contextualized word embedding)이다. 문맥을 고려한 임베딩에선 개별 단어뿐만 아니라 문장 전체의 문맥적 의미를 함축하기 때문에 임베딩된 벡터를 인공지능을 통해 학습시켰을 때 기존의 단어 임베딩보다 학습 효과가 좋은 것으로 알려져있다. 그림 1은 단어 임베딩과 문맥을 반영한 단어 임베딩의 차이를 그림으로 나타낸 것이다. 단어 임베딩은 같은 문장에서 사용된 동일한 단어가 사용된 의미에 상관없이 같은 벡터를 출력하지만, 문맥을 고려한 임베딩의 경우 파악한 문맥에 맞게 각각의 벡터를 출력하게 된다. 문맥을 고려한 단어 임베딩 종류로는 BERT(Bidirectional Encoder Representtions from Transformer), ELMo(Embeddings from Language Models), GPT(Generative Pre-Training)등이 있다.

본 연구에서는 BERT가 사용되었다. BERT의 출력으로 얻을 수 있는 단어 벡터들은 문맥을 파악하여 생성된 벡터이기 때문에 단어의 의미뿐만 아니라 문장의 의미 또한 유지시킨다. 단어 벡터는 12개의 encoder layer를 거쳐 출력되는데, 마지막 레이어의 출력보다 마지막 4개 레이어 출력의 연결이 벡터의 표현을 더 잘 나타낼 수 있다. BERT는 출력 뒤에 원하는 분류 모델을 추가시켜 분류 학습을 하게 된다. 본 연구에선 설명가능한 인공지능을 위해 문장 벡터에 어떤 단어 벡터가 영향을 미쳤는지 해당 라벨로 분류되었는지 분석하였다.

1. **BERT(Bidirectional Encoder Representtions from Transformer)**

BERT는 인공지능 기반 자연어처리 분야에 등장한 구글의 강력한 언어모델이다. 기본적으로 Wikipedia 및 Book data와 같은 대용량 unlabeled data로 모델을 사전 학습(pre-training)하여 문맥을 파악할 수 있도록 구축된 모델이다. 사전학습된 모델 BERT에 분류하고자 하는 task에 맞게 labeled data로 추가 학습하여 분류 성능을 높인다. BERT는 사전학습에는 Masked Language Model(MLM)과 Next Sentence Prediction(NSP) 두 가지 방법이 사용된다. MLM은 랜덤한 위치의 토큰을 마스킹 시킨 뒤, 나머지 토큰들을 기반으로 마스킹된 토큰의 단어를 예측하고, NSP는 두 개의 문장을 입력하여 두 문장이 연속된 문장인지 예측하며 학습 성능을 높인다.

BERT Encoder는 MLM과 NSP를 위해 Encoder-Decoder 아키텍처로 이루어진 Transformer의 Transformer Encoder 구조를 기반으로 한다. Transformer는 RNN(Recurrent Neural Network)과 마찬가지로 시퀀스 데이터를 처리할 수 있도록 설계된 모델이다. 하지만 RNN과 다르게 Transformer는 시퀀스 데이터를 순차적으로 처리하지 않는다. 순차적으로 처리되는 RNN은 문장이 길어지는 경우 기울기 소실 문제(Vanishing gradient problem)로 인해 초기 토큰에 대한 정보가 사라질 수 있다는 단점이 있는 반면, Transformer는 병렬처리하여 초기 토큰의 정보를 보존할 수 있다. 또한 데이터 병렬처리 방법은 학습 시간을 단축시킨다.

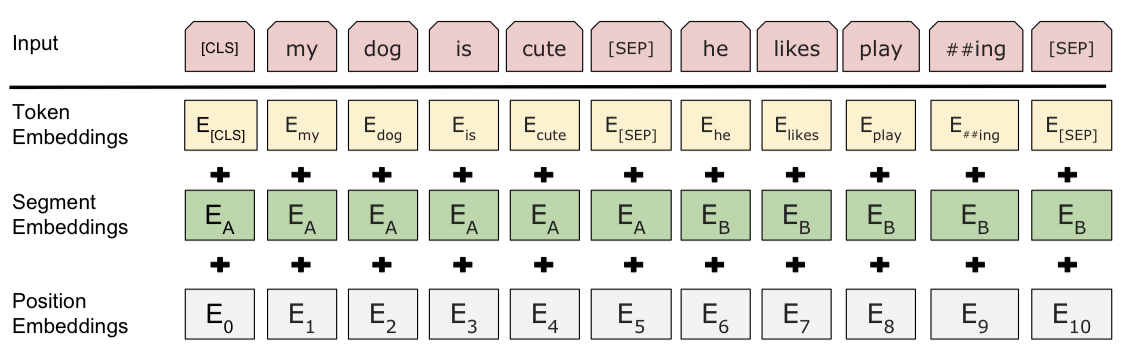


그림 2. BERT의 입력데이터 임베딩

BERT의 입력으로는 단어 토큰화된 토큰들이 사용된다. 그림 1의 Input에서 [CLS] 토큰은 BERT Encoder의 전체 layers를 거치고 난 후 token sequence의 결합된 의미를 가지게 되어 분류에 사용되는 토큰이며, [SEP] 토큰은 문장이 두 개 들어갈 때 문장을 구분하는 것에 사용되는 토큰이다. BERT Encoder에 입력하기 위해선 Embedding 과정을 거쳐야 하는데, Token embeddings과 position embeddings이 사용되는 Transformer Encoder과는 다르게, BERT Encoder에서는 토큰이 어느 문장에 속하는지 알 수 있도록 segment embeddings까지 더하여 입력 데이터를 생성한다.

BERT Encoder는 Multi-Head Attention이 사용되는데 이는 임베딩된 입력 데이터의 차원을 Head 수만큼 나누어 각각 Attention을 계산하고 추후에 합치는 과정이다. Attention 은 모든 토큰 간의 상관관계에 따라 가중치를 계산하는 메커니즘이다. 주어진 토큰 벡터 Query에 대해서 다른 토큰 Key와의 관계를 계산하여 가중치를 구하게 되는데 이를 Value에 적용시켜 출력값을 생성한다. 모든 토큰에 대한 attention 계산은 다음과 같이 최적화된 행렬 계산으로 표현될 수 있다.

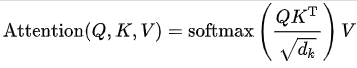


그림 3. Attention 계산 수식

설정한 layers 수만큼 BERT Encoder가 반복된 후 [CLS] 토큰은 token sequence의 결합된 의미를 가지게 되는데, 여기에 분류 모델을 추가되면 문장 분류 학습이 가능하게 된다. BERT는 다양한 task에 사용될 수 있으며, 입력 데이터와 분류 모델을 어떻게 Fine-tuning 하느냐에 따라 다른 유형으로 변형 가능하다.

표 1. BERT 클래스 종류

|  |  |
| --- | --- |
| **클래스** | **요약** |
| BertModel | 모든 모델의 틀  (사전 학습 완료) |
| BertForPreTraining | MLM과 NSP가 추가된 모델  (사전 학습 완료) |
| BertForMaskedLM | MLM가 추가된 모델  (사전 학습 완료) |
| BertForNextSentencePrediction | NSP가 추가된 모델  (사전 학습 완료) |
| BertForSequenceClassification | 문장 분류 모델이 추가된 모델  (사전 학습되었으나 분류 모델은 추가 훈련되어야 함) |
| BertForTokenClassification | 토큰 분류 모델이 추가된 모델  (사전 학습되었으나 분류 모델은 훈련되어야 함) |
| BertForQuestionAnswering | 질문에 맞는 토큰 분류 모델이 추가된 모델  (사전 학습되었으나 분류 모델은 훈련되어야 함) |

표 1은 BERT를 다양한 유형으로 사용할 수 있게 해주는 클래스 종류다. 모든 모델의 틀이 되는 BertModel을 기반으로 다양한 사전학습에 사용되는 MLM과 NSP 모델이 각각 추가된 클래스와 두 가지 모델 모두 추가된 클래스가 있다. 해당 클래스들은 모두 사전 학습되었으며 추가 학습시킬 필요가 없고, 이와 다르게 사전 학습이 되어있으나 추가 학습이 필요한 클래스들도 있다. BertForSequenceClassification 클래스는 문장을 label에 맞게 분류해주는 모델이 추가되어있기에, 분류 모델을 원하는 분류 유형에 맞는 데이터셋으로 학습시킬 필요가 있다. BertForTokenClassification 클래스 또한 문장의 토큰 각각을 label에 맞게 분류해주므로 추가 학습이 필요하다. 마지막으로 BertForQuestionAnswering 클래스는 Q&A를 위해 문장에서 정답을 찾아낼 수 있도록 추가 학습시켜야한다.

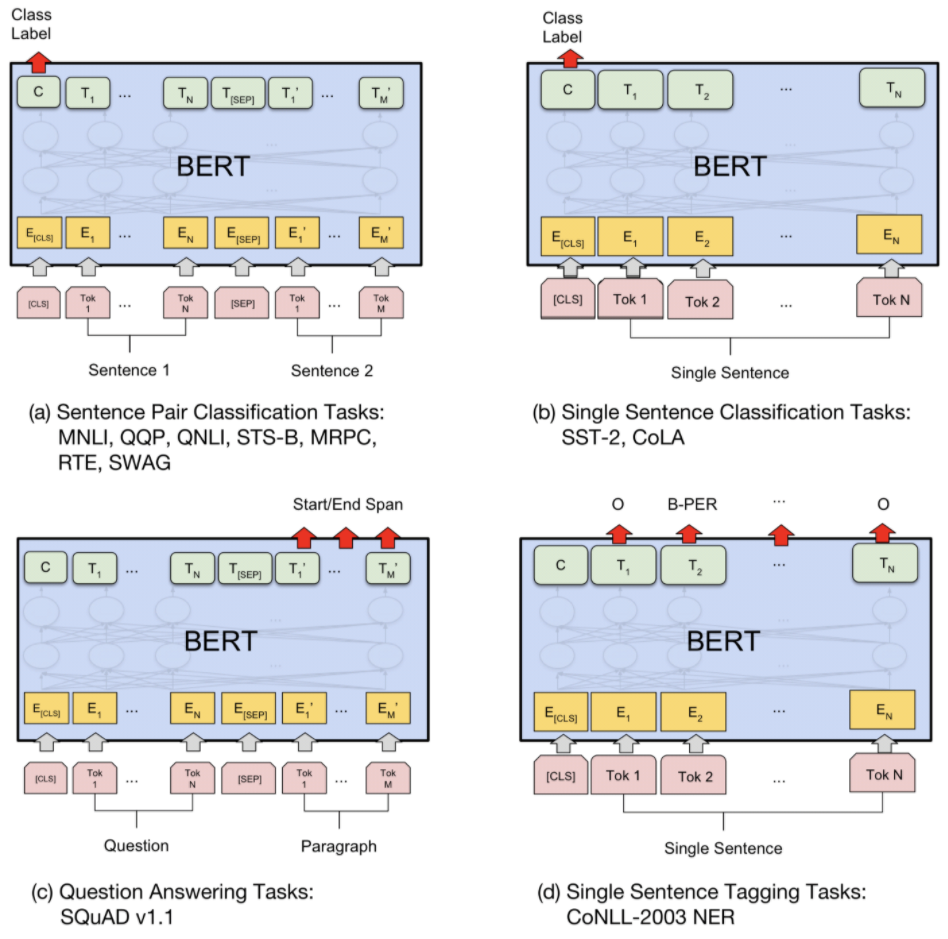


그림 4. BERT의 사용 유형

그림 3의 (a)와 (b)와 같은 경우는 BertForSequenceClassification 클래스를 생성하여 사용할 수 있는 유형으로, 분류하고 싶은 개수에 따라 분류 모델을 조정하고, 문장의 의미를 가지는 [CLS] 토큰의 벡터를 사용하여 class를 분류한다. (c)와 같은 Question Answering(QA) task 경우엔 BertForQuestionAnswering 클래스를 생성하여 사용할 수 있는 유형으로, Question와 Paragraph를 [SEP] 토큰으로 분류하고, Question에 정답을 Paragraph에서 찾아낸다. (d)와 같은 경우는 BertForTokenClassification 클래스를 생성하여 사용할 수 있는 유형으로, Named Entity Recognition(NER)이나 형태소 분석처럼 각 토큰이 어떤 class를 가지는지 각각을 분류할 수 있다.

BERT는 위의 내용처럼 downstream task 학습을 통해 사전학습된 parameter들을 변경하며 진행되는 Fine-tuning 접근법뿐만 아니라 Feature-based 접근법으로도 사용 가능하다. Feature-based 접근법은 BERT를 통해 진행된 Feature extraction을 특정 분류를 위한 추가 모델에 입력하여 학습시키는 형태이다. Feature-based 접근법은 다른 모델인 ELMo(Embeddings from Language Model)에서도 사용되는 접근법으로, 사용하는 이유는 Fine-tuning 접근법을 사용했을 때 사용되는 pooler는 분류 모델에 들어가기 전에 [CLS] 토큰을 처리하는 과정인데, pooler의 출력 값은 분류에는 도움을 주지만 문장의 의미를 표현하기엔 좋은 값이 아니기 때문이다.

* (This output is usually not a good summary of the semantic content of the input, you’re often better with averaging or pooling the sequence of hidden-states for the whole input sequence.) [출처]

표 2. BERT의 Feature-based 접근법

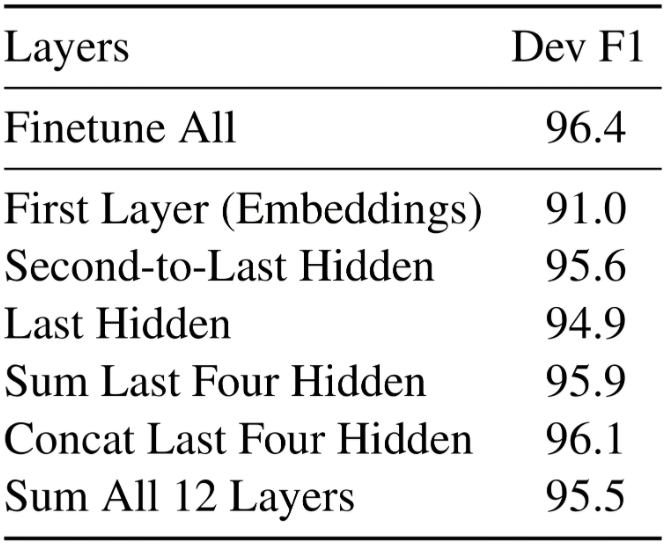


그림 4는 BERT 논문에서 Feature-based 접근법들을 사용하여 실험한 결과를 정리한 표다. 그림 4에서 볼 수 있듯이, Concat Last Four Hidden 값을 사용하면 Fine-tuning 접근법의 성능과의 차이가 크게 나지 않는다. 이를 보아, BERT는 Feature-based에서도 효과적이라고 볼 수 있다.

1. **BERT 세부 기능 분석**

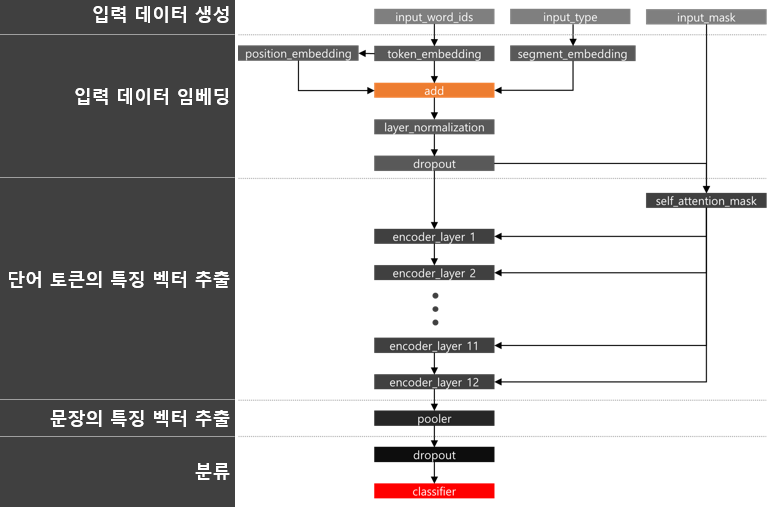


그림 5. BERT의 블록 다이어그램

그림 4는 BertForSequenceClassification 클래스를 생성하였을 때 전체 모델의 구조를 블록 다이어그램으로 표현한 것이다. 마지막 단계에 분류 모델을 추가하여 문장 분류가 이루어진다. BERT의 입력으로는 세 가지 데이터가 준비되어야 한다. 첫 번째로 입력 문장을 토큰화 한 뒤, 토큰을 index 값으로 변환시킨 input\_word\_ids 데이터가 필요하다. 두 번째로는 여러 문장이 입력으로 들어올 때 토큰이 어느 문장에 소속되어 있는지 알 수 있도록 input\_type 데이터가 있어야 하며, 마지막으로 토큰과 토큰이 아닌 패딩 위치를 알 수 있도록 해주는 input\_mask 데이터가 필요하다.

입력 받은 데이터들은 임베딩 과정을 거치게 되는데 input\_word\_ids 데이터로 token\_embedding 벡터와 토큰의 순서를 알 수 있는 position\_embedding 벡터를 얻게 된다. input\_type 데이터로 소속 문장을 알 수 있도록 해주는 segment\_embedding 벡터를 얻게 되며, input\_mask는 다음 단계인 encoder 단계에서 토큰과 패딩을 구분할 수 있도록 활용된다. 임베딩 과정을 통해 얻은 세 가지 벡터를 합하여 임베딩된 입력 데이터를 얻을 수 있다.

임베딩된 입력 데이터는 토큰의 특징 벡터를 추출할 수 있는 encoder로 입력된다. 이 과정에서 데이터는 Multi-Head Attention을 통해 각 단어 사이의 관계성에 맞게 가중치가 계산된다. 설정된 여러 layer를 거치게 되면서 입력된 문장의 문맥을 파악하게 되고 그에 맞는 특징 벡터를 출력한다.

encoder 단계에서 출력된 토큰의 특징 벡터들 중에서 첫 번째 토큰인 [CLS] 토큰은 다음 단계인 pooler의 입력으로 사용된다. pooler에서는 분류 모델이 분류를 더 효율적으로 진행할 수 있도록 문장 벡터인 [CLS] 토큰을 처리하는 과정이다. 분류 모델은 pooler의 결과인 문장 벡터를 입력으로 받아 결과에 맞는 label을 학습하게 된다.

1. **실험**

표 4. 리뷰 감정 탐지 서비스 실험 환경

|  |  |
| --- | --- |
| **프로그램** | Colab |
| **런타임** | GPU |
| **모델** | BertForSequenceClassification |
| **토크나이저** | BertTokenizer |
| **데이터셋** | Naver sentiment movie corpus |
| **데이터 수** | 학습 데이터: 150,000 |
| 테스트 데이터: 50,000 |
| **라벨** | 긍정: 100,000 |
| 부정: 100,000 |
| **테스트 정확도** | 87% |

실험을 위해 BERT 모델에 긍정 및 부정으로 이루어진 영화 리뷰 데이터를 학습시켰으며, 각 라벨로 분류될 때 어떤 단어의 영향이 컸는지 테스트했다. 임베딩된 벡터는 사칙 연산을 통해 단어와 문장 사이의 의미적 관계를 도출할 수 있다. 이를 기반으로 우리는 문장 벡터와 단어 벡터들 사이의 유사도를 측정하였을 때, 유사도가 높은 벡터가 문장에 영향을 크게 미쳤다고 판단했다. 표 3은 해당 테스트를 위한 실험 환경이다.

실험에서는 Transformers가 제공하는 클래스 중 BertForSequenceClassification를 사용했다. BertForSequenceClassification는 데이터 분류를 위해 단일 Linear layer가 추가된 BERT 모델이다. 클래스 생성 시 사전 학습된 BERT 모델과 추가된 분류 레이어가 입력 데이터에 대하여 특정 분류 작업을 학습한다. 사전 학습된 BERT 모델 중에서도 다양한 언어가 지원되는 bert-base-multilingual-cased 모델이 사용되었는데, 해당 모델은 Wikipedia 지분율이 높은 상위 104개의 언어로 학습되었기에 영어 이외의 언어로도 사용 가능하다. 사용 가능한 언어 목록에 한국어도 포함되어있어 해당 모델을 선택하였다.



그림6. BertTokenizer의 한국어 토큰화

생성된 모델에 데이터를 입력하기 전에 그림 5와 같이 문장을 토큰으로 분리하는 토큰화 과정이 필요하다. 토큰화에 사용될 토크나이저도 모델과 마찬가지로 Transformers의 사전 학습된 토크나이저인 BertTokenizer가 사용되었다. BertTokenizer는 WordPiece를 기반의 토크나이저로, WordPiece란 서브워드 토큰화(Subword Tokenization) 기반의 토크나이즈 알고리즘이다. 이는 자주 등장하여 이미 학습되어 있는 단어는 두고, 자주 등장하지 않는 단어를 더 작은 단위로 분리하여 의미 있는 단어로 토큰화하는 방법이다. 예를 들어 'ab'라는 단어가 있을 때 'ab'가 등장할 확률보다 'a'와 'b'로 등장할 확률이 높다면 더 작게 분리하는 것이다. 영어에 맞는 형태소 분석이 사용되지 않고, 분리된 토큰 사이에 '##' 기호로 연결을 표시하여 한국어도 처리가 가능하다. BertTokenizer를 사용하여 토큰화된 토큰들은 index 값으로 변환하여 모델에 입력으로 들어가게 된다.

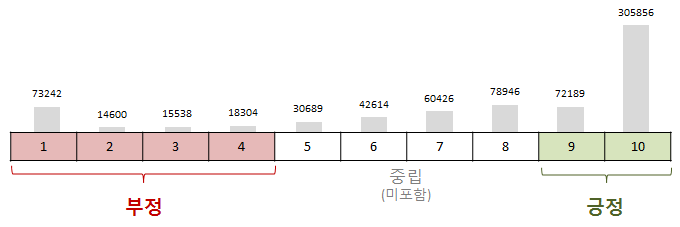


그림 7. 네이버 영화 리뷰 데이터 수집

실험 데이터로는 한국어 데이터 텍스트 분류 테스트에 많이 사용되는 네이버 영화 리뷰 데이터 nsmc(Naver sentiment movie corpus)가 사용되었다. 학습 데이터 150,000개와 테스트 데이터 50,000개를 합쳐 총 200,000개로 구성된 데이터로써 문장이 긍정일 때 label이 1, 부정일 때 label이 0으로 표시되어 있다. 네이버 영화의 리뷰 중 영화당 100개의 리뷰를 수집하였으며, 그림과 같이 1~10점까지의 평점 중에서 중립적인 평점인 5~8점 리뷰는 제외하고 1~4점은 부정으로, 9~10점은 긍정으로 포함시켰다. 긍정, 부정 label는 동일한 비율로 각 100,000개 수집되었다.

모델과 토크나이저, 데이터셋이 준비되었으면 첫 번째로 데이터셋을 BertTokenizer을 사용하여 토큰화하는 것이 필요하다. 토큰화된 토큰들은 숫자로 된 index로 변환하여 input\_word\_ids 데이터를 생성하였다. BERT 모델은 입력 시퀀스의 길이 제한이 있기 때문에 최대 시퀀스 길이에 맞춰 길이를 토큰을 잘라내거나 모자란 부분을 0으로 패딩 처리하여야 한다. 이때 문장에서 토큰이 존재하는 부분과 의미없는 부분을 알려주는 input\_mask 데이터를 생성하였다. 학습 시 input\_mask 데이터가 입력되면 BERT 모델에서 패딩 부분은 학습하지 않아 속도를 향상시킬 수 있다. 전처리된 학습 데이터를 분리하여 10%는 검증 데이터로 활용되었다.

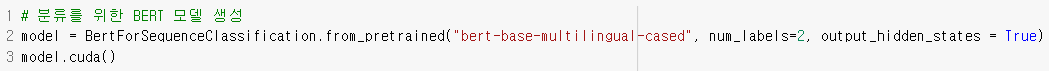


그림 8. 모델 생성

모델을 생성할 땐 그림과 같이 output\_hidden\_states를 True로 설정해야 BERT Encoder의 계산 값을 출력할 수 있다. optimizer로는 AdamW가 사용되었으며 epochs를 4로 설정하여 반복하였다. 생성된 모델에 전처리된 학습 데이터를 사용하여 학습시킨 후, 테스트 데이터도 위와 같은 전처리를 거쳐 모델을 테스트한 결과로 87%의 정확도를 얻을 수 있었다.

우리는 BERT Encoder를 거쳐 출력된 토큰 벡터와 문장 벡터를 보고 어떤 토큰이 문장에 큰 영향을 주었길래 분류 모델이 해당 label로 분류를 했는지 분석하고자 하였다. 첫 번째로 알 수 있었던 결과는 그림 8과 같이 BERT Encoder의 마지막 layer에서 출력된 토큰 벡터들을 히스토그램으로 그려보면 유사한 패턴을 가진다는 것이다.

뿐만 아니라 여러 문장으로 테스트했을 때도 같은 label로 분류되는 문장들 사이에 유사한 패턴이 존재한다는 것을 알 수 있었다. 그림 9의 첫 번째 히스토그램은 label이 부정으로 분류된 문장들의 문장 벡터들을 나타낸 것이며, 두 번째 히스토그램은 label이 긍정으로 분류된 문장들의 문장 벡터들을 나타낸 것이다. 각 10개(더 추가)의 문장으로 테스트하였으며, 문장들의 백터 값을 비교했을 때 유사도가 크게 높진 않았지만 각 분류 label에 따라 다른 패턴을 가진다는 것을 알 수 있었다. 각 label에 맞는 패턴이 있다는 근거를 기반으로 label 패턴에 가깝게 생성된 문장 벡터와 유사도가 높은 토큰 벡터가 label 분류에 영향을 미쳤다고 판단했다.

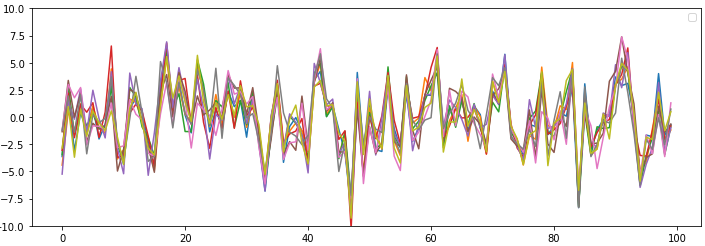


그림 9. 토큰 벡터 히스토그램



그림 11. 코사인 유사도 계산 코드

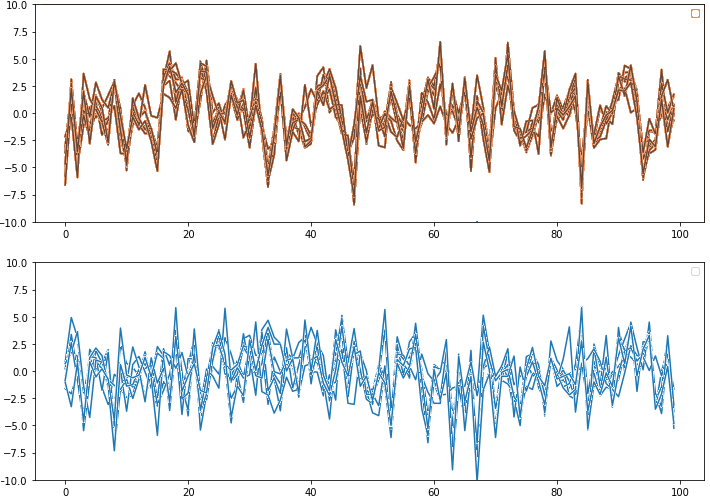


그림 10. 부정 및 긍정 문장 벡터 히스토그램

코사인 유사도를 사용하여 문장 벡터와 토큰 벡터 두 벡터 간의 유사도를 계산했다. BERT의 출력 벡터는 768차원을 가지게 된다. 두 벡터가 완전히 동일한 경우에는 유사도 결과는 1의 값을 가지며 두 벡터가 관련성이 없을수록 0에 가까워진다. 문장 벡터는 토큰 벡터의 합으로 이루어진 벡터이기 때문에 768차원을 구성하고 있는 정보들은 같다. 문장 벡터와 토큰 벡터들의 코사인 유사도를 계산하여 문장의 특징 벡터와 가장 유사한 토큰, 즉 단어는 어떤 것들이 있는지 분석했다.

그림은 코사인 유사도 테스트한 여러 문장 중 하나인 ‘재미없다 지루하고. 같은 음식 영화인데도 바베트의 만찬하고 넘 차이남....’ 이라는 문장의 결과를 히스토그램으로 나타낸 것이다. 각 토큰들과 코사인 유사도 계산을 한 [CLS] 토큰은 문장 벡터를 의미하며, 숫자 값은 유사도 결과를 나타낸 것이다. 해당 문장 벡터는 분류 모델을 거쳤을 때 부정으로 분류된 문장이다. 문장 벡터의 패턴은 그림 10의 첫 번째 히스토그램인 부정 패턴과 유사하게 출력된다. 부정 패턴에 가까운 문장 벡터와 각 토큰들의 코사인 유사도를 계산했을 때 결과를 보면, 토큰들 중에서도 유사도가 높은 부분이 존재한다는 것을 알 수 있다. 히스토그램을 네 등분하여 비교했을 때 첫 번째 부분과 나머지의 결과 값이 차이가 나는 것을 볼 수 있다. 첫 번째 부분의 토큰으로는 ‘재’, ‘##미’, ‘##없’, ‘##다’, ‘지’, ‘##루’, ‘##하고’, ‘.’ 등으로 부정의 의미를 가지는 단어들로 구성되어 있다. 나머지 부분의 내용을 보면 감정을 나타내기보단 타영화와 비교하는 내용으로써, 긍정인지 부정인지를 알 수 있는 감정적인 단어가 사용되지 않았음을 알 수 있다.

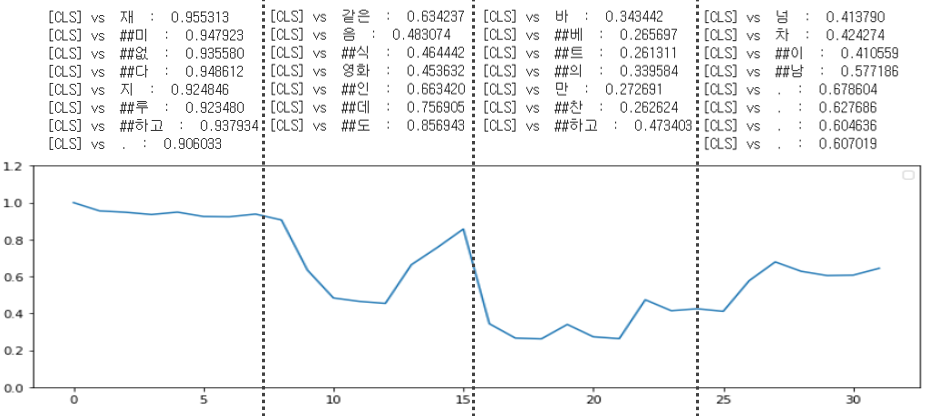


그림 12. 문장 벡터와 토큰 벡터의 코사인 유사도

우리는 50개의 문장으로 위와 같은 테스트를 반복하여 문장 벡터와 토큰 벡터의 코사인 유사도를 수동으로 확인했다. 히스토그램으로 강조되는 부분의 토큰이 감정을 나타내는 단어가 맞는지 확인하였으며, 긍정 및 부정으로 분류될 때 각 label의 문장에서 어떤 단어들이 강조되고 있는지 정리하였다. 50개의 문장을 테스트하였을 때 분류에 맞는 단어가 강조되고 있다고 판단되었다.

더 많은 문장에서 강조되는 단어가 어떤 것들이 있는지 분석하기 위해 for문을 사용하여 코사인 유사도가 높은 부분의 키워드를 출력하였다. 1000개의 문장에서 강조되는 단어를 출력하였다. BERT의 토큰화는 형태소 단위로 토큰화되지 않기 때문에 감정을 나타내는 명사를 정확하게 찾아낼 수는 없었으나 긍정으로 분류된 문장에서는 ‘좋다’, ‘깔끔해서’, ‘짱이다’, ‘최고’, ‘감동’ 등의 긍정적인 부분이 코사인 유사도가 높게 계산되었으며, 부정으로 분류된 문장에서는 ‘지루하다’, ‘재미없다’, ‘아깝다’, ‘실망’, ‘유치하다’ 등의 부정적인 부분이 코사인 유사도가 높게 계산되었다. 다음 표는 1000개의 문장에서 강조된 단어들의 통계를 나타낸 것이다. (통계내서 표 추가)

## 제 2 절 설명 가능한 가짜 뉴스 탐지 서비스

1. **가짜뉴스**

# 제 4장 결론

**참고문헌**

[1]