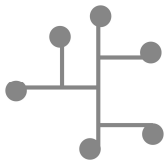
『데이타베이스연구』 논문지 접수 양식

논문 기본 정보	
논문 제목(한글)	단어 문맥 및 시퀀스에 대한 조인트 임베딩을 가지는 멀티모달 문서분류 딥러닝 아키텍처
논문 제목(영어)	Multimodal deep networks with joint embedding of word context and sequence for text classification
저 자 명	권순관[1], 김한준[2] SoonKwan Kwon, Han-joon Kim
저자 소속 및 직위	서울시립대학교 전자전기컴퓨터공학과, 석사과정[1] 서울시립대학교 전자전기컴퓨터공학과, 교수[2]
논문 세부분야	
교신저자 정보	
성 명	김한준
주 소	서울시 동대문구 서울시립대로 163 서울시립대학교 정보기술관 408호
연락처	+82-02-6490-2339
e-mail	khj@uos.ac.kr

※ 필요시 연락 가능한 메일주소 및 전화번호로 기재해주시길 바랍니다.



단어 문맥 및 시퀀스에 대한 조인트 임베딩을 가지는 멀티모달 문서분류 딥러닝 아키텍처

Multimodal deep networks with joint embedding of word context and sequence for text classification

요 약

최근 텍스트 데이터의 폭발적인 증가와 텍스트 생성모델의 발전으로 SNS 데이터 및 리뷰 데이터의 감성 분류 가짜 뉴스 탐지 등 자동문서분류(Text Classification)의 중요성이 더욱 부각되고 있다. 본 논문에서는 보다 정확한 문서분류를 위해 문맥을 고려한 단어 임베딩(Contextual Word Embedding) 기반 텐서공간모델(Tensor Space Model)과 Transformer 모델을 효과적으로 결합한 멀티모달 딥러닝 아키텍처를 제안한다.

문서분류 성능을 높이기 위해서 문서분류 아키텍처는 의미, 문맥, 순서 정보를 동시에 고려하여 학습되어야 한다. 텐서공간모델은 하나의 문서를 단어-개념(Term-by-Concept) 행렬로 표현하는 것인데, 이는 다의어(Polysemy) 문제와 단어 순서 정보를 고려하지 않았다. 우리는 완벽한 문서분류를 지향하기 위해 텐서공간모델에 단어의 의미 정보와 문맥 정보를 포함시키면서 동시에 단어 순서 정보를 학습할 수 있는 2-채널 분류 아키텍처를 고안했다. 순서 정보를 학습하기 위해 자연어처리 기법에서 널리 사용되는 Transformer 모델을 활용했으며 결과적으로 우리가 제안한 아키텍처는 일종의 멀티모달(Multimodal) 딥러닝 구조를 가진다. 우리는 6개의 영문 텍스트 데이터셋들을 활용하여 제안된 아키텍처의 성능 개선을 증명했다.

주제어: 딥러닝, 문서분류, 텐서공간모델, 임베딩, 멀티모달

Abstract

Recently, the explosive increase in the amount of text data and the rapid spread of text generative models has caused automatic text classification for information (e.g., social data, review data, and fake news) to become increasingly important. This paper proposes a multimodal deep learning architecture that effectively combines contextual word embedding and the Transformer model under the tensor space representation model to achieve more reliable text classification.

To improve text classification performance, the classification model must be trained to simultaneously consider semantic, context, and sequence information. The tensor space representation model represents a single document as a term-by-concept matrix that contains the semantic information of words; however, it does not accommodate the polysemy problem or word sequence information. To achieve near-perfect document classification, we propose a two-channel classification architecture that can learn all three: semantic, context, and sequence information of words under a tensor space model. In our approach, the Transformer model is utilized to learn word sequence information; as a result, our proposed architecture produces a multimodal learning model for text classification. Using six textual datasets, we demonstrate the performance improvement of our proposed multimodal text classification architecture.

Keywords: Deep Learning, Text classification, Tensor Space Model, Embedding, Multi-modal

1. 서 론

1.1 연구의 배경

최근 인터넷의 발전과 스마트폰의 보급으로 SNS의 사용이 급증하였고, 이로 인해 대량의 텍스트 데이터가 빠르게 생성되고 있다. 이런 데이터는 고객의 니즈를 파악하는데 매우 유용하나, 그 방대한 양으로 인해 수작업으로 분석하기에는 한계가 있다. 또한, 생성형 인공지능의 발전에 따라 가짜 뉴스와 같은 허위 정보가 확산되는 현상도 발생하고 있어, 이를 자동으로 판별하고 분류하는 기술의 필요성이 절실하게 느껴지고 있다. 이에 따라 인공신경망으로 학습한 임베딩을 활용해 자동문서분류 성능을 향상시키는 연구가 활발히 이루어지고 있다 [1-7]. 단어 수준의 임베딩 벡터는 단어의 의미를 반영한 벡터이므로 이것이 문서분류에 도움이 된다. 다의어의 경우나 하나의 단어가 문맥에 따라 근소하게 다른 역할을 하는 경우가 있다. 이러한 경우에도 상황 별로 달라지는 의미를 반영해 임베딩 해줄 수 있는 문맥 임베딩 기법 ELMo [8]와 BERT [9] 등이 등장했다.

문서분류 성능을 향상시키기 위해서 문서분류 아키텍처는 의미 정보, 문맥 정보, 순서 정보를 동시에 고려하여야 한다. 본 연구진의 이전 연구에서는 단어의 의미 정보를 반영하기 위해 문서를 문서-단어-개념 축으로 표현한 3차원 텐서공간모델 [10]을 제안했다. 본 연구는 이전 연구에서 나아가 의미, 문맥, 순서 정보를 모두 효과적으로 학습할 수 있는 2-채널 분류 아키텍처를 제안한다. 2-채널 분류 아키텍처는 텐서공간모델의 Concept 축을 문맥 임베딩인 ELMo 혹은 BERT 벡터로 구성하여 의미, 문맥 정보를 처리할 수 있는 Context 채널과 동시에 순서 정보를 처리할 수 있는 Transformer [11] 모델 기반 Sequence 채널을 포함한다.

본 논문의 기여는 다음과 같다. 문맥 임베딩 기법과 텐서공간모델을 결합하여 의미, 문맥 정보를 포함하는 텐서공간모델을 구성하였고 순서 정보를 동시에 반영할 수 있는 효과적인 분류 기법 2-채널 분류 아키텍처를 고안하였다. 제안 기법의 성능을 증명하기 위해 기존에 존재했던 머신러닝 모델과 2-채널 분류 아키텍처의 성능을 6개의 영문 텍스트 데이터셋으로 비교하였으며 2-채널 분류 아키텍처의 하이퍼파라미터를 최적화하는 과정을 제시하였다.

1.2 논문의 구성

본 논문의 구성은 다음과 같다. 2장에서 배경 연구들을 소개하고 관련 연구인 단어 임베딩을 사용한 문서분류 사례와 최신 문서분류 연구를 소개한다. 3장에서 구체적인 제안 기법의 상세를 서술하였으며 최적화 과정과 기존 머신러닝 모델과의 비교 실험 결과를 4장에서 확인할 수 있다. 마지막으로 5장에서 결론과 향후 연구에 대해 논하고 논문을 마무리한다.

2. 배경 및 관련 연구

2.1 배경연구

2.1.1 문맥을 고려한 임베딩 기법: ELMo, BERT

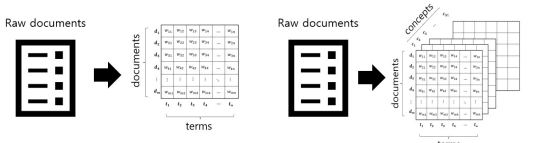
임베딩이란 비정형 데이터인 텍스트의 단어를 컴퓨터가 이해할 수 있도록 벡터 공간으로 표현하는 기법을 뜻한다. 인공신경망 구조로 학습된 임베딩 모델은 유사한 단어의 임베딩 벡터는 가깝게 유사하지 않은 단어는 먼 임베딩 벡터로 표현하도록 학습된다. 대규모 말뭉치(Copus)로 모델을 사전 학습하여 임베딩하는 기법 Word2Vec [12], GloVe [13] 등이 등장했고 이후 다의어와 문맥을 고려하도록

설계된 문맥 기반 임베딩 기법 ELMo, BERT가 등장했다.

ELMo는 대규모 말뭉치로 Bi-Direction LSTM 모델을 사전 학습하고 BERT는 Transformer 인코더 구조의 Transformer block을 쌓아 올린 모델을 사전 학습하여 문맥 정보를 학습할 수 있다. 두 모델 모두 문장을 입력으로 받아 문맥을 반영한 단어 수준의 임베딩 벡터를 출력할 수 있다.

2.1.2 텐서공간모델 (Tensor Space Model)

초기 텍스트 표현모델인 BoW(Bag of Words) [14]는 문서에서 등장한 단어의 빈도수를 활용해 2차원의 DTM(Document-Term Matrix)으로 표현했다. 같은 단어라도 단어가 등장한 문서에 따라 해당 단어의 중요도가 다르게 나타나므로 특정 문서 내 단어의 중요도에 따라 DTM에 가중치를 부여한 TF-IDF [14]가 등장했다. [10]의 저자는 2차원 표현모델에서 단어가 가지는 풍부한 의미 정보를 담기 위해서 Concept축이 추가된 새로운 3차원 표현모델 텐서공간모델을 제안했다. 초기의 텐서공간모델은 Concept 축에 단어의 의미 정보를 포함시키기 위해 Concept 벡터를 위키피디아 페이지로부터 정의했다. Concept 축의 각 요소는 위키피디아 페이지로 정의되고 단어의 역색인 (Inverted Index)을 통해서 Concept 벡터가 추출된다. 의미 정보가 포함된 텐서공간모델은 문서분류 성능을 개선했다.



2차원 표현모델 (BoW 모델) 3차원 표현모델 (텐서공간모델)

[그림 1] BoW 모델과 텐서공간모델의 차이

2.1.3 어텐션 메커니즘 (Attention Mechanism)

seq2seq [15]는 NLP task 중 기계어 번역을 위해 대표적으로 사용되는 인코더-디코더 구조의 모델이다. 이 모델의 단점은 RNN 기반으로 구성되어 입력 시퀀스의 길이가 길어지면 기울기 소실이 발생했고 인코더에서 하나의 벡터에 모든 정보가 압축되며 정보 손실이 발생했다. 어텐션 메커니즘 [16]은 이러한 단점들을 해결하기 위해 등장한 방법론으로 query, key, value 구조를 활용하여 디코더에서 출력 단어를 예측하는 매 시점마다 인코더의 입력 시퀀스를 참고하고 예측 단어와 더 관련 있는 입력 단어에 집중해 디코더의 예측 성능을 향상시켰다. 어텐션 메커니즘은 번역뿐만 아니라 컴퓨터 비전, 인공신경망을 활용하는 다양한 분야에서 적용될 수 있으며, [17, 18]에서는 어텐션 메커니즘을 활용해 문서분류 성능을 향상시켰다.

2.1.4 Transformer

Transformer는 seq2seq 모델과 마찬가지로 인코더-디코더 구조로 이루어져 있다. 기존 모델과 다른 점은 인코더와 디코더가 RNN 기반 모델이 아닌 포지션 임베딩과 Multi-head Self-Attention, Position wise FFNN(Feed-forward Neural Network) 등의 서브층으로 구성된다. 겹겹이 쌓인 인코더와 디코더 층에서는 각 시점의 연산들이 병렬적으로 수행되어 RNN 기반 모델과 비교해 속도가 빠르고 성능이 개선되었다. Transformer는 기계어 번역 task 뿐만 아니라 자동 문서분류 task 및 다양한 NLP task에서 활용되고 있다 [19-22].

2.2 관련 연구

단어 임베딩 기법이 발전하면서 이를 자동 문서분류에 활용한 다양한 연구들이 이루어졌다. [23]에

서 제안된 TextCNN 모델은 단어 임베딩 행렬을 합성곱 신경망(CNN)을 활용해 효과적으로 단어의 특징을 추출할 수 있었다. [1]에서 저자는 단어 임베딩 행렬과 더불어 품사 정보가 포함된 POS TAG Channel을 TextCNN 구조에 추가하여 SemiEval -2014, 2015, 2016 데이터셋에서 좋은 성능을 보여주는 향상된 TextCNN 기법을 제안했다. [5]의 저자는 감성 분석 성능을 높이기 위해 다양한 사전 학습된 임베딩 벡터로 Multi-Channel을 구성하고 CNN과 어텐션 매커니즘을 적용해 분류에 있어 더 안정적인 모델을 제안했다. [2]의 저자는 특정 단어의 중요도가 클래스에 따라 다르게 나타난다고 주장하였으며 TextCNN 구조에 TF-IDF 기반의 단어 가중치 시스템을 적용하여 문서분류 성능을 향상시키는 Multi-Channel TextCNN을 제안했다.

Graph Convolutional Networks (GCN)을 활용하여 자동 문서분류 성능을 높이려는 연구도 있었다. [24]의 저자는 TextGCN을 제안하여 문서분류 성능을 높였다. TextGCN은 문서와 단어를 그래프 노드로 표현하고 그에 대한 관계를 edge로 나타내는 방식으로 텍스트 데이터의 복잡한 구조를 그래프 형태로 표현했다. [25]에서는 TextGCN을 개선한 모델인 BertGCN을 제안했다. BERT와 GCN을 결합한 BertGCN은 사전 학습된 BERT을 활용해 문서를 노드로 표현하고 GCN을 학습하여 당시 여러 데이터셋에서 SOTA 성능을 달성했다.

최근에는 대규모 언어 모델 (Large Language Model, 이하 LLM) [26]이 발전하면서 프롬프트를 활용한 학습 방식인 In-context learning으로 문서분류를 효과적으로 수행할 수 있는 연구가 시도됐다. In-context learning이란 LLM의 프롬프트에 요구사항 및 예시 (Demonstration)를 입력하여 LLM이 수행하는 특정 task에 대한 성능을 개선하는 프롬프트 엔지니어링 (Prompt engineering) 기법이다. In-context learning은 미세조정 (Fine tuning) 기법과 비교하여 사전 학습된 모델로 학습한다는 공통점이 있으나 미세조정 방식과 다르게 사전 학습 모델의 가중치를 업데이트하지 않고 학습한다는 차이점이 있다. [27]에서는 In-context learning을 효과적으로 수행하는 Clue And Reasoning Prompting (CARP)를 제안했다. 논문의 저자는 문서분류 성능을 높이기 위해선 복잡한 언어 현상에 따른 강력한 추론 능력이 중요하다고 주장하였고 이를 위해 분류하고자 하는 문서에서 Clue와 Reasoning을 추출하여 분류 성능을 높일 수 있었다.

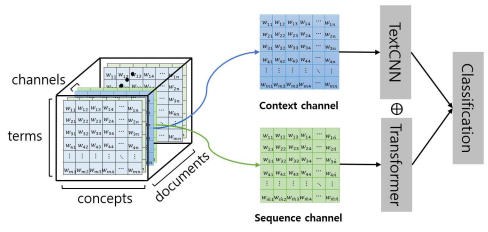
3. 제안 기법

3.1 개요

본 논문의 저자는 3차원 텐서공간모델에 의미 정보와 문맥 정보를 포함하기 위해 텐서공간모델의 Concept 축을 문맥 임베딩 벡터로 구성한다. 이렇게 구성된 텐서공간모델은 단일 채널로써 TextCNN 구조로 분류할 수 있는데, 본 논문에서는 이러한 분류 아키텍처를 1-채널 분류 아키텍처라 칭한다. 우리는 1-채널 분류 아키텍처에서 순서 정보가 담긴 새로운 채널을 추가함으로써 의미, 문맥, 순서 정보를 효과적으로 학습할 수 있는 일종의 멀티 모달 딥러닝 아키텍처를 제시하고 이를 2-채널 분류 아키텍처라 칭한다 (그림 2 참조). 제안 기법에 대해 상세하게 서술하기 앞서 본 논문에서 사용하는 용어를 정리하고자 한다.

본 논문에서 제안하는 2-채널 텐서공간모델은 문맥 기반 임베딩 행렬로 구성된 채널과 순서 정보가 담긴 Positional 임베딩 행렬로 구성된 채널을 포함한다. 문맥 기반 임베딩 행렬로 구성된 채널은 ELMo 또는 BERT 임베딩 벡터를 활용했고 본 논

문에서는 문맥 기반 임베딩 행렬을 Context 채널이라 칭한다. 그리고 Positional 임베딩 행렬로 구성된 또 다른 채널은 Sequence 채널이라 한다. 2-채널 분류 아키텍처의 Context 채널에서 ELMo를 활용한 경우 2-Channel Tensor(ELMo)로 표기하고 BERT를 활용했다면 2-Channel Tensor(BERT)로 표기한다. 마찬가지로 1-채널 분류 아키텍처는 활용한 임베딩에 따라 1-Channel Tensor(ELMo) 또는 1-Channel Tensor(BERT)로 표기했다.



[그림 2] 2-채널 분류 아키텍처 개념도

이어지는 3장 2절부터 3장 4절까지 2-채널 분류 아키텍처가 어떻게 구성되고 학습되는지 상세하게 서술한다.

3.2 Context 채널의 생성 과정

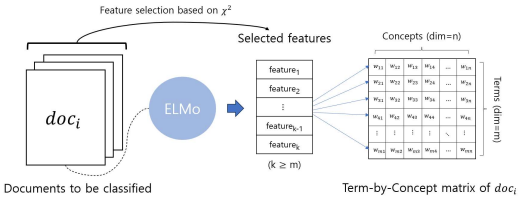
Context 채널을 구성하기 위해서 먼저 문맥 기반 임베딩 ELMo(혹은 BERT)를 사전 학습해야 한다. ELMo의 경우 분류할 데이터셋으로 학습하는데 계산 비용 절감을 위해 임베딩 벡터 출력이 256차원 되도록 설정해주었다. BERT의 경우 Hugging face에서 제공하는 transformer 패키지 내 사전 학습된 모델을 사용한다.

학습된 임베딩 모델로 분류할 데이터셋의 문서를 임베딩한다. ELMo를 활용하여 임베딩 한다면 문서에 등장하는 각 단어에 대한 임베딩 벡터를 얻을 수 있다. 예를 들어 입력인 문서가 'This is a multimodal deep learning architecture'라 가정하자. 이 문서가 사전 학습된 ELMo 임베딩의 입력으로 들어간다면 'This', 'is', 'a', 'multimodal', 'deep', 'learning', 'architecture'에 해당하는 7개의 256차원 벡터로 표현될 것이다. 분류할 데이터셋의 문서를 임베딩하여 행은 문서 내의 단어, 열은 임베딩 벡터인 단어 임베딩 행렬을 얻을 수 있다.

BERT로 문서를 임베딩하는 경우에는 문서에 포함된 각 단어가 아닌 서브워드 토큰에 대한 임베딩 벡터를 출력한다는 점에서 차이가 있다. 텐서 공간 모델의 개념 축에는 단어 수준의 임베딩 벡터가 필요하기 때문에 한 단어에 대한 서브워드 토큰 임베딩 벡터의 평균값이 단어 임베딩 벡터로 정의된다. 예를 들어, 문서 내에 'embeddings'라는 단어가 존재한다면 이 단어는 'em', '##bed', '##ding', '##s'라는 4개의 서브워드 토큰으로 분할되어 768차원 임베딩 벡터 4개가 생성된다. 이 네 개의 벡터를 평균하여 얻은 단일 768차원 벡터가 'embeddings'라는 단어의 임베딩 벡터로 정의하는 방식이다. 이렇게 BERT를 활용한 방식으로도 문서 내의 단어에 대한 단어 임베딩 행렬을 얻을 수 있다. 이 때 임베딩 벡터의 크기를 n으로 정의하여 ELMo의 경우에는 n이 256, BERT의 경우에는 n이 768이 된다. 다음에 이어지는 설명은 ELMo를 예시로 진행되지만, BERT의 경우에도 동일한 프로세스가 적용된다.

다음으로 Context channel의 Term 축을 구성할 분류에 도움이 되는 단어를 찾는다. 분류할 각 데이터셋에서 불용어를 제거하고 χ^2 통계량을 기반으로 문서분류에 도움이 되는 k개의 단어를 선별한다. 아래 이미지의 예시에서 k는 10,000으로 설정되어 10,000개의 단어를 선별하였다. 분류할 데이터셋의 문서 중 선별한 단어가 가장 많이 포함된 문서를 찾고 그 문서의 선별된 단어의 수를 m으로 정의한다.

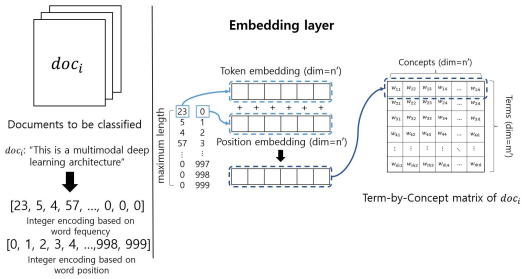
각 문서에 대해 행의 길이는 m, 열의 길이는 n이며 모든 값이 0으로 이루어진 초기 Context 채널이 되는 행렬을 생성한다 (그림 3 참조). 문서에 선별된 단어가 존재한다면 위에서 구했던 단어 임베딩 행렬에서 선별된 단어의 ELMo 임베딩 벡터를 찾아 초기 Context 채널 행렬의 첫 번째 행부터 차례대로 할당한다. 이렇게 구성된 Context 채널은 χ^2 통계량 기반 중요한 단어에 대한 의미 정보 및 문맥 정보를 포함한다.



[그림 3] 문서가 Context 채널로 변환되는 과정

3.3 Sequence 채널의 생성 과정

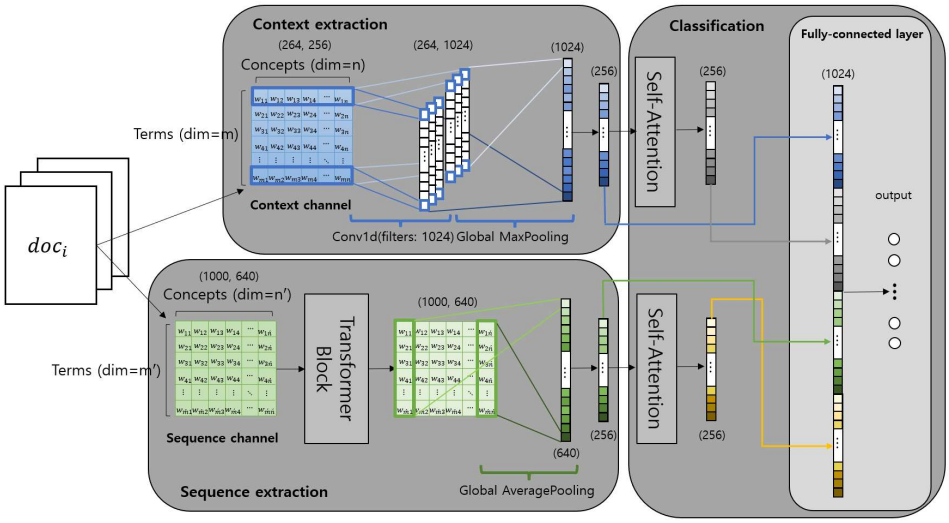
Sequence 채널의 구조는 Transformer 인코더의 Position 임베딩 행렬과 동일하다. 문서 내 단어의 배열을 단어 시퀀스라고 지칭하는데 이것이 모델의 입력으로 들어가 Sequence 채널로 변환된다. 모든 문서에 대한 단어 시퀀스의 길이를 동일하게 설정해주기 위해 입력 단어 시퀀스의 최대 길이를 설정하고 이를 m′으로 정의한다. 각 문서에서 나타나는 입력 단어 시퀀스는 정수 인코딩이 두 번 수행된다. 첫 번째 정수 인코딩에서는 단어의 출현 빈도에 따라 정수가 부여되고, 앞에서 설정한 m′에 따라 zero 패딩이 적용된다. 두 번째 정수 인코딩은 단어의 출현 위치에 따라 0부터 m′-1까지 순서대로 정수가 부여된다. 정수 인코딩된 단어 시퀀스는 임베딩 layer를 통해 각 단어 마두 개의 n′차원 임베딩 벡터로 변환된다. 여기에서 n′은 직접 설정할 수 있는 하이퍼파라미터이다. 두 개의 임베딩 벡터는 각각 Token 임베딩 벡터와 Position 임베딩 벡터라 칭하는데 이들은 학습 과정에서 업데이트 되는 non-static한 임베딩 벡터로 단어의 의미와 순서 정보를 학습하게 된다. Token 임베딩 벡터와 Position 임베딩 벡터의 합산 벡터가 하나의 단어에 해당하는 하나의 임베딩 벡터가 된다. 단어 시퀀스 내 모든 단어에 대한 임베딩 벡터들이 모여 임베딩 행렬이 되고 이 행렬이 바로 Sequence 채널이 된다 (그림 4 참조). Sequence 채널은 학습이 진행됨에 따라 문서 내 단어 시퀀스의 의미 정보와 순서 정보를 포함할 수 있다.



[그림 4] 문서가 Sequence 채널로 변환되는 과정

3.4 2-채널 분류 아키텍처

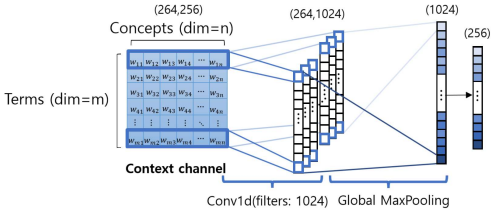
분류할 문서가 Context 채널과 Sequence 채널로 변환되면 각 채널은 2-채널 분류 아키텍처 내부 딥러닝 구조를 통해 학습된다. Context 채널은 TextCNN 구조로 학습되고 Sequence 채널은 Transformer 구조로 학습된다. 두 구조에서 학습되어 형성된 벡터를 feature map 벡터라고 부르는데, TextCNN과 Transformer에서 학습된 feature map 벡터들은 Self-Attention layer를 통과하여 중요도가 높은 요소에 가중치가 부여된 가중 feature map 벡터를 출력한다. Self-Attention layer를 통과하기 전과 후의 모든 feature map 벡터들은 하나



[그림 5] 2-채널 분류 아키텍처 상세 (IMDB 데이터셋 예시)

의 벡터로 결합 (Concatenate) 되고 fully-connected layer를 통해 최종적으로 분류된다. 그러나 특정 데이터셋(WELFake, 20News- groups, R8)에서는 Self-Attention layer를 적용하여도 성능이 향상되지 않으므로 이는 데이터셋에 따라 선택적으로 적용 여부를 결정한다.

그림 5는 4장에서 모델 성능 측정에 활용된 데이터셋 중 IMDB 데이터셋을 예시로 표현된 2-채널 분류 아키텍처의 상세가 묘사되었다. 그림에서 볼 수 있듯이 2-채널 분류 아키텍처는 3가지 영역(Context extraction, Sequence extraction, Classification)로 나뉘어 구성된다. 먼저 Context extraction 내부를 표현한 그림 6은 TextCNN 구조로 Context 채널이 학습되는 과정을 자세히 보여준다. Context 채널을 입력으로 합성곱 연산이 한 번 수행된다. 합성곱 연산에 사용되는 filter의 수는 1,024개이며, 각 filter의 shape는 (1,256)이다. 합성곱 신경망을 통해 생성된 feature map은 Global Max-pooling 연산을 통해 1,024차원의 벡터로 표현된다. 이 벡터는 fully-connected layer를 통과하여 의미 정보와 문맥 정보를 포함하는 256차원의 추상화된 feature map 벡터가 된다.

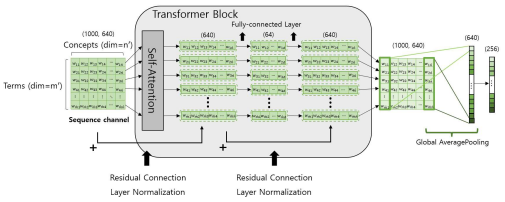


[그림 6] Context extraction 영역 내부 구조

그림 7은 Sequence extraction 내부에서 Transformer 인코더 구조를 사용해 Sequence 채널이 학습되는 과정을 자세히 보여준다. 입력으로 들어온 Sequence 채널은 하나의 Transformer block을 통과한다. Transformer block을 적층하지 않은 이유는 Transformer block을 적층하여도 유의미한 성능이 향상이 없어 계산 비용을 줄이기 위함이다. Transformer block 적층 실험은 4장에서

확인할 수 있다.

Sequence 채널의 각 시점의 임베딩 벡터가 Transformer block의 입력으로 들어오면 Multi-head Attention layer를 거쳐 가중치가 부여되는데 head 수는 1로 설정하여 Self-Attention과 동일한 연산이 수행된다. 또한 [11]에서 제안된 Transformer 모델과 마찬가지로 입력값과 출력값을 합산하는 residual connection이 적용돼 학습 과정에서의 정보 손실을 절감하고 layer normalization을 통해 일반화 성능을 높일 수 있도록 하였다. Transformer block 내 Self-Attention layer를 통과한 임베딩 벡터는 두 번의 fully-connected layer를 통해 차원이 축소된 후 원래의 차원으로 복원된다. 이 과정에서 다시 residual connection과 layer normalization이 순차적으로 수행된다. 앞선 과정으로 Transformer block을 통과한 Sequence 채널은 열 방향의 Global Average-pooling 연산으로 평탄화된다. 평탄화된 벡터는 앞서 설정했던 n′차원 임베딩 벡터 크기와 동일한 크기가 되고 fully-connected layer를 통과하여 의미 정보와 순서 정보를 포함하는 256차원의 추상화된 feature map 벡터가 된다.



[그림 7] Sequence extraction 영역 내부 구조

학습된 두 feature map 벡터는 모두 Self-Attention layer를 통과하면서 중요한 차원에 더 집중하여 가중치가 부여된 가중 feature map 벡터를 출력한다. Self-Attention layer를 통과하기 전 feature map 벡터와 가중 feature map 벡터는 모두 연결(concatenate)되어 1024차원의 벡터가 된다. 이는 fully-connected layer를 통해 예측 확률이 가장 높은 class로 분류된다.

4. 실험 방법 및 평가

4.1 실험에 활용된 데이터셋 소개

제안한 모델의 성능을 평가하기 위해 감성 분석 평가에 주로 쓰이는 IMDB 데이터셋과 가짜 뉴스 탐지를 위한 WELFake 데이터셋을 활용했다. 또한 4개의 영문 뉴스 데이터셋 (20Newsgroups, R8, News Category, AG news)을 사용하여 뉴스 토픽 분류 성능을 측정했다. 실험에 활용된 모든 데이터셋은 특수 문자 처리와 소문자 변환을 포함한 텍스트 전처리가 수행되었다. 그리고 각 데이터셋은 주어진 메모리 환경 내에서 원활하게 실험할 수 있도록 설정되었다. 샘플 수가 지나치게 많은 데이터셋의 경우 무작위로 샘플링하여 train 데이터와 test 데이터의 크기를 조절하였고 데이터셋의 클래스 불균형 문제가 심한 경우 해당 클래스를 제외하거나 균형을 조정해주었다. 결과적으로 Train 데이터와 Test 데이터의 비율은 약 80:20으로 나뉘었다. 실험에 활용된 데이터셋의 세부 사항은 표 1에서 확인할 수 있다.

[표 2] 실험 데이터셋 설명

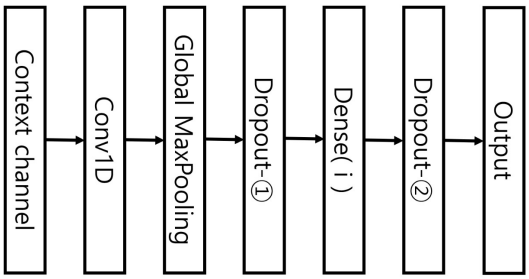
Dataset	Sample (original)	Class (original)	Train data	Test data	Class (experiment)
IMDB	50,000	2	16,000	4,000	2
WELFake	72,134	2	16,134	4,000	2
20Newsgroups	18,828	20	15,062	3,766	20
R8	7,673	8	5,484	2,189	8
News Category	209,527	42	8,000	2,000	10
AG News	127,600	4	16,000	4,000	4

4.2 2-채널 분류 아키텍처 문서분류 실험

제안 모델인 2-채널 분류 아키텍처와 기존 모델로 표 1의 데이터셋을 분류하여 정확도를 측정하였고 그에 따라 성능을 비교하였다.성능을 비교한 모델은 다음과 같고 제안 모델의 표기는 3.1을 따른다: 1-Channel Tensor(ELMo), 1-Channel Tensor (BERT), 1-Channel Tensor(ELMo), 1-Channel Tensor(BERT), Transformer classifier, 전통적인 머신러닝 알고리즘 (Support Vector Machine, Naïve Bayes, and Logistic Regression). 각 데이터셋과 모델마다 달라지는 하이퍼파라미터를 확인하기 위해 그림 8-10에 나타난 컨셉 다이어그램과 표 2을 참조하면 된다. 편의상 사용하는 함수명은 Python에서 사용하는 용어로 표현했다.

4.2.1 1-채널 분류 아키텍처 하이퍼파라미터 설정

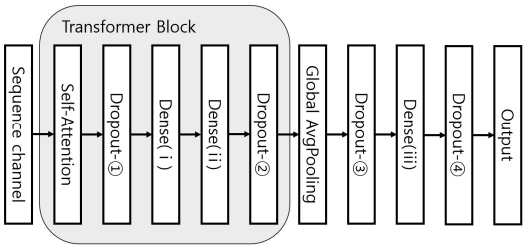
1-채널 분류 아키텍처는 Context 채널로만 구성된 텐서공간모델을 TextCNN 구조로 분류하는 모델이다. 이 모델의 구조는 그림 6에서 마지막 layer에 fully-connected layer를 추가한 형태로 구성되어 있으며, 이를 통해 출력을 해당 예측 클래스로 분류한다. 텐서공간모델의 Context 채널이 입력으로 들어오면 합성곱 연산을 수행하는데 filter의 크기는 (1, 임베딩 벡터 차원 수)이며, 각 데이터셋마다 filter의 수를 최적화하였다. Dropout-①,② layer의 Dropout 비율은 0 또는 0.5, Output layer의 활성 함수는 Softmax, 나머지 활성 함수는 ReLU, 배치 크기는 256으로 설정하였다. 자세한 데이터셋 별 하이퍼파라미터는 그림 8과 표 2를 참조하면 된다.



[그림 8] 1-채널 분류 아키텍처 컨셉 다이어그램

4.2.2 Transformer classifer 하이퍼파라미터 설정

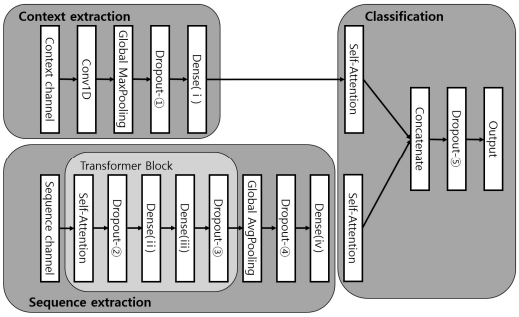
Transformer classifier는 Transformer 모델의 인코더 구조를 활용한 분류 모델이다. 이 모델의 구조는 그림 7에서 마지막 layer에 fully-connected layer를 추가한 형태로 구성되어 있으며, 이를 통해 출력을 해당 예측 클래스로 분류한다. Transformer block 내의 Dropout-①,② layer의 비율은 기존 Transformer 모델과 동일하게 0.1로 설정되었고, 나머지 Dropout-③,④ layer의 비율은 0 또는 0.5로 설정되었다. Sequence 채널의 임베딩 벡터 차원과 Dense(ⅰ), Dense(ⅲ)은 데이터셋 마다 최적화하였다. Dense(ⅱ)은 축소된 임베딩 벡터의 차원이 복원되는 지점으로 입력 임베딩 차원과 동일하게 설정한다. Output layer의 활성 함수는 Softmax, 나머지 활성 함수는 ReLU, 배치 크기는 256으로 설정하였다. 자세한 데이터셋 별 하이퍼파라미터는 그림 9와 표 2를 참조하면 된다.



[그림 9] Transformer classifier 컨셉 다이어그램

4.2.3 2-채널 분류 아키텍처 하이퍼파라미터 설정

제안 기법인 2-채널 분류 아키텍처에 적용된 합성곱 연산은 앞서 언급한 1-채널 분류 아키텍처와 동일하게 설정됐고 filter의 수는 데이터셋 별로 최적화되었다. 또한 각 데이터셋 별로 최적화된 하이퍼파라미터는 Sequence 채널의 임베딩 벡터 차원, Dense(ⅲ)를 제외한 Dense layer 노드 수, 학습된 feature map에 대한 Self-Attention 적용 여부가 있다. Dropout의 경우 앞에서 설명한 Transformer classifier과 동일하게 Transformer block 내의 비율은 0.1 그 외는 0 또는 0.5로 두었다. Output layer의 활성 함수는 Softmax, 나머지 활성 함수는 ReLU, 배치 크기는 256으로 설정하였다. 자세한 데이터셋 별 하이퍼파라미터는 그림 10과 표 2를 참조하면 된다.



[그림 10] 2-채널 분류 아키텍처 컨셉 다이어그램

4.2.4 하이퍼파라미터 최적화

딥러닝 모델의 최적 하이퍼파라미터를 찾기 위하여 변화를 줄 하이퍼파라미터에 우선순위를 정한다. 우선순위가 높은 하이퍼파라미터를 변화시키며 성능이 가장 좋은 설정을 고정하고 다음 순위의 하이퍼파라미터를 탐색하는 방식으로 최적화를 진행했다. 모든 조합을 고려하여 하이퍼파라미터를 탐색하기엔 시간과 비용이 너무 많이 들어 해당 방식을 채택했다. 제안 기법인 2-채널 분류 아키텍처는 먼저 Dropout-①,④,⑤에 대해 0 혹은 0.5으로 적용하는데 grid search를 통해 탐색하고 최적값을 고정했다. 다음으로 학습된 feature map 벡터의 결합 전 수행되는 Self-Attention layer에 대한 적용 유무에 따른 성능 변화를 살펴보았다. 나머지 연속적인 하이퍼파라미터는 2의 거듭제곱 값으로 변화를 주며 Position 임베딩 벡터 차원 수, Conv1D 필터 수,

[표 2] 딥러닝 모델에 대한 최적 하이퍼파라미터 표

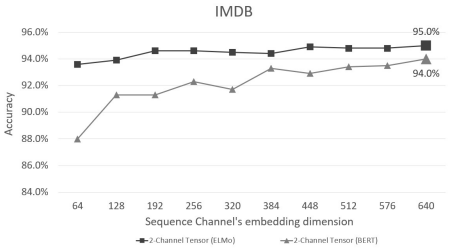
Dataset	Model	Dropout					Self-Attention	Position Embedding	Conv1D # of filter	Dense				Epoch
		①	②	③	④	⑤				(ⅰ)	(ⅱ)	(ⅲ)	(ⅳ)	
IMDB	A	0.5	0	-	-	-	-	-	1024	256	-	-	-	15
	B	0.5	0.5	-	-	-	-	-	1024	256	-	-	-	15
	C	0	0.1	0.1	0.5	0.5	True	640	1024	256	64	640	256	20
	D	0	0.1	0.1	0.5	0.5	True	640	1024	256	32	640	256	45
	E	0.1	0.1	0.5	0	-	-	64	-	32	64	256	-	5
WELFake	A	0.5	0.5	-	-	-	-	-	1024	256	-	-	-	20
	B	0.5	0.5	-	-	-	-	-	1024	256	-	-	-	20
	C	0	0.1	0.1	0.5	0	False	192	1024	256	128	192	256	10
	D	0	0.1	0.1	0.5	0	False	512	512	256	32	512	256	5
	E	0.1	0.1	0.5	0	-	-	128	-	128	128	256	-	5
20News-groups	A	0.5	0	-	-	-	-	-	256	256	-	-	-	50
	B	0.5	0	-	-	-	-	-	512	1024	-	-	-	45
	C	0	0.1	0.1	0.5	0.5	False	384	256	256	32	384	256	10
	D	0	0.1	0.1	0.5	0.5	False	512	1024	1024	32	512	56	10
	E	0.1	0.1	0.5	0.5	-	-	256	-	256	256	256	-	50
R8	A	0.5	0.5	-	-	-	-	-	512	256	-	-	-	35
	B	0.5	0.5	-	-	-	-	-	1024	1024	-	-	-	45
	C	0.5	0.1	0.1	0.5	0	False	256	256	256	32	256	256	15
	D	0	0.1	0.1	0.5	0.5	False	256	256	256	32	256	56	20
	E	0.1	0.1	0.5	0	-	-	256	-	64	256	256	-	25
News Category	A	0.5	0	-	-	-	-	-	1024	1024	-	-	-	20
	B	0.5	0	-	-	-	-	-	1024	1024	-	-	-	20
	C	0	0.1	0.1	0.5	0.5	True	640	1024	256	32	640	256	45
	D	0	0.1	0.1	0	0.5	True	768	1024	256	64	768	56	45
	E	0.1	0.1	0.5	0	-	-	512	-	64	512	512	-	20
AG News	A	0.5	0	-	-	-	-	-	256	256	-	-	-	50
	B	0.5	0	-	-	-	-	-	512	1024	-	-	-	45
	C	0	0.1	0.1	0.5	0.5	True	640	1024	256	32	640	256	30
	D	0	0.1	0.1	0.5	0.5	True	384	1024	1024	32	384	256	10
	E	0.1	0.1	0.5	0.5	-	-	256	-	32	256	256	-	50

※ Model 표기
A: 1-Channel Tensor(ELMo)
B: 1-Channel Tensor(BERT)
C: 2-Channel Tensor(ELMo)
D: 2-Channel Tensor(BERT)
E: Transformer classifier

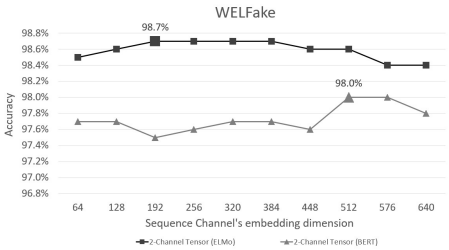
Dense 노드 수 순으로 탐색했다. Sequence 채널의 Position 임베딩 벡터 차원 수의 상세 최적 값을 찾기 위해 최적화를 추가로 진행했다. 앞서 찾은 다른 하이퍼파라미터 값을 고정하고 Position 임베딩 벡터의 차원 수를 128에서 최대 1280까지 128 간격으로 증가시키며 성능을 측정했다. IMDB, WELFake 그리고 20Newgroups 데이터셋의 경우 메모리 제한으로 인해 예외적으로 64부터 최대 640까지 증가시켜 최적값을 탐색했다 (간격은 64). 각 데이터셋별로 Position 임베딩 벡터의 차원 수 변화에 따른 성능 변화를 선 그래프로 표현했다. 선 그래프는 그림 11-16에서 확인할 수 있고 최종적으로 결정된 하이퍼파라미터는 표 2에 나타난다.

전통적인 머신러닝 알고리즘인 SVM, Naïve

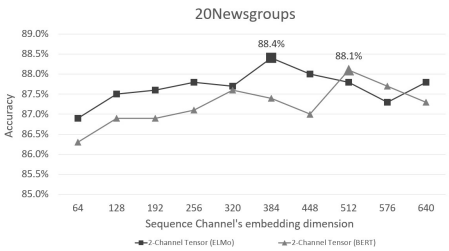
Bayes와 Logistic Regression에 대한 하이퍼파라미터 최적화는 grid search를 통해 수행되었다. SVM과 Logistic Regression은 C값에 대한 변화(0.1에서 1000까지 10의 거듭제곱씩 증가)와 규제에 대한 변화(L1, L2)를 주어 탐색하였다. Naïve Bayes의 경우는 alpha 값에 대한 변화(0.01에서 100까지 10의 거듭제곱씩 증가)를 주어 최적값을 탐색하였다.



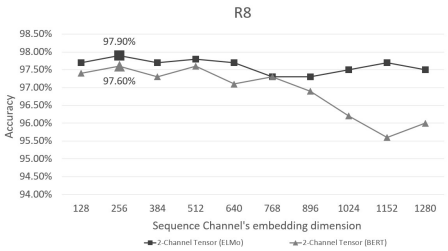
[그림 12] 최적 Position 임베딩 벡터 크기 탐색 (IMDB)



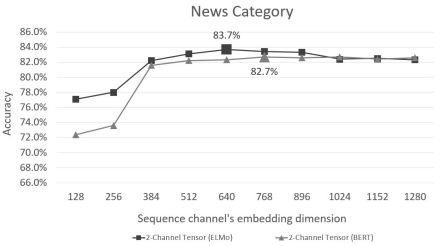
[그림 12] 최적 Position 임베딩 벡터 크기 탐색 (WELFake)



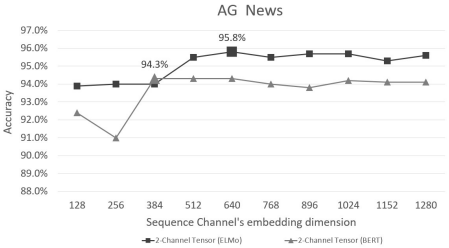
[그림 13] 최적 Position 임베딩 벡터 크기 탐색 (20Newsgroups)



[그림 15] 최적 Position 임베딩 벡터 크기 탐색 (R8)



[그림 16] 최적 Position 임베딩 벡터 크기 탐색 (News Category)



[그림 17] 최적 Position 임베딩 벡터 크기 탐색 (AG News)

4.3 실험 결과

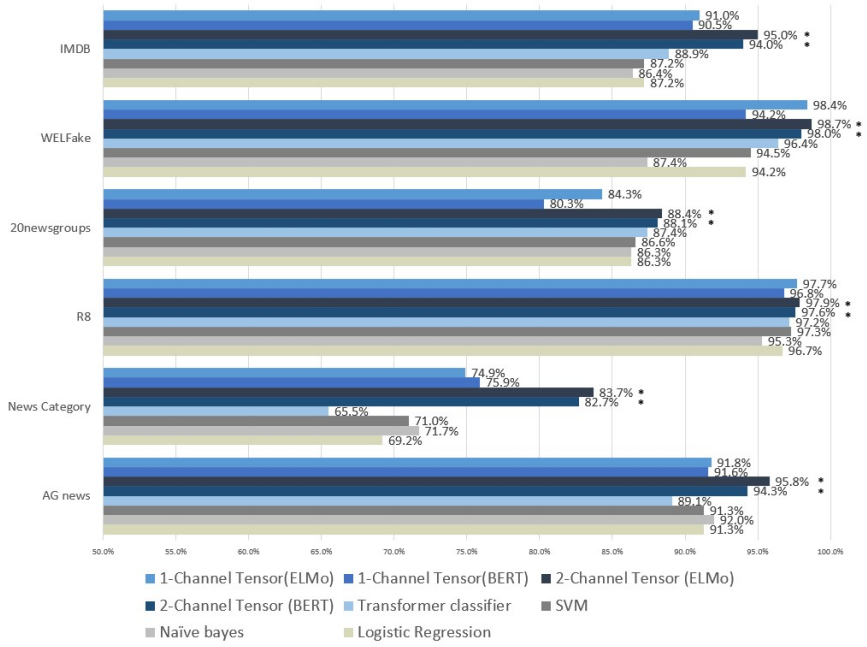
표 3과 그림 17에서 6개 영문 텍스트 데이터셋 대한 머신러닝 모델들의 분류 실험 결과를 확인할 수 있다. 각 데이터셋에서 성능이 가장 좋은 모델에

Bold 표기를 하였다. 제안한 모델인 2-채널 분류 아키텍처에 대한 Base line 모델을 1-채널 분류 아키텍처로 설정했을 때 6개의 데이터셋에서 모두 정확도 성능이 향상된 것을 확인할 수 있다. WELFake와 R8 데이터셋의 경우 1-채널 분류 아키텍처에 비해 2-채널 분류 아키텍처의 성능 개선이 미미했지만, 다른 데이터셋에서는 제안 모델의 성능 개선이 눈에 띄게 나타났다. 특히 IMDB, News Category, AG News 데이터셋에서는 2-채널 분류 아키텍처의 성능이 2 순위 모델과 비교해 상당한 차이를 보였다. 2-채널 분류 아키텍처과 2 순위 모델 간의 성능 격차는 IMDB 데이터셋에서 약 4%, AG News 데이터셋에서 약 3.8%, News Category 데이터셋에서 약 8.8%의 성능 개선이 있었다. 따라서 제안 기법이 1-채널 분류 아키텍처에서 처리할 수 없던 순서 정보를 보완했다고 볼 수 있다.

실험 결과를 보면 ELMo를 활용한 분류 모델이 BERT를 활용한 분류 모델보다 성능이 높게 측정되는 것을 확인할 수 있다. 이러한 결과는 텐서공간모델이 단어 수준의 임베딩 벡터를 요하기 때문으로 추측되는데 서브워드 토큰 수준 임베딩 벡터를 결합해 단어 수준 임베딩 벡터를 얻는 BERT 보다 단어 수준 임베딩 벡터를 바로 출력하는 ELMo가 텐서공간모델에 더 적합한 것으로 보인다.

[표 4] 6개 데이터셋에 대한 문서분류 실험 결과

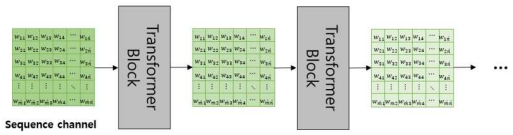
Dataset	Model	Accuracy(%)
IMDB	1-Channel Tensor(ELMo)	91.0%
	1-Channel Tensor(BERT)	90.5%
	2-Channel Tensor(ELMo)	95.0%
	2-Channel Tensor(BERT)	94.0%
	Transformer classifier	88.9%
	SVM	87.2%
	Naïve Bayes	86.4%
	Logistic Regression	87.2%
WELFake	1-Channel Tensor(ELMo)	98.4%
	1-Channel Tensor(BERT)	94.2%
	2-Channel Tensor(ELMo)	98.7%
	2-Channel Tensor(BERT)	98.0%
	Transformer classifier	96.4%
	SVM	94.5%
	Naïve Bayes	87.4%
	Logistic Regression	94.2%
20News-groups	1-Channel Tensor(ELMo)	84.3%
	1-Channel Tensor(BERT)	80.3%
	2-Channel Tensor(ELMo)	88.4%
	2-Channel Tensor(BERT)	88.1%
	Transformer classifier	87.4%
	SVM	86.6%
	Naïve Bayes	86.3%
	Logistic Regression	86.3%
R8	1-Channel Tensor(ELMo)	97.7%
	1-Channel Tensor(BERT)	96.8%
	2-Channel Tensor(ELMo)	97.9%
	2-Channel Tensor(BERT)	97.6%
	Transformer classifier	97.2%
	SVM	97.3%
	Naïve Bayes	95.3%
	Logistic Regression	96.7%
News Category	1-Channel Tensor(ELMo)	74.9%
	1-Channel Tensor(BERT)	75.9%
	2-Channel Tensor(ELMo)	83.7%
	2-Channel Tensor(BERT)	82.7%
	Transformer classifier	65.5%
	SVM	71.0%
	Naïve Bayes	71.7%
	Logistic Regression	69.2%
AG News	1-Channel Tensor(ELMo)	91.8%
	1-Channel Tensor(BERT)	91.6%
	2-Channel Tensor(ELMo)	95.8%
	2-Channel Tensor(BERT)	94.3%
	Transformer classifier	89.1%
	SVM	91.3%
	Naïve Bayes	92.0%
	Logistic Regression	91.3%



[그림 17] 6개 데이터셋에 대한 문서분류 실험 결과 (제안 기법 우측에 * 표기)

4.4 Transformer block 적층 실험

Transformer block의 입력이 되는 Sequence 채널과 출력의 형태는 동일하므로 2-채널 분류 아키텍처의 Transformer block을 겹겹이 쌓아 학습할 수 있다. 따라서 2-채널 분류 아키텍처의 Transformer block을 1개부터 6개까지 적층하여 성능이 개선되는지 실험을 수행하였다. 실험 환경의 메모리 제한에 따라 3개 데이터셋(R8, News Category, AG News)으로 Transformer 적층 실험을 수행하였다.



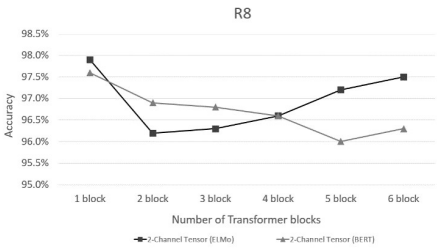
[그림 18] Transformer block 적층

[표 5] 6개 데이터셋에 대한 문서분류 실험 결과

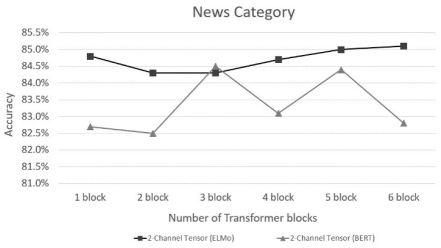
Dataset	Model	# of Transformer blocks					
		1	2	3	4	5	6
IMDB	A	97.9%	96.2%	96.3%	96.6%	97.2%	97.5%
	B	97.6%	96.9%	96.8%	96.6%	96.0%	96.3%
News Category	A	84.8%	84.3%	84.3%	84.7%	85.0%	85.1%
	B	82.7%	82.5%	84.5%	83.1%	84.4%	82.8%
AG News	A	95.8%	96.1%	96.1%	95.7%	95.8%	95.9%
	B	94.3%	94.3%	94.4%	94.3%	94.2%	94.5%

※ Model 표기
A: 2-Channel Tensor(ELMo)
B: 2-Channel Tensor(BERT)

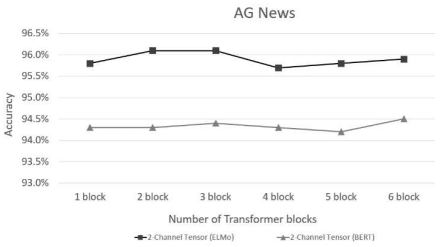
실험 결과는 표 4와 그림 19-21에서 확인할 수 있다. Transformer block을 적층해도 유의미한 성능 향상은 확인할 수 없었고 Transformer block의 수에 따라 계산 비용이 비례하여 증가하므로 2-채널 텐서공간모델은 하나의 Transformer block을 활용하는 것이 합리적이다.



[그림 20] Transformer block 수에 따른 정확도 (R8)



[그림 21] Transformer block 수에 따른 정확도 (News Category)



[그림 22] Transformer block 수에 따른 정확도 (AG News)

4.5 LLM 기반 문서분류 기법과의 성능 비교

본 논문에서 LLM을 활용한 실험은 포함하지 않는다. 다만 이 절에서는 4장 3절과 [27]에서 진행한 실험 결과를 바탕으로 제안 기법인 2-채널 분류 아키텍처와 fine-tuning 기법 모델 그리고 LLM을 활용한 In-context learning 기법 모델의 문서분류 성능을 간접적으로 비교하고 제안 기법의 효용성에 대해 논한다.

본 논문과 [27]에서 활용된 데이터셋 중 중복되는 데이터셋은 AG News와 R8 데이터셋으로 해당 데이터셋들이 비교의 대상이 된다. 표 5에서 두 데이터셋에 대한 각 모델의 정확도와 파라미터 크기를 확인할 수 있다.

파라미터 크기는 언어 모델을 사전 학습한 파라미터의 수와 분류 모델을 학습한 파라미터의 수를 합산한 결과이다. Fine-tuning 기법 RoBERTa-Large는 [28]에서 제안되었으며 학습된 파라미터의 수는 355M이다. DeBERTa은 [29]에서 제안되었고 학습된 해당 논문에서 나타나는 DeBERTa-base 모델의 파라미터 수는 134M이다. 문서분류 task를 위해 각 언어 모델을 fine-tuning 하는 과정에서 추가로 학습되는 파라미터 수는 α, β, γ, δ와 같이 표기한다. 마찬가지로 제안 기법의 파라미터 크기는 ELMo(256차원)의 파라미터 수 14M와 BERT의 파라미터 수 110M를 분류 모델의 학습 파라미터 수를 합산한 결과이다.

정확도를 기준으로 제안 기법보다 성능이 다소 높게 측정된 모델이 존재했으나 파라미터 크기를 기준으로 비교해 본다면 제안 기법이 더 우세하다. 특히, ELMo를 활용한 2-채널 분류 아키텍처의 경우 다른 모델들과 비교해 파라미터 크기가 매우 낮다. 따라서 제안 기법은 경량화된 모델로 준수한 성능을 보여준다고 할 수 있다.

[표 5] 최신 분류 기법과의 정확도 및 파라미터 크기 비교 (M: Million, B: Billion)

Dataset	Model		Accuracy(%)	Parameter Size
AG News	Fine-tuning method	RoBERTa-Large	95.6%	355M+α
		DeBERTa	95.3%	134M+β
		BERTGCN	95.7%	-
	In-context learning method	GPT-3: Vanilla	94.1%	175B
		GPT-3: CoT	94.9%	175B
		GPT-3: CARP	96.4%	175B
	Proposed method	2-Channel Tensor(ELMo)	95.8%	46M
		2-Channel Tensor(BERT)	94.3%	130M
R8	Fine-tuning method	RoBERTa-Large	97.8%	355M+γ
		DeBERTa	98.3%	134M+δ
		BERTGCN	98.2%	-
	In-context learning method	GPT-3: Vanilla	95.6%	175B
		GPT-3: CoT	95.6%	175B
		GPT-3: CARP	98.9%	175B
	Proposed method	2-Channel Tensor(ELMo)	97.9%	19M
		2-Channel Tensor(BERT)	97.6%	115M

5. 결론

문서분류 성능을 향상시키기 위해 분류 아키텍처는 의미, 문맥, 순서 정보를 모두 고려하여 학습되어야 한다. 본 논문은 문맥 임베딩 벡터를 활용하여 텐서공간모델에 의미, 문맥 정보를 포함시켰고 이것을 입력으로 순서 정보를 처리할 수 있는 2-채널 분류 아키텍처를 제안했다. 2-채널 분류 아키텍처는 문맥 임베딩 벡터로 구성된 Context 채널과 Transformer 기반 Position 임베딩 벡터로 구성된 Sequence 채널을 각각 학습시켜 의미, 문맥, 순서 정보를 효과적으로 처리하도록 고안됐다. 4장에서 6개의 영문 뉴스 데이터셋으로 2-채널 분류 아키텍처가 비교 모델보다 성능이 우수하다는 것을 실험적으로 입증하였다. 하이퍼파라미터를 최적화한 과정과 최적 하이퍼파라미터 표를 제시하였으며 이를 통해 논문을 읽는 독자가 제안 기법인 2-채널 분류 아키텍처를 쉽게 구현하여 감성 분석, 가짜 뉴스 탐지 등 다양한 문서분류에 활용할 수 있도록 하였다. 본 연구에서의 성능 향상이 실제 응용 환경에서도 기여할 수 있길 기대한다.

LLM이 발전됨에 따라 문서분류에 도움이 되는 키워드 추출과 이를 통한 추론 텍스트 생성이 가능하다. 본 논문에서 채널을 확장한 시도가 성공적으로 이루어졌기에 새로운 정보가 담긴 채널이 추가되고 이를 효과적으로 분류할 수 있다면 문서분류 성능은 더 향상될 수 있을 것이다. 향후 연구로는 LLM을 활용해 분류 대상 문서로부터 키워드를 추출하고 생성된 추론 텍스트로 새로운 채널을 구성하는 기법을 탐색하고자 한다.

참고 문헌

[1] A. Da'u and N. Salim, "Aspect Extraction on User Textual Reviews Using Mul-ti-Channel Convolutional Neural Network", *PeerJ Computer Science*, vol. 5, 2019, e191.

[2] B. Guo, C. Zhang and J. Liu, "Improving text classification with weighted word embeddings via a multi-channel TextCNN model", *Neurocomputing*, vol. 363, 2019, pp. 366-374.

[3] H. Peng et al., "Knowing What, How and Why: A Near Complete Solution for Aspect-Based Sentiment Analysis", *Proc. AAAI Conf. on Artificial Intelligence*, 2020, pp. 8600-8607.

[4] L. Maltoudoglou et al., "Well-Calibrated Confidence Measures for Multi-Label Text Classification with a Large Number of Labels", *Pattern Recognition*, vol. 122, 2022.

[5] J.W. Lin, T.D. Thanh and R.G. Chang, "Multi-Channel Word Embeddings for Sen-timent Analysis", *Soft computing*, vol. 26, 2022, pp. 12703-12715.

[6] A. Mueller et al., "Label Semantic Aware Pre-Training for Few-Shot Text Classification", *Proc. Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 8318-8334.

[7] J. Bird, A. Ekárt and D.R. Faria, "Chatbot Interaction with Artificial Intelligence: human data augmentation with T5 and language transformer ensemble for text classification", *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, 2023, pp. 3129-3144.

[8] M.E. Peters et al., "Deep Contextualized Word Representations", *Proc. 2018 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* 2018, vol. 1, pp. 2227-2237.

[9] J. Devlin et al., "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding", *Proc. 2019 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4171-4186.

[10] H.J. Kim, J.Y. Chang, "A Semantic Text Model with Wikipedia-Based Concept Space", *Journal of Society for e-Business Studies*, vol 19, 2014, pp. 107-123.

[11] A. Vaswani et al., "Attention is all you need", *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998-6008.

[12] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space", *arXiv*, 2013; doi: 10.48550/arXiv:1301.3781.

[13] J. Pennington, R. Socher and C.D. Manning, "GloVe: Global Vectors for Word Representation", *Proc. 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532-1543.

[14] B.Y. Ricardo and R.N. Berthier, *Modern Information Retrieval: The Concepts and Technology behind Search, 2/E*, Addison Wesley, 2011, pp. 62-77.

[15] I. Sutskever, O. Vinyals and Q.V. Le,

"Sequence to Sequence Learning with Neural Networks", *Advances in neural information processing systems*, 2014, vol. 27.

[16] D. Bahdanau, K.H. Cho and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate", *arXiv,* 2014; doi: 10.48550/arXiv:1409.0473.

[17] J. Xie et al., "Chinese Text Classification Based on Attention Mechanism and Feature-Enhanced Fusion Neural Network", *Computing*, vol. 102, 2020, pp. 683-700.

[18] G. Chrysostomou and N. Aletras, "Improving the Faithfulness of Attention-based Explanations with Task-specific Information for Text Classification", *arXiv,* 2021; doi: 10.48550/arXiv:2105.02657.

[19] W.C. Chang et al., "Taming Pretrained Transformers for Extreme Multi-Label Text Classification", *Proc. 26th ACM SIGKDD international Conf. on knowledge discovery & data mining*, 2020, pp. 3163-3171.

[20] T. Kano, S. Sakti and S. Nakamura, "Transformer-based direct speech-to-speech translation with transcoder", *in IEEE Spoken Language Technology Workshop*, 2021, pp. 958-965.

[21] J. Yang et al., "TABLEFORMER: Robust Transformer Modeling for Table-Text Encoding", *Proc. Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 528-537.

[22] G. Yu et al, "Dual-Branch Attention-in-Attention Transformer for Single-Channel Speech Enhancement", *in IEEE International Conf. on Acoustics, Speech and Signal Processing*, 2022, pp. 7847-7851.

[23] Y. Kim, "Convolutional Neural Networks for Sentence Classification", *Proc. 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746-1751.

[24] L. Yao, C. Mao and Y. Luo, "Graph Convolutional Networks for Text Classification", *Proc. AAAI Conf. on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 7370-7377.

[25] Y. Lin et al., "BertGCN: Transductive Text Classification by Combining GNN and BERT", *Findings of the Association for Computational Linguistics* (ACL-IJCNLP 2021), 2021, pp. 1456 −1462.

[26] W.X. Zhao et al., "A Survey of Large Language Models", *arXiv,*, 2023, doi: 10.48550/arXiv.2303.18223.

[27] X. Sun et al., "Text Classification via Large Language Models", *arXiv,*, 2023, doi: 10.48550/arXiv.2305.08377

[28] Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach", *arXiv*, 2019, doi: 10.48550/arXiv.1907.11692

[29] H. Pengcheng et al., "DeBERTa: Decoding-enhanced BERT with Disentangled Attention", *arXiv*, 2021, doi: 10.48550/arXiv.2006.03654