[UCI]I804:24011-200000362628

2021년 2월

석사학위 논문

딥러닝 기반 문장 중요도를 고려한 중심 문장 추출 방법

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 은 희

딥러닝 기반 문장 중요도를 고려한 중심 문장 추출 방법

Method of Extracting the Topic Sentence
Considering Sentence Importance based on Deep Learning

2021년 2월 25일

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 은 희

딥러닝 기반 문장 중요도를 고려한 중심 문장 추출 방법

지도교수 신 주 현

이 논문을 공학석사학위신청 논문으로 제출함.

2020년 10월

조선대학교 산업기술창업대학원

소프트웨어융합공학과

김 은 희

김은희의 석사학위논문을 인준함

위원장 조선대학교 교수 김 판 구 (인)

위 원 조선대학교 교수 반 성 범 (인)

위 원 조선대학교 교수 신 주 현 (인)

2020년 11월

조선대학교 산업기술창업대학원

목 차

표 목 차

그 림 목 차

ABSTRACT

Method of Extracting the Topic Sentence Considering Sentence Importance based on Deep Learning

EunHee Kim
Advisor : Prof. JuHyun Shin, Ph.D.
Department of Software Convergence Engineering
Graduate School of Industrial Technology and Entrepreneurship, Chosun University

As data increases exponentially in the era of the 4th industrial revolution, it is becoming more important to efficiently acquire important information from among a number of data. In particular, in the case of text data, since the length of the text data is long and important sentences and non-critical sentences are mixed in the document, it is difficult to grasp the topic. Therefore, a method of efficiently extracting only sentences of high importance in the document is required.

Methods of extracting sentences with high importance include a method of extracting important words and important sentences by calculating the frequency of words in a sentence, and a method of calculating a sentence score by calculating the similarity between sentences based on a graph to determine a ranking. Recently, various studies are being conducted to summarize documents using deep learning models.

In this paper, we propose a method of extracting the central sentence by defining the sentence with high importance as a topic sentence. The proposed method is to calculate the probability of the central sentence by using the pre-learned language models ELMo, BERT, and the LSTM model that can classify sentences by learning

the flow of context. We propose a method of extracting the central sentence by combining the probability of the central sentence and the other features that affect the importance of the sentence.

Ⅰ. 서론

A. 연구 배경 및 목적

4차 산업혁명 시대를 맞아 데이터가 기하급수적으로 증가함에 따라 수많은 데이터 중에서 중요도를 판단하여 자신에게 필요한 정보만을 효율적으로 습득하는 일이 더욱 중요해지고 있다. 그래서 이를 대신해주는 요약 서비스에 대한 수요가 증가하면서 서머리(summary) 콘텐츠 산업이 다양하게 발전하고 있다.

[그림 1-1] 주요 요약 콘텐츠 서비스[1]

특히 매일 새로운 뉴스가 쏟아지는 시사 분야에서 서머리 콘텐츠에 대한 수요가 높아 다양한 요약 서비스가 제공되고 있다. 대표적인 요약 서비스로는 네이버 뉴스의 본문 요약봇이 있다. 네이버 요약봇은 자동요약기술 아이리스(IRIS)를 기반으로 구현되어, 문장의 중요도를 분석하고, 문서가 작성된 형식에 따라 중요한 문장을 추출하여 요약 결과를 보여준다[2].

[그림 1-2] 네이버 본문 요약봇

문서 요약에 대한 연구는 과거부터 진행되어 왔으나, 문서 요약에 대한 필요성 증가 및 머신러닝 기술의 발전으로 문서 요약과 관련된 연구는 다양한 방법으로 진행되고 있고 이를 적용한 여러 가지 서비스들이 제안되고 있다. 문서 요약이란 제시된 문서에서 전달하고자 하는 내용을 포함한 짧은 문서인 요약문을 생성하는 과정으로, 요약문을 생성하는 방법에는 문서에서 중요 문장을 추출하여 사용하는 방법인 추출 요약(Extractive summarization) 방식과 문서의 내용을 기반으로 새로운 요약문을 생성하는 생성 요약(Abstractive summarization) 방식으로 구분할 수 있다.

하나의 문서 내에서는 중요한 문장과 그렇지 못한 문장의 구분이 있으며, 이러한 문장 중요도의 차이는 각각의 문장에 나타나는 특성에 따라 문장 중요도에도 영향을 미친다[3]. 본 논문에서는 중심 문장(Topic Sentence)을 문서 내 중요도가 높은 문장으로 정의하여 중심 문장을 추출하는 추출 요약 방법에 관한 연구를 진행하고자 한다.

문장의 중요도를 계산하여 요약문을 추출하는 대표적인 추출 요약 방법으로는 TextRank 모델이 있다. TextRank 모델은 문서를 그래프로 표현하고 각 문장의 중요도를 측정하기 위한 랭크 알고리즘을 제안하였다. 그러나 TextRank에서 사용한 문장 간 유사도 계산 방법은 문장 내 단어 간의 의미적 유사성을 충분히 고려하지 못한다는 문제점이 있다[4]. 이에 본 연구에서는 단어 간 의미적 유사성을 고려할 수 있는 워드 임베딩 모델인 ELMo를 적용하여 문장을 임베딩 하고자 한다.

또한 중심 문장이 일반 문장과는 다른 특징을 가질 것이라는 가설을 세우고, 딥러닝 방법으로 중심 문장과 일반 문장을 분류하는 학습을 통해 딥러닝 기반 분류 모델이 중심 문장의 특징을 학습했다고 가정하고, 이때 나온 중심 문장 예측값을 중심 문장의 특성으로 사용한다. 이와 함께, 문서 내에서 문장의 중요도에 영향을 주는 요인으로 문장의 위치에 따른 가중치, 문서 제목 및 다른 문장과의 유사도 결합하여 문장의 중요도를 계산하고 계산된 값의 순위에 따라 중심 문장을 추출하고자 한다.

영어의 경우 CNN/daily 말뭉치 등 대규모의 자원이 공개되어 있으나, 한국의 경우 네이버와 다음 등 포털사이트에서 제공하는 요약봇 서비스의 기반이 되는 말뭉치를 공개하지 않고 있다. 이에 따라 대규모의 고품질 한국어 요약 말뭉치를 확보하기가 어려웠다. 그러나 최근 국립국어원에서는 문서 요약 말뭉치를 공개하여 제공하고 있다. 이 문서 요약 말뭉치는 신문 기사 4,389건에서 추출한 중심 문장과 기사를 요약하여 작성한 문장(Summary)으로 구성되어 있다. 따라서 본 논문에서는 국립국어원의 문서 요약 말뭉치를 실험 데이터로 사용하고 추출된 요약문과 실제 요약문의 일치 정도를 평가 항목으로 사용하여 성능을 평가하고자 한다.

B. 연구 내용 및 구성

본 연구의 주요 내용은 중심 문장(Topic Sentence)의 특징을 딥러닝 기반 분류 모델로 판별하여 그 예측값을 중심 문장일 확률로 간주하고 이를 문장의 중요도에 영향을 미치는 특성인 제목 및 다른 문장과의 유사도, 문장 위치별 가중치와 결합하여 문서에서 중심 문장을 추출하는 방법을 제안하고자 한다. 본 논문의 구성은 다음과 같다.

본 장인 서론에 이어 2장 관련 연구에서는 대표적인 추출 요약 기법인 TextRank 알고리즘 및 본 연구에 사용된 ELMo, BERT 언어 모델 및 기존의 중심 문장 추출 연구에 대해 살펴봄으로써 본 연구 내용의 이해를 돕는다.

3장에서는 데이터 수집 및 텍스트 전처리 과정, 문장 중요도 계산에 사용할 특성을 추출하는 과정, LSTM 및 BERT 모델을 활용한 중심 문장 분류 학습 결과를 추출된 특성과 결합하여 문장별 중요도를 계산하는 과정 및 문장별 중요도에 따라 중심 문장을 추출 방법을 제시한다.

본 연구에 사용된 기술은 [표 1-1]과 같다.

사용 기술	사용용도
Konlpy	문서 토큰화 작업
ELMo	문장 임베딩에 사용되는 언어 모델
BERT	중심 문장 판별을 위한 사전 훈련된 언어 모델
LSTM	중심 문장 판별을 위한 딥러닝 모델
TextRank	비교 모델로 중심 문장을 추출하기 위한 알고리즘

[표 1-1] 사용 기술

4장에서는 본 연구에 사용된 실험 데이터의 수집 방법 및 데이터 셋에 대해 설명하고, LSTM 및 BERT 모델을 활용한 중심 문장 판별 학습 결과의 효율성을 평가하기 위해 분류성능평가지표인 정확도(Accuracy), 재현율(Recall), 정밀도(Precision)를 측정하여 분류성능을 평가하고, 제안한 방법에 대한 요약 성능은 ROUGE-1 스코어 및 실제 요약문 대비 추출 요약문의 추출 정도를 구하여 기존의 연구 방법과 비교하여 성능이 향상됨을 검증한다.

마지막으로 5장에서는 본 연구에 대해 전체적인 결과를 요약하고, 향후 연구를 제시하며 마무리한다.

Ⅱ. 관련 연구

A. 중심 문장의 개념

중심 내용이란 용어는 다양하게 정의된다. 중심 내용과 관련된 다양한 개념을 소개한 James F. Baumann는 '중심 내용의 이해와 수업(문선모 역, 1995)'에서 산문에서 중요한 정보를 나타내기 위해 사용되는 9개의 상이한 용어들에 대한 개념 및 정의들에 관련된 혼란을 명료화하려고 시도하여 9개의 중심 내용 반응유형을 확인하고 정리하였다[5]. 그중 본 연구와 관련된 내용은 다음과 같다.

- ○ 핵심 단어(Key Word) : 문서에서 가장 중요한 단일 개념을 일컫는 한 단어 또는 용어
- ○ 선택적 요약(Selective Summary) : 문서로부터 가장 상위의 그리고 중요한 단어들 및 구들을 선택, 결합함으로써 이루어지는 문서의 명백한 내용의 요약
- ○ 토픽(Topic) : 문서로부터 특정 내용을 드러내지 않고 문서의 주제를 일컫는 구
- ○ 주제문(Topic Sentence) : 전체로서의 단락 또는 문서가 무엇을 진술하는지를 가장 완전하게 알리는 단락 또는 문서 내의 단일 문장

위 내용 중 본 논문에서 중심 문장으로 정의하는 것은 주제문과 선택적 요약을 결합한 것으로 볼 수 있다. 따라서 본 논문에서는 위에서 제시한 주제문을 중심 문장(Topic Sentence)으로 사용하고자 한다.

문장의 중요도를 결정하는 방법 중 각각의 문장에 출현한 용어의 중요도에 따라 해당 문장의 중요도를 결정하는 방법이 있다. 이는 문장의 중요도를 구분하는 데 유용하게 사용될 만한 용어들을 선택하는 작업이다. 학습 문서에 나타나는 모든 용어를 선택할 수 없으므로 문서에 나타나는 용어들의 정보량을 계산하고 정보량이 큰 용어들만을 선택하는 연구가 진행되어왔다. 용어를 선택하는 방법에는 단어 출현 빈도, 상호 정보 척도, 정보 획득량, 카이 제곱 통계량 등이 있다[3]. 그러나 이러한 연구들은 문장의 문맥적인 정보를 반영하지 못한다는 문제가 있다.

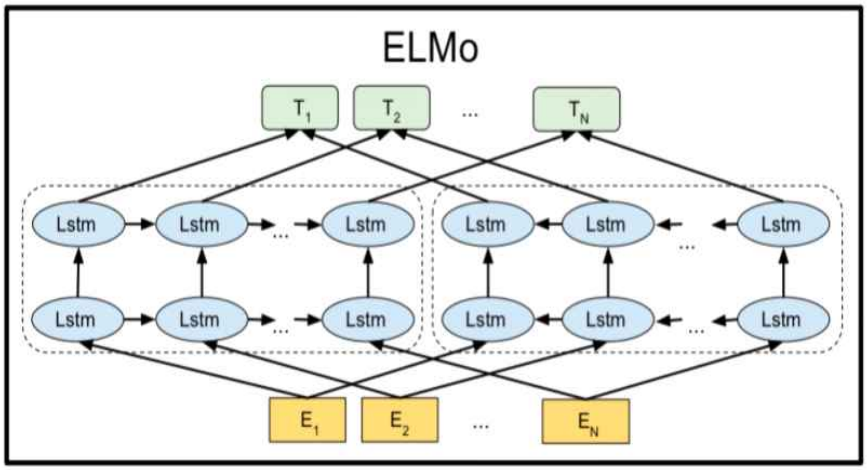
B. 임베딩 모델

임베딩 모델은 학습 과정과 활용 방법에 따라 두 가지로 나뉘는데, 임베딩 모델이 학습을 완료한 뒤 반환하는 고정된 벡터값을 활용하는 Word2Vec, FastText, GloVe 모델과 같은 Static Embedding Model과 임베딩 모델 자체를 불러와 학습에 적용하는 사전 학습 기반 언어 모델(Pre-trained language model)인 ELMo와 BERT가 있다[6].

1. ELMo

Word2Vec과 같은 단어 단위 임베딩에서는 동의어 처리가 안 되는 문제가 발생하는데, 이를 해결하기 위해서는 같은 단어라도 문맥에 따라 다른 벡터를 만들어 내는 방법이 필요하다. 그러기 위해서는 문장, 혹은 그 이상의 수준에서 임베딩을 수행하고 그 결과로 각 단어 혹은 전체 입력에 대해 벡터를 만들어 같은 단어가 입력으로 제시되더라도 주변 문맥에 따라 다른 벡터로 임베딩하는 기술이 필요한데, ELMo(Embeddings from Language Model)는 10억 개 단어에 대해 사전 훈련된 언어 모델(Pre-trained language model)로 기존 단어 기반 임베딩과 달리 문장을 임베딩하는 방법론이다[7]. Bank라는 단어를 예를 들면 Bank Account(은행 계좌)와 River Bank(강둑)에서의 Bank는 전혀 다른 의미를 갖는데, Word2Vec이나 GloVe 등으로 표현된 워드 임베딩 벡터들은 이를 제대로 반영하지 못한다는 단점이 있다. Word2Vec이나 GloVe 등의 임베딩 방법론으로 Bank란 단어를 [0.2 0.8 -1.2]라는 임베딩 벡터로 임베딩하였다고 하면, 이 단어는 Bank Account(은행 계좌)와 River Bank(강둑)에서의 Bank는 전혀 다른 의미임에도 불구하고 두 가지 상황 모두에서 [0.2 0.8 -1.2]의 벡터가 사용된다.

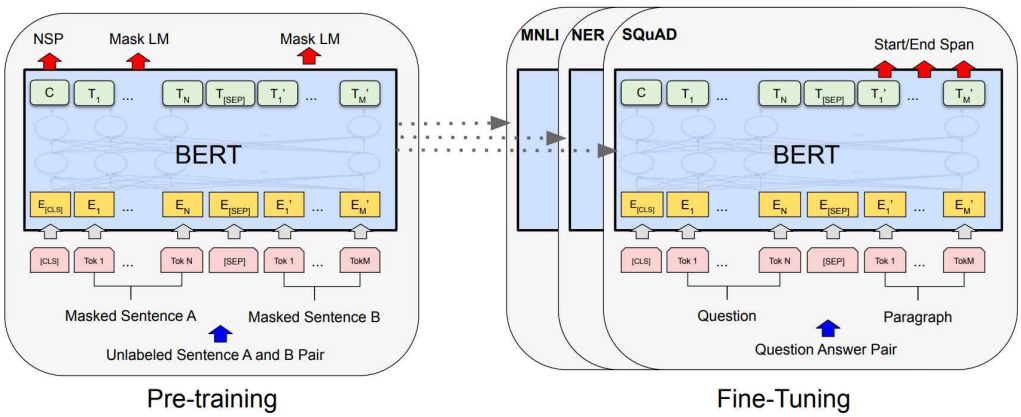
반면 ELMo는 문장 수준의 임베딩을 수행하여 그 결과로 문맥을 반영한 워드 임베딩(Contextualized Word Embedding)이 가능하다는 특징이 있다. ELMo는 2018년 Peters et al.의 논문을 통해 공개되었다. ELMo는 [그림 2-1]과 같이 문장을 정방향으로 예측하는 언어 모델과 역방향으로 예측하는 언어 모델을 각각 학습시킨 후, 각각의 언어 모델에서 얻은 벡터를 합치는 방법을 사용하였다[8].



[그림 2-1] ELMo가 문장을 임베딩하는 방법

2. BERT

BERT(Bidirectional Encoder Representations from Transformers)는 2018년 11월 구글이 공개한 인공지능 언어 모델로 전이 학습(Transfer Learning), 트랜스포머(Transformer) 구조를 활용하여 각종 자연어 처리 태스크(Task)를 해결한다[9].

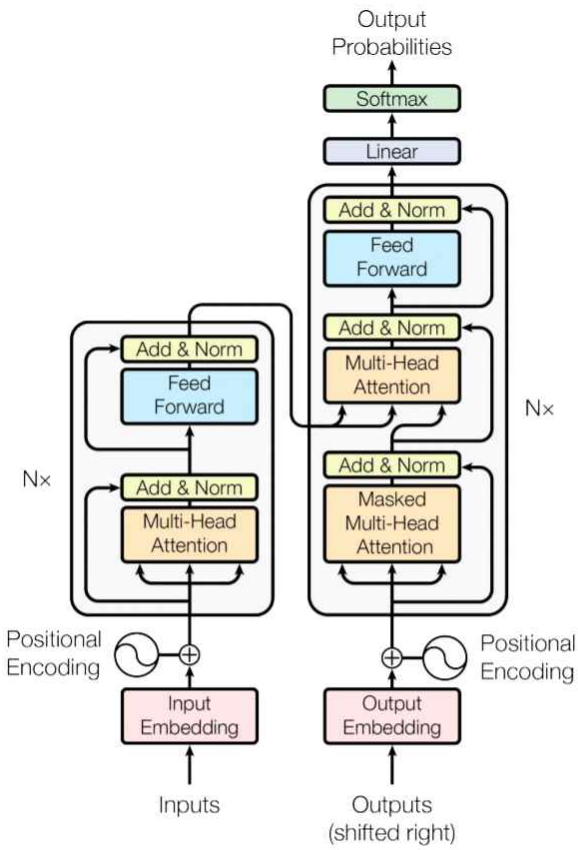


[그림 2-2] BERT 모델

BERT는 사전 학습 기반 언어 모델(Pre-trained language model)로 데이터가 미리 어느 정도 학습되어 있어서 이 모델에 해결하고자 하는 자연어 처리 태스크에 관련된

데이터로 파인 튜닝(Fine-tuning) 하는 방식으로 자연어 처리 문제를 해결하는데, BERT는 [그림 2-3]과 같은 트랜스포머 블록을 기본 모델로 사용한다.

트랜스포머(Transformer) 구조는 어텐션 메커니즘(Attention Mechanism)을 기반으로 하는데 어텐션 메커니즘은 기계번역을 위한 시퀀스 투 시퀀스(Sequence-to-Sequence) 모델에 처음 도입되었으며 아키텍처는 [그림 2-3]과 같다. 번역하고자 하는 언어를 입력으로 해서 이를 벡터로 만드는 앞부분을 인코더(Encoder)라고 하고, 인코더가 출력한 벡터를 입력으로 해서 타겟 언어를 출력하는 뒷부분을 디코더(Decoder)라고 한다. 그러나 입력으로 주어지는 문장의 길이가 길어질수록 모델의 성능이 나빠져 정확도가 떨어지게 된다. 이를 개선하기 위해 모델이 중요한 부분에 집중하게 만드는 것이 어텐션 메커니즘의 핵심이라고 할 수 있다.

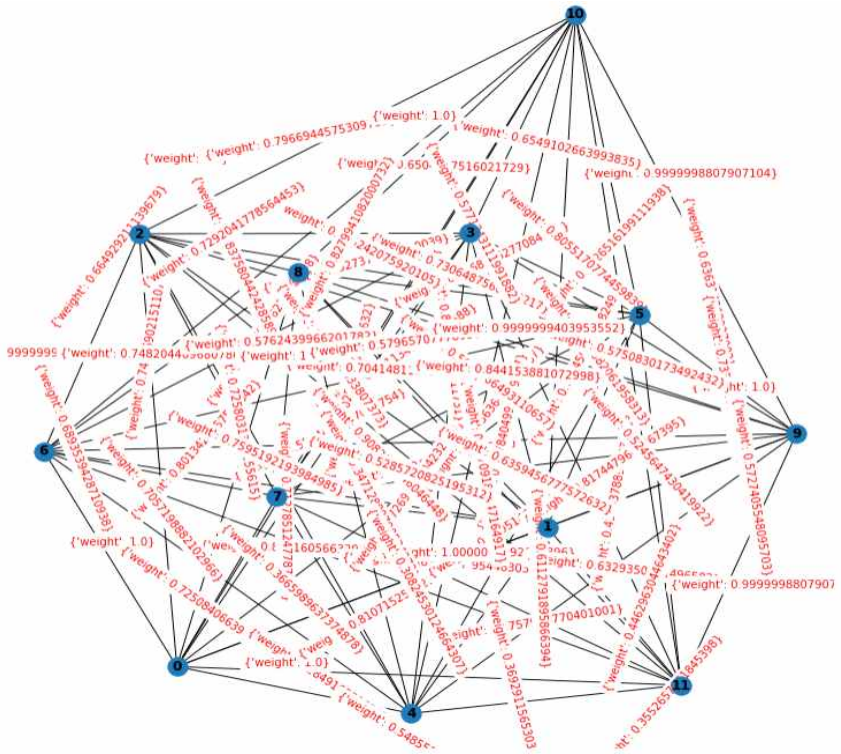


[그림 2-3] Attention Mechanism

C. 추출 요약 연구

1. TextRank 알고리즘을 이용한 한국어 중심 문장 추출

TextRank 알고리즘은 추출 요약을 위한 대표적인 그래프 기반 순위 알고리즘으로 PageRank[10]를 텍스트에 적용하여 중심 단어 및 중심 문장을 추출한다. TextRank Model[11]을 활용하여 중심 문장을 추출하기 위해서는 문장을 단위로 하여 문장 간 유사도를 기반으로 [그림2-4]와 같이 문장 유사도 그래프를 만든 다음, 그래프에 PageRank 알고리즘을 적용하여 각 노드(단어 또는 문장)의 순위를 계산한다. 이 순위가 높은 순서대로 중요 단어와 중요 문장이 된다.



[그림 2-4] ELMo 문장 임베딩을 적용한 문장 유사도 그래프

TextRank 알고리즘에서 간선의 가중치를 반영한 노드 V_i의 점수 $WS(V_i)$는 식 (1)의 방법으로 계산한다.

$$WS(V_i) = (1-d) + d * \sum_{V_j \in \ln(V_i)} \frac{W_{ji}}{\sum_{V_{k \in Out(V_j)}} W_{jk}} WS(V_j) \qquad (1)$$

한 문장을 나타내는 노드 V_i의 점수는 V_i에 진입 연결된 노드 V_j와의 유사도 W_{ij}를 V_j로부터 진출 연결된 노드 V_k와의 유사도 W_{jk}의 합으로 나눈 것을 노드 V_j의 점수와 곱하고, 이를 모든 V_j에 대해 반복하여 합한 후 변수 d를 고려하여 계산한다. 변수 d는 그래프에서 노드를 임의로 움직이며 방문하는 방문자가 더 이상 다른 노드에 방문하지 않을 확률을 의미한다[3]. [표2-1]은 하나의 뉴스 기사를 구성하는 문장을 기준으로 TextRank를 적용하여 계산한 문장 점수를 반영하여 순위를 매긴 결과와 실제 중심 문장 여부를 나타낸 것이다.

문장 순서	점수	순위	실제 중심 문장 여부
0	0.086084	6	○
1	0.083862	8	○
2	0.088520	4	
3	0.091816	1	○
4	0.086064	7	
5	0.088790	2	-
6	0.082630	9	
7	0.088595	3	-
8	0.087368	5	
9	0.080095	10	
10	0.060580	12	
11	0.075589	11	

[표 2-1] TextRank를 적용한 문장 점수와 실제 중심 문장 여부

TextRank에서는 결국 자주 출현하는 단어가 많이 포함된 문장이 높은 순위를 차지하게 된다. 즉, 중요 단어가 많이 포함된 문장을 중심 문장으로 간주할 수 있다.

[표 2-1]은 본 연구의 비교 연구로 주어진 문장을 토큰화하여 명사만 추출하고 이를 ELMo로 워드 임베딩한 다음 문장을 구성하는 단어 임베딩 값의 평균을 문장 벡터로 하여 TextRank 알고리즘을 적용하여 문장 중요도 점수를 계산한 결과이다.

홍진표의 연구에서는 한국어 품사 태거를 이용하여 어절 내 조사, 어미, 접사 정보를

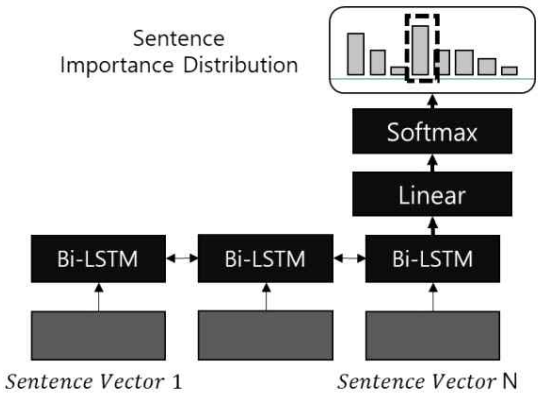
제거하여 어근을 추출하고, 문장 간 가중치를 계산한다. 이때, TextRank의 경우 방향성이 없기 때문에 W_{ij}와 W_{ji}가 같은 값을 가지게 된다. 그리고 이들 가중치는 문장 i와 문장 j 사이에 공통으로 존재하는 단어들을 이용하여 식 (2)와 같이 문장 유사도 계산식을 만들어 사용하였다[12].

$$Sim(S_i,S_j) = \frac{|\{w_k|w_k \in S_i \wedge w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}$$

$$TR(V_i) = (1-d) + d\sum_{j=0}^{N-1} \frac{Sim(S_i,S_j)}{\sum_{k=0}^{N-1} Sim(S_j,S_k)} \times TR(V_j) \quad (2)$$

2. 추출 요약 네트워크

김원우의 연구[13]에서는 입력 문서에서 중요한 문장을 추출하기 위해 합성 곱 신경망(Convolutional Neural Network, CNN) 기반의 문장 인코더와 양방향 LSTM(Bidirectional Long Short-Term Memory) 기반의 중요 문장 추출기를 결합하여 그림과 같은 구조를 제안하였다. CNN-양방향 LSTM 기반의 중요 문장 추출기는 추출 요약을 위해 문장 벡터를 입력으로 사용하였다[14].



[그림 2-5] CNN-양방향 LSTM 기반의 중요 문장 추출기

양방향 LSTM 기반의 중요 문장 추출기는 입력 문장의 벡터들을 순서대로 받아들여 문장 중요도 분포를 생성하여 중요 문장을 추출하였다[15]. 이는 딥러닝 기반의 중심 문장 판별 과정이라고 할 수 있다.

3. 딥러닝 기반의 2단계 한국어 문서 요약

문서를 이루고 있는 모든 문장이 주제를 가진 중심 문장은 아니다. 전반적인 내용 이해를 위해 배경지식을 설명한 문장, 연관된 다른 내용을 담은 문장, 주제와 관련 없는 개인의 감정을 담은 문장 등 문서의 내용을 요약하는 데에 있어서 중요하지 않은 노이즈로 판단되는 문장들도 있다. 전재원의 연구에서는 추출 요약을 통해 이런 노이즈 문장을 제외한 중요 문장들만 추출하여 추출된 문장들을 기반으로 요약문을 생성하는 방법을 제안한다. 또한 추출 요약의 성능을 올리기 위해 Maximal marginal relevance(MMR) 방법을 활용하여 문장을 선택할 때 이미 선택된 문장들과 중복된 문장을 줄이고 중요 정보가 담긴 문장을 선택할 수 있도록 하였다[15]. MMR이란 문서의 주제와 관련성이 높은 문장을 선택할 때 이전에 선택된 문장들과 유사도를 비교하여 이전 문장들과의 유사도가 낮은 문장을 선택하도록 하는 방법론이며 선택된 문장 간의 정보 중복 문제를 해결할 수 있다. 다음 식 (3)은 MMR 계산식을 나타낸다.

$$Arg \max_{S_j \in D} \left[\alpha \cdot Sim_1(S_i, Q) - (1-\alpha) \cdot \max_{S_j \in R} (Sim_2(S_i, S_j)) \right] \quad (3)$$

위 식 (3)에서 D는 문서를 이루고 있는 문장들의 전체 집합을 의미하고 R은 D의 부분 집합으로 이미 선택된 문장들의 집합이다. S는 각 집합을 이루고 있는 하나의 문장을 의미한다. 문장과 문장의 유사도는 코사인 유사도와 내적 유사도 함수를 사용한다.

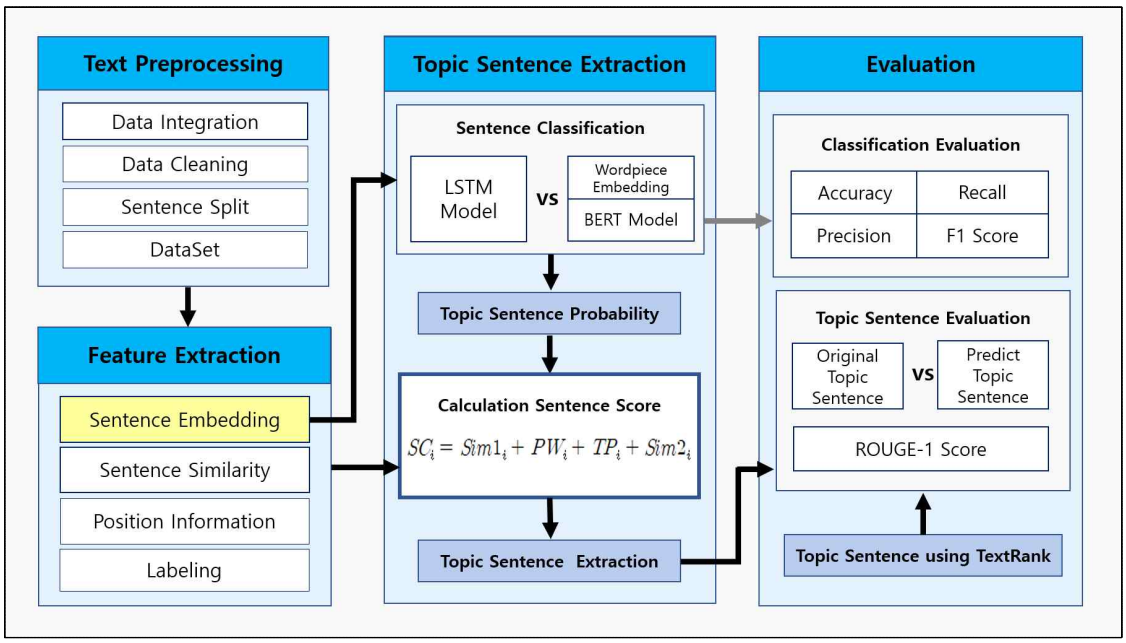
추출 모델은 인코더(encoder)와 문장을 추출하는 추출기(extractor)로 이루어졌다. 이때 사용한 인코더는 3가지로 평균 인코더, 순환 신경망 인코더, 합성 곱 신경망 인코더로 구현되어 있으며 학습 시 이 중 하나를 선택한다. 평균 인코더는 단순하게 문장을 이루고 있는 각 단어 임베딩을 모두 합해서 평균을 계산하여 문장 임베딩으로 이 평균값을 그대로 사용한다. 순환 신경망 인코더는 양방향 순환 신경망을 사용한다. 인코더에서 문장을 입력받아서 양방향 순환 신경망을 거쳐 나온 정방향의 마지막 은닉층의 출력과 역방향의 마지막 은닉층의 출력을 연결하여 문장 임베딩으로 사용한다. 합성곱 신경망 인코더는 입력받은 문장이 max pooling을 거쳐 모든 합성 곱 필터를 거쳐 나온 출력을 연결하여 문장 임베딩으로 사용한다. 3가지 인코더 중 선택된 인코더를 통해 생성된 문장 임베딩을 추출기를 통해 추출할 확률을 계산한다. 추출기는 2가지 방식으로 구현되었다. 순환 신경망 추출기와 sequence-to-sequence 추출기로 구현되어 있으며 인코더와 마찬가지로 학습 시 한 가지를 선택하여 사용한다[15].

Ⅲ. 딥러닝 기반 중심 문장 추출 방법

본 장에서는 국립국어원에서 제공하는 '모두의 말뭉치'에서 '신문 말뭉치'와 '문서 요약 말뭉치'를 실험에 맞게 전처리한 후 BERT와 LSTM 모델을 이용하여 중심 문장 확률을 계산하고, 문장 중요도 계산을 위한 특성으로 각 문장별 기사 제목과의 유사도, 기사 내 다른 문장과의 유사도, 문장 위치 정보를 추출한다. 추출된 특성을 문장 중요도 계산에 반영하고 문장별 순위를 구하여 중심 문장을 추출하는 방법을 제안하고자 한다.

A. 시스템 구성도

[그림 3-1]은 제안하는 중요 문장 추출 방법의 전체 시스템 구성도이다. 데이터를 전처리하는 부분과 문장 중요도를 계산할 때 필요한 특성을 추출하는 단계, 추출된 특성을 통해 문장별 점수를 계산하여 중심 문장을 추출하고 이를 평가하는 단계로 나뉜다.



[그림 3-1] 시스템 구성도

Text Preprocessing 과정은 json 형식으로 되어있는 '신문 말뭉치'와 '요약문서 말뭉치'를 뉴스 기사 id를 기준으로 통합하고, 특수문자 및 기사 내용과 관련 없는 데이터를 제거하여 실험에 필요한 데이터를 구성한다.

Feature Extraction 과정은 전처리 과정에서 구성한 데이터 셋을 대상으로 문장 중요도를 계산하는 데 사용할 특성인 제목과의 유사도, 기사 내 다른 문장과의 유사도, 문장의 기사 내 위치 정보, 기사 id를 추출하고, 기사 내 모든 문장에 대해 중심 문장 여부를 라벨링하여 다음 과정에 필요한 데이터 셋을 구성한다.

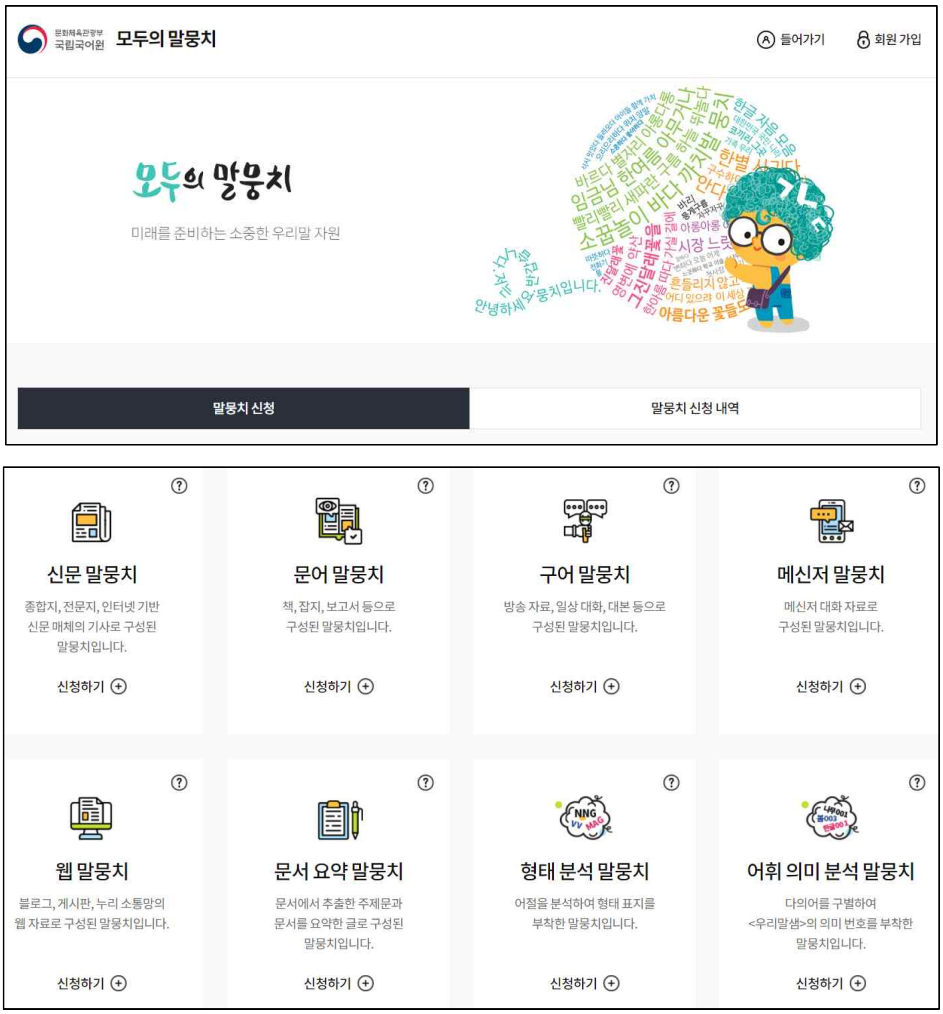
Topic Sentence Extraction 과정에서는 훈련 데이터를 BERT 모델 및 ELMo 언어 모델로 임베딩한 벡터값을 LSTM 모델로 각각 분류 학습한 다음, 딥러닝 모델을 기반으로 각 기사문에 대한 예측값을 중심 문장의 특성, 즉 중심 문장일 확률로 간주하여 문장 중요도를 계산하는 특성으로 추출한다. 이를 Feature Extraction 과정에서 추출한 특성인 문장 간 유사도 및 문장 위치별 가중치와 결합하여 기사 내 각 문장의 중요도 점수를 계산하여 문장의 순위를 구하고 이를 기준으로 중심 문장을 추출한다.

마지막 평가 단계에서는 Topic Sentence Extraction에서 활용한 딥러닝 모델의 분류 성능을 분류성능평가지표로 평가하고, 본 연구에서 제안한 방법을 적용한 중심 문장 추출 성능은 실제 중심 문장과의 일치 비율 및 ROUGE-1 스코어를 통해 TextRank 방법과 비교하는 과정을 거친다.

B. 데이터 수집 및 전처리

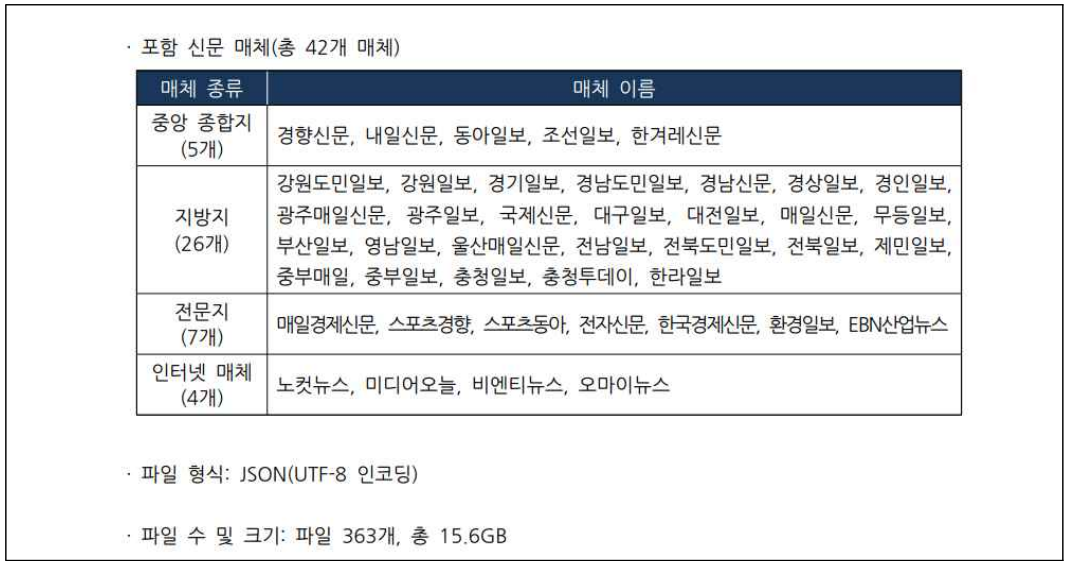
1. 데이터 수집

본 절에서는 데이터 수집 방법에 관하여 기술한다. [그림 3-2]와 같이 국립국어원 모두의 말뭉치(https://corpus.korean.go.kr/)에서는 다양한 말뭉치를 제공하고 있다. 본 연구에서는 '신문 말뭉치'와 '문서 요약 말뭉치'를 사용하였다.



[그림 3-2] 국립국어원 모두의 말뭉치

국립국어원 신문 말뭉치는 2009년부터 2018년까지 10년 동안 생산된 신문 기사로 1억여 어절, 기사 총 353만여 건으로 구성되어 있다. [그림 3-3]은 신문 말뭉치 사용 설명서의 일부로, 파일 형식이 UTF-8로 인코딩된 JSON(JavaScript Object Notation) 파일로 되어있음을 알 수 있다.



· 포함 신문 매체(총 42개 매체)

매체 종류	매체 이름
중앙 종합지 (5개)	경향신문, 내일신문, 동아일보, 조선일보, 한겨레신문
지방지 (26개)	강원도민일보, 강원일보, 경기일보, 경남도민일보, 경남신문, 경상일보, 경인일보, 광주매일신문, 광주일보, 국제신문, 대구일보, 대전일보, 매일신문, 무등일보, 부산일보, 영남일보, 울산매일신문, 전남일보, 전북도민일보, 전북일보, 제민일보, 중부매일, 중부일보, 충청일보, 충청투데이, 한라일보
전문지 (7개)	매일경제신문, 스포츠경향, 스포츠동아, 전자신문, 한국경제신문, 환경일보, EBN산업뉴스
인터넷 매체 (4개)	노컷뉴스, 미디어오늘, 비엔티뉴스, 오마이뉴스

· 파일 형식: JSON(UTF-8 인코딩)

· 파일 수 및 크기: 파일 363개, 총 15.6GB

[그림 3-3] 신문 말뭉치 설명서

국립국어원 '문서 요약 말뭉치'는 위에서 제시한 '신문 말뭉치'에서 추출한 기사 4,389건을 대상으로 기사에서 추출한 중심 문장(topic_sentences)과 기사를 요약하여 작성한 문장(summary)으로 구성된 말뭉치로 기사에서 추출한 중심 문장(topic) 13,167개, 기사 요약 작성 문장(summary) 13,167개로 이루어졌다. 각 말뭉치의 데이터 형식은 [표 3-1]과 같이 키-값 형태의 JSON(JavaScript Object Notation)으로 '신문 말뭉치'는 document_id를 기준으로 단락별로 구분된 기사 본문 데이터로 구성되어 있고 '문서 요약 말뭉치'는 '신문 말뭉치'의 해당 기사와 'document_id' 공유하며, 기사 제목(head), 해당 기사의 중심 문장(topic_sentences), 요약 문장(summary)으로 구분되어 있다.

문서 요약 말뭉치	"document_id": "NWRW1800000021.6", "subclass": "NA", "head": "與, 쟁점법안 대폭후퇴 끝 타결", "subhead": "", "topic_sentences": ["여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다.", "이로써 지난해 12월 10일 개회 이래 4주 동안 야당의 본회의장 불법 점거와 국회의장의 질서유지권 발동 등으로 가파르게 대치했던 정국이 정상화됐다.", "여야는 최대 쟁점이었던 미디어 관계 법안 8건 중 언론중재법 등 여야 간 이견이 없는 법안 2건만 이번 임시국회에서 협의 처리하고 신문·방송 겸영 허용과 대기업 방송 진출 허용 등 여야가 맞서는 6개 법안은 이른 시일 안에 합의 처리하도록 노력하기로 했다."],
신문 기사 말뭉치	{ "id": "NWRW1800000021.6", "metadata": { "title": "동아일보 2009년 기사", "author": "조수진 기자 jin0619@donga.com", "publisher": "동아일보사", "date": "20090107", "topic": "NA", "original_topic": "정치" }, "paragraph": [{ "id": "NWRW1800000021.6.1", "form": "與, 쟁점법안 대폭후퇴 끝 타결" }, { "id": "NWRW1800000021.6.2", "form": "여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다. 이로써 지난해 12월 10일 개회 이래 4주 동안 야당의 본회의장 불법 점거와 국회의장의 질서유지권 발동 등으로 가파르게 대치했던 정국이 정상화됐다." },

[표 3-1] 말뭉치 json 구성

수집된 데이터를 실험을 위한 데이터 셋으로 구성하기 위해 '신문 말뭉치' 파일 363개와 '문서 요약 말뭉치' 1개를 뉴스 기사 id를 기준으로 결합한다. 그리고 [표 3-1]과 같이 분리되어 있는 json 파일을 파이썬의 json 모듈 및 pandas 모듈을 이용하여 [표 3-2]의 데이터 결합 과정을 거쳐 [그림 3-4]와 같이 실험에 필요한 기본 데이터 프레임(pandas.DataFrame)으로 구성한다.

```
contents = []
ids = []
for news_id in news_ids:
    fname = './말뭉치/' + news_id.split('.')[0]
    filename = fname + '.json'
    with open(filename, encoding="utf-8") as json_file:
        json_data = json.load(json_file)
        for doc in json_data['document']:
            if doc['id'] == news_id:
                contents.append(doc['paragraph'])
                ids.append(news_id)
                                    (중략)
df = pd.DataFrame({'id': news_ids, 'title': titles, 'topic':topic_sentences,
                   'summary':summary_sentences, 'article': contents})
```

[표 3-2] 말뭉치 데이터 결합 코드

	id	title	topic	summary	article
0	NWRW1800000021.6	與, 쟁점법안 대폭후퇴 끝 타결	[여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다...	[한나라당 홍준표, 민주당 원혜영, 선진과 창조의 모임 문국현 원내대표는 6일 임시...	[여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다...
1	NWRW1800000021.8	[크로스 미디어]"장보고號 힘내세요 우리들이 있잖아요"	[지난해 12월 중순 카리브 해에 있는 바하마를 떠난 장보고호는 이달 2일 미국령 ...	[지난해 10월 9일 미국 아나폴리스 항을 떠나 450여 일간의 해양 탐사에 나선 ...	["박사님, 항해 힘드셔도 조금만 참으시고 우리나라를 위한 연구 많이 하세요. 승환...

[그림 3-4] 데이터 프레임 생성 결과

2. 전처리

기본 데이터 프레임으로 구성한 후 텍스트 전처리 과정은 수집한 데이터를 실험에 적합하게 전처리하는 과정으로 정규 표현식(Regular Expression)을 통해 한글, 영어, 숫자만을 남기고 기사 내용과 상관없는 불필요한 특수문자 등을 제거한다.

```
import re
def preprocess_sentence(sentence):
    sentence = re.sub(r'\([^)]*\)', '', sentence) #()괄호로 닫힌 문자열 제거
    sentence = re.sub(r'\[[^]]*\]', '', sentence) #[]괄호로 닫힌 문자열 제거
    sentence = re.sub('\d{2,3}-\d{3,4}-\d{4}','',sentence) #전화번호 형식 제거
    sentence = re.sub('\d{3,4}-\d{4}','',sentence) #전화번호 형식 제거
    #한글, 영문, 숫자, ., % 이외 문자 제거
    sentence = re.sub('[^A-Za-z0-9가-힣\.%]', ' ', sentence)
    sentence = re.sub('.*면에 관련기사$', '', sentence) #불필요한 내용 제거
    return sentence
def preprocess(sentence_list):
    clean_text = []
    for s in sentence_list:
        clean_text.append(preprocess_sentence(s).strip())
    return clean_text
```

[표 3-3] 특수문자, 전화번호 등 불필요한 데이터 제거 과정

정제된 데이터를 Feature Extraction에 필요한 문장 단위 데이터 셋으로 구성하는 과정을 수행한다. 전체 기사문을 문장 단위로 분리하고, 각 기사문에서의 문장 위치를 라벨링 하는 과정을 거친다. 이때, 기사별로 구성하는 문장의 수가 다르므로, [표 3-4]의 코드를 이용하여 각 기사문의 길이에 비례하여 Min-Max Scaling 하여 0부터 10의 범위로 구분하여 라벨링 한다.

Min-Max Scaling은 변수들의 척도(Scale)가 서로 다를 경우, 상호 비교를 위해 표준화하는 방법 중 하나로 최소값(Min)과 최대값(Max)을 사용해서 0 ~ 1 사이의 범위(range)로 데이터를 표준화해주는 방법이다.

```
def calc_pos_t(sentence, sentences):
    max = len(sentences)-1
    min = 0
    x = sentences.index(sentence)
    pos = (x - min) / (max - min)
    return round(pos,1)
```

[표 3-4] Min-Max Scaling을 적용한 문자별 위치 라벨링 함수

[그림 3-5]는 [표 3-5] 코드를 이용하여 각 문장을 단어 단위로 분리한 개수를 문장 길이(sentence_len)로 계산하고, 각 문장의 기사 본문 내 위치 정보와 결합하여 데이터 셋을 구성한 결과이다.

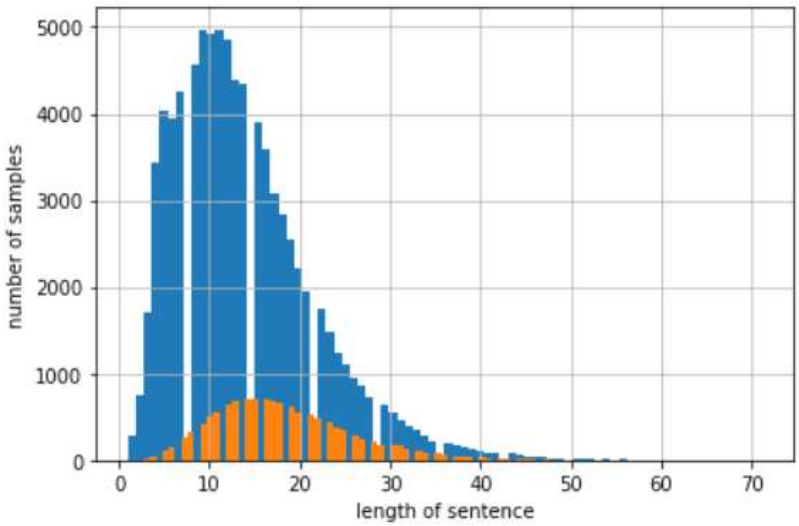
```
data1["sentence_len"] = data1.apply(lambdax : len(x["sentence"].split()), axis=1)
```

[표 3-5] 단어 단위 문장 길이를 구하는 함수

	newsid	title	sentence	position	sentence_len
0	newsid0	쟁점법안 대폭후퇴 끝 타결	여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다.	0.0	12
1	newsid0	쟁점법안 대폭후퇴 끝 타결	이로써 지난해 12월 10일 개회 이래 4주 동안 야당의 본회의장 불법 점거와 국회...	0.1	20
2	newsid0	쟁점법안 대폭후퇴 끝 타결	한나라당 홍준표 민주당 원혜영 선진과 창조의 모임 문국현 원내대표는 이날 국회 귀빈...	0.2	29
3	newsid0	쟁점법안 대폭후퇴 끝 타결	여야는 이견이 없는 민생법안 등 100여 건을 이번 임시국회에서 협의 처리하되 아직...	0.3	25
4	newsid0	쟁점법안 대폭후퇴 끝 타결	그러나 여야는 이날 합의에서 상당수 쟁점법안의 처리 시기는 물론이고 법안 상정 여부...	0.4	26

[그림 3-5] 데이터 셋 구성

구성된 데이터 셋의 문장 길이 분포를 살펴보면 [그림 3-6]과 같다. 문장 길이가 2개 이하 이거나, 60개 이상인 경우 대부분 라벨이 0이므로 [표 3-6]과 같이 단어의 개수가 3개 미만이거나 60개 이상인 단어를 불리언 인덱싱(Boolean indexing)으로 제거하는 과정을 거친다.



[그림 3-6] 문장 길이 분포

```
#data5는 [그림 3-5]에서 제시한 데이터프레임
data5 = data5[data5['sentence_len'] >= 3] #단어 개수 2개 이상인 행 제거
data5 = data5[data5['sentence_len'] < 60] #단어 개수 60 이상인 행 제거
```

[표 3-6] 불리언 인덱싱 과정

C. 문장 중요도 특성 추출

본 절에서는 문장 중요도 계산에 필요한 문장 특성인 각 문장의 제목과의 유사도, 기사 내 다른 문장과의 유사도, 문장 위치에 따른 가중치를 구한다.

1. ELMo 워드 임베딩을 활용한 문장 유사도

각 문장을 명사 중심으로 토큰화하고 ELMo로 워드 임베딩하여 [그림 3-7]과 같이 128차원의 벡터값으로 표현한다. 각 문장의 벡터값은 코사인 유사도 식 (4)를 이용하여 기사 제목 및 기사 본문 내 다른 문장과의 유사도를 구하는 과정을 거친다.

```
여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다


array([ 0.48413116,  0.02125069, -0.5875636 , -0.2950697 ,  0.26770344,
        0.12444406,  0.69861555,  0.32422206,  0.27966654, -0.41357046,
       -0.37435704,  0.20517652,  0.00965248, -0.04077113,  1.048938  ,
       -0.2876489 , -0.57986677,  0.20061953, -0.04756998, -0.00739985,
       -0.15793768,  0.26954025,  0.43857905,  0.3757811 , -0.67592204,
       -0.1513905 , -0.36356336,  0.114916  , -0.30246648, -0.6039482 ,
       -0.5596775 ,  0.65573716, -0.40903753,  0.21421635, -0.623251  ,
        0.504211  , -0.6075504 , -0.08347736, -0.2510773 , -0.39836356,
        1.0925462 , -0.10889237, -0.86108357, -0.48504406,  0.3686896 ,
       -0.6900034 ,  1.2635028 ,  0.18607666, -0.58939606,  0.20880525,
        0.66628075, -0.6588631 , -0.38959393,  0.2727479 ,  0.06292176,
       -1.0207047 ,  0.652409  , -0.3936878 ,  0.24917386,  0.438222  ,
       -0.10943795,  0.38181597, -0.29306242,  0.0574393 , -0.680269  ,
       -0.40044206,  0.6503669 , -0.31609887, -0.19838266, -0.20558572,
        0.54982734,  0.66497195, -0.30903804, -0.36206803, -0.43479192,
       -0.14229877,  0.26497334, -0.5316558 ,  0.02163842,  0.251706  ,
       -0.03081695, -0.20023641, -0.1339931 ,  0.6546439 ,  0.19447827,
       -0.14711756, -0.34749928, -0.858346  ,  0.44693595,  0.07012479,
        0.12517634, -0.10543625, -0.9831672 , -0.03618566, -0.5376618 ,
       -0.01353827, -0.32151252, -0.5898766 , -0.40794373,  0.23541911,
       -0.38519236,  0.15233281, -0.12275665,  0.8938507 ,  0.37201658,
       -0.48923934,  0.30044812, -0.3981959 , -0.26033002, -0.7226772 ,
       -0.00821269,  0.86163455, -0.31612805, -0.70643896,  0.4564102 ,
        0.14626737, -0.2805233 , -0.04903889,  0.02880669, -0.43825358,
       -0.00991744, -0.46457362, -0.2747316 ,  0.78515047, -0.8694726 ,
       -0.04711483, -0.02330188, -0.94347364], dtype=float32)
```

[그림 3-7] ELMo 문장 임베딩 예

코사인 유사도는 두 벡터 간의 코사인 각도를 이용하여 구할 수 있는 두 벡터의 유사도를 의미한다. 두 벡터의 방향이 완전히 동일한 경우는 1의 값을 가지며, 90도의 각을 이루면 0, 180도의 반대 방향을 가지면 -1의 값을 갖게 된다.

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} \quad (4)$$

뉴스 기사의 각 문장(sentence)과 기사 제목(title), 뉴스 기사를 구성하는 문장들과의 코사인 유사도를 계산하여 제목과의 유사도(similarity_title), 다른 문장과의 유사도(similarity) 컬럼을 구성한 데이터프레임 결과는 [그림 3-8]과 같다.

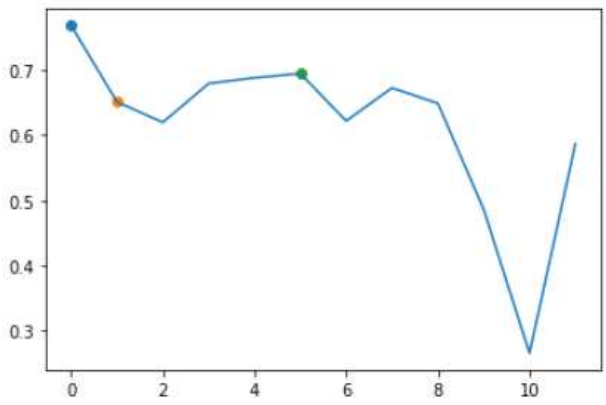
	sentence	similarity_title	similarity
0	여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다.	0.767785	[0.99999994, 0.71113425, 0.725084, 0.83416057,...
1	이로써 지난해 12월 10일 개회 이래 4주 동안 야당의 본회의장 불법 점거와 국회...	0.650596	[0.71113425, 1.0, 0.67956364, 0.71445644, 0.65...
2	한나라당 홍준표 민주당 원혜영 선진과 창조의 모임 문국현 원내대표는 이날 국회 귀빈...	0.619279	[0.725084, 0.67956364, 1.0, 0.8107153, 0.72580...
3	여야는 이견이 없는 민생법안 등 100여 건을 이번 임시국회에서 협의 처리하되 아직...	0.678760	[0.83416057, 0.71445644, 0.8107153, 1.0000001,...
4	그러나 여야는 이날 합의에서 상당수 쟁점법안의 처리 시기는 물론이고 법안 상정 여부...	0.687531	[0.7482691, 0.6504138, 0.72580314, 0.8129954, ...

[그림 3-8] 문장 유사도

[그림 3-9]는 문장 간 유사도 벡터값을 matplotlib 라이브러리를 활용하여 시각적으로 표현한 결과이다. 위쪽은 기사문의 제목 벡터 이미지이고, 아래쪽은 기사문의 첫 번째 문장이자 기사문의 중심 문장으로 유사도가 0.76으로 유사도가 높다는 것을 알 수 있다.

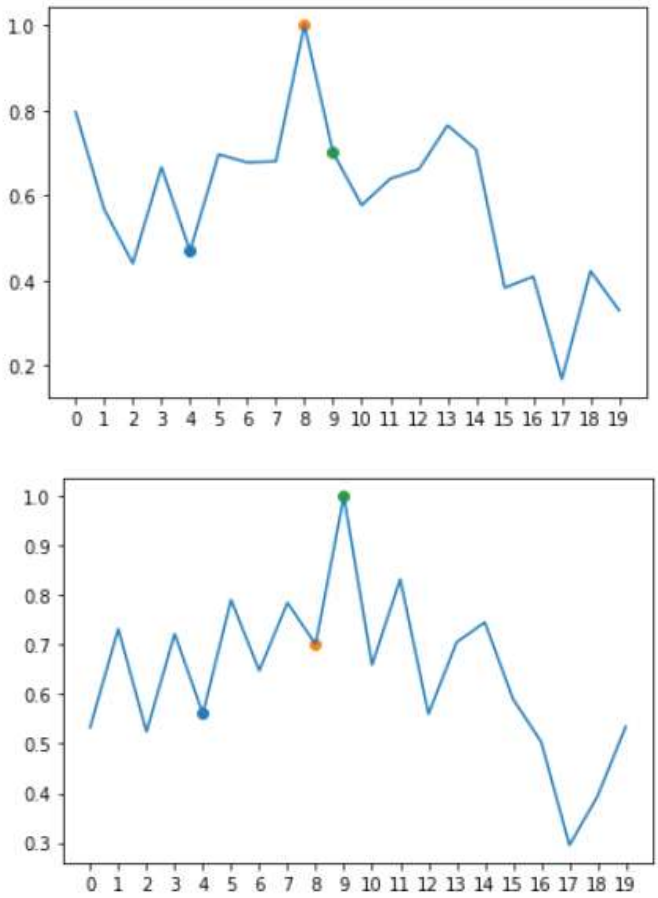
[그림 3-9] 문장 벡터 이미지 표현

뉴스 기사를 구성하는 각 문장과 뉴스 기사 제목과의 유사도가 중심 문장 여부와 어떤 관계가 있는지 그래프로 표현해 보면 [그림 3-10]과 같다. 가로 축은 문장 인덱스 번호로 한 문서 내 0번 ~ 11번 문장을 나타내고, 점으로 표시된 위치는 중심 문장을 나타낸다. 이를 통해 중심 문장은 대체로 기사 제목과의 유사도가 크다는 것을 알 수 있다.



[그림 3-10] 문서 제목과 중심 문장의 유사도 관계 탐색

[그림 3-11]은 중심 문장을 기준으로 기사 내 다른 문장과의 유사도 분포를 탐색해 보는 과정이다. 가로축은 문장 인덱스이고, 점으로 표시된 위치는 중심 문장을 표시한 것이다. 이를 통해 문장 유사도를 고려할 때, 중심 문장을 중심으로 일부 주변 문장의 유사도만을 고려하고자 한다.



[그림 3-11] 중심 문장 기준 인근 문장과의 유사도 탐색

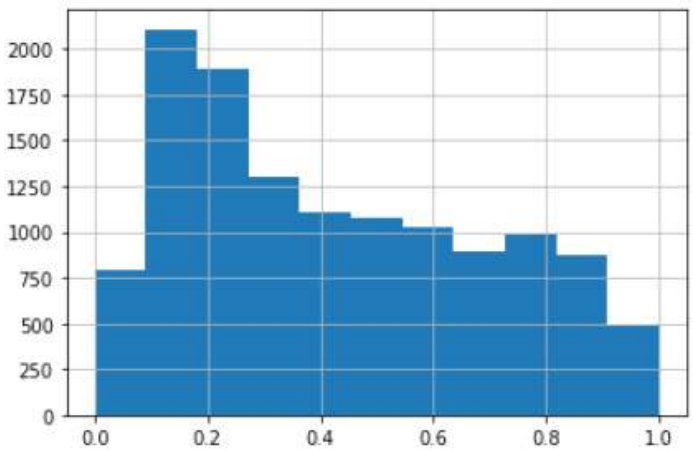
2. 문장 위치별 가중치

일반적으로 기사문을 작성할 때 중심 문장을 글의 앞부분에 배치하는 경우가 많다. [표 3-7]은 실험 데이터에서 중심 문장의 위치를 10개 구간으로 구분하여 구간별 개수

를 구한 것이고 [그림 3-12]는 위치별 중심 문장 분포를 히스토그램으로 표현한 것이다. 본 절에서는 해당 문장의 기사문 내 위치에 따른 특성을 반영하고자 문장 위치별 가중치를 [표 3-7] 기준으로 적용한다.

위치 구분	중심 문장의 개수	비율
0	797	0.063
1	2108	0.168
2	1895	0.151
3	1296	0.103
4	1111	0.089
5	1075	0.086
6	1024	0.082
7	990	0.079
7	896	0.071
9	872	0.069
10	489	0.039
합계	12553	1

[표 3-7] 위치별 중심 문장 개수



[그림 3-12] 위치 구간별 중심 문장 분포

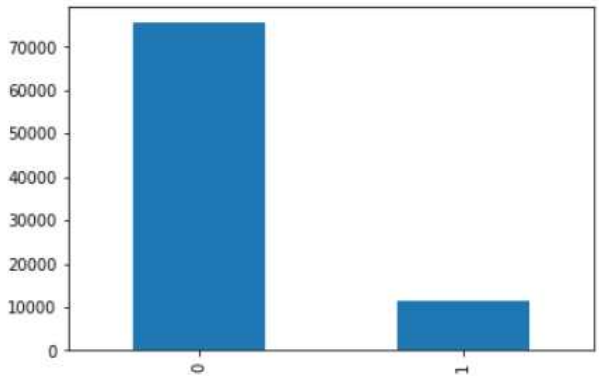
D. 중심 문장 추출 방법

본 절에서는 앞에서 추출한 특성과 LSTM 모델과 사전 학습된 언어 모델인 BERT를 통해 학습한 결과를 중심 문장 확률로 가정하고, 문장별 중요도 점수를 계산하는 식을 고안하여 중심 문장을 추출한다.

1. 딥러닝을 활용한 중심 문장 판별

딥러닝 모델을 통해 문장의 긍정과 부정을 분류하는 많은 연구가 진행되고 있다. 이에 착안하여 중심 문장도 딥러닝 모델이 중심 문장 여부를 분류 학습하는 과정에서 중심 문장의 특성을 학습할 수 있다고 가정하여 실험을 진행하였다. 실험에 사용할 딥러닝 모델은 BERT와 LSTM 모델이다.

실험 데이터는 전체 기사 중 4,000개 기사, 86,912개 문장을 사용하는데, 데이터 내 0, 1 라벨별 샘플수를 확인하면 [그림 3-13]과 같이 라벨이 0인 데이터는 75,487개, 라벨이 1인 데이터는 11,425개이다. 중심 문장 추출 단계에서 데이터 편중을 제거하기 위해 [표 3-8]의 코드로 라벨이 0인 데이터의 수를 줄이는 과정을 거친다.



[그림 3-13] 라벨별 샘플 수 비교

```
data_label_0 = data_label_0[:11425]
train = pd.concat([data_label_0, data_label_1], axis=0)
train = train.sample(frac=1).reset_index(drop=True)
```

[표 3-8] 라벨별 샘플 수 조정

a. LSTM 모델 활용 문장 판별 방법

LSTM 모델로 문장의 중심 문장 여부를 판별하기 위해 ELMo로 임베딩된 1차원 데이터를 2차원으로 변환한 벡터를 X, 중심 문장 여부를 0과 1의 숫자로 표시한 라벨(Label)을 Y로 지정한다. 신경망층 Dense의 하이퍼파라미터(Hyper Parameter)는 Units을 1, 활성화 함수(Activation)는 시그모이드(Sigmoid) 함수로 설정하였다.

```
model = Sequential()
model.add(LSTM(10))
model.add(Dense(1, activation='sigmoid'))

es = EarlyStopping(monitor='val_loss', mode='min', verbose=2, patience=3)
mc = ModelCheckpoint('LSTM_model.h5', monitor='val_acc',
                     mode='max', verbose=1, save_best_only=True)
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(X_train, Y_train, epochs=15,
                    callbacks=[es, mc], batch_size=32,
                    validation_data=(X_test, Y_test))
```

[표 3-9] LSTM 모델 구조

학습된 모델로 데이터의 문장별 분류 결과를 확률값으로 리턴하여 이를 해당 문장의 중심 문장 확률로 간주하고 문장별 중요도를 계산하는데 사용한다.

b. BERT 모델 활용 문장 판별 방법

BERT 언어 모델은 트랜스포머를 기반으로 한다. 그래서 실험에 이를 활용하기 위해서는 Hagging Face의 transformers 패키지와 구글의 setencepiece 패키지를 설치하여 실험을 진행한다. transformers 패키지는 BertTokenizer 등 BERT 언어 모델을 활용하는 데 필요한 다양한 모듈을 지원한다.

BERT 언어 모델의 입력은 sentencepiece와 같은 BPE(Byte Per Encoding) 방식의 토크나이저를 활용하여 [표 3-10]과 같이 '[CLS]' 토큰과 '[SEP]' 토큰을 문장의 시작과 끝 위치에 삽입해주는 과정이 필요하다.

```
sentences = ["[CLS] "+ str(sentence) + " [SEP]" forsentence insentences]
```

[표 3-10] BERT 모델의 입력 형식

[표 3-11]은 BERT 토크나이저 모듈을 활용하여 구글이 제공하는 Multilingual BERT 모델을 사용하는 코드이다. 이 모델은 100개 이상의 언어에 적용할 수 있는 다국어 모델로 BPE(Byte-pair encoding) 방식의 토크나이저인 sentencepiece를 사용한다.

```
tokenizer
= BertTokenizer.from_pretrained('bert-base-multilingual-cased',
                                do_lower_case=False)
tokenized_texts = [tokenizer.tokenize(sent) forsent insentences]
print (sentences[0])
print (tokenized_texts[0])
```

```
[CLS] 그러나 민주당은 갈수록 강경해지고 있다. [SEP]['[CLS]', '그러나', '민', '##주', '##당', '##은', '갈', '##수', '##록', '강', '##경', '##해', '##지고', '있다', '.', '[SEP]']
```

[표 3-11] BERT 모델 토큰화 코드와 토큰화 결과

BERT 모델을 사용하려면 최소한 8GB의 GPU 메모리가 필요한데, [표 3-12]는 GPU 디바이스 검사 코드이고, [표 3-13]은 BertForSequenceClassification 모듈을 활용하여 중심 문장 분류에 활용하기 위해 모델을 생성하는 코드이다.

```
# GPU 디바이스 이름 구함
device_name = tf.test.gpu_device_name()
# GPU 디바이스 이름 검사
ifdevice_name == '/device:GPU:0':
    print('Found GPU at: {}'.format(device_name))
else:
    raise SystemError('GPU device not found')
```

[표 3-12] GPU 디바이스 검사 코드

```
# 분류를 위한 BERT 모델 생성
model = BertForSequenceClassification.from_pretrained(
                "bert-base-multilingual-cased", num_labels=2)
model.cuda()

# 옵티마이저 설정
optimizer = AdamW(model.parameters(),
                  lr = 2e-5, # 학습률
                  eps = 1e-8 # 0으로 나누는 것을 방지하기 위한 epsilon 값)
# 에폭수
epochs = 5
# 총 훈련 스텝 : 배치반복 횟수 * 에폭
total_steps = len(train_dataloader) * epochs
# 처음에 학습률을 조금씩 변화시키는 스케줄러 생성
scheduler = get_linear_schedule_with_warmup(optimizer,
num_warmup_steps = 0,
num_training_steps = total_steps)
```

[표 3-13] BERT 모델 생성 코드

2. 문장별 중요도 계산 및 중심 문장 추출

본 절에서는 앞 절의 과정을 통해 구성된 [그림 3-14]와 같은 데이터 프레임을 기준으로 문장별 중요도를 계산하여 중심 문장을 추출하고자 한다.

	newsid	sentence	position	similarity_title	similarity	posweight	score	label
0	newsid0	여야가 6일 임시국회 종료 이틀을 남기고 주요 쟁점법안 처리 문제를 일괄 타결했다.	0.0	0.767785	[0.99999994, 0.71113425, 0.725084, 0.83416057,...	0.063	0.974012	1
1	newsid0	이로써 지난해 12월 10일 개회 이래 4주 동안 야당의 본회의장 불법 점거와 국회...	0.1	0.650596	[0.71113425, 1.0, 0.67956364, 0.71445644, 0.65...	0.168	0.999474	1
2	newsid0	한나라당 홍준표 민주당 원혜영 선진과 창조의 모임 문국현 원내대표는 이날 국회 귀빈...	0.2	0.619279	[0.725084, 0.67956364, 1.0, 0.8107153, 0.72580...	0.151	0.999667	0
3	newsid0	여야는 이견이 없는 민생법안 등 100여 건을 이번 임시국회에서 협의 처리하되 아직...	0.3	0.678760	[0.83416057, 0.71445644, 0.8107153, 1.0000001,...	0.103	0.999568	0
4	newsid0	그러나 여야는 이날 합의에서 상당수 쟁점법안의 처리 시기는 물론이고 법안 상정 여부...	0.4	0.687531	[0.7482691, 0.6504138, 0.72580314, 0.8129954, ...	0.089	0.999704	0

[그림 3-14] 문장별 중요도 계산을 위한 속성

문장별 중요도를 계산하기 위해 [그림 3-14]에서 'similarity_title', 'similarity', 'posweight', 'score' 4개의 속성을 사용한다. 'similarity_title'은 각 문장의 제목과의 유사도를 의미하고, 'similarity'는 다른 문장과의 유사도, 'posweight'은 문장 위치별 가중치 적용값, 'score'는 LSTM 모델 분류 학습을 통해 추출된 확률값을 의미한다. 그 중 'similarity' 속성은 기준 문장과 기사 내 다른 문장과의 유사도를 구한 리스트 형태로 되어있는데, 모든 문장의 유사도를 사용하지 않고, 기준 문장을 중심으로 이후 2개 문장의 유사도만을 사용하도록 [표 3-14]의 코드를 활용하여 값을 변환하는 과정을 거친다.

```
def sum_sim(simlist): #sum(simlist[base:base+3])-1
    base = simlist.index(max(simlist))
    result = (sum(simlist[base:base+3])-1)
    return result
```

[표 3-14] 문장 유사도 변환 함수

문장별 중요도를 계산하기 위해 [표 3-15]와 같이 추출한 각 속성이 문장 중요도에 미치는 영향을 살펴보는 과정을 거친다. 제목과의 유사도, 위치 가중치, 문장 유사도, TS probability의 4가지 속성을 각각 적용하고 또한 속성들을 결합하여 적용해본 결과 4가지 속성을 모두 결합한 방법의 중심 문장 추출 성능이 가장 높게 나온 것을 확인하였다. 따라서 문장 중요도 점수를 계산하는데 4가지 속성을 모두 결합하여 적용한다.

	LSTM	BERT
① 제목과의 유사도	0.2798	
② 위치 가중치	0.2394	
③ 문장 유사도	0.2398	
④ TS probability	0.2084	0.2074
①+②	0.2972	0.2972
①+②+③	0.3192	0.3087
①+②+④	0.3192	0.3165
①+②+③+④	0.3271	0.3245

[표 3-15] 문장 중요도 계산 방법별 중심 문장 추출 성능 비교

다음 식(5)은 추출된 속성값을 적용하여 문장 중요도 점수를 계산하는 식이다.

$$SC_i = Sim1_i + PW_i + TP_i + Sim2_i \quad (5)$$

- SC_i : i번째 문장 중요도 점수
- $Sim1_i$: i번째 문장과 제목과의 유사도
- PW_i : I번째 문장의 문장 위치별 가중치
- TP_i : I번째 문장의 중심 문장 확률값
- $Sim2_i$: i번째 문장 이후 2문장의 유사도

위 식 (5)에서 문장별 중요도 점수 SC_i는 각 문장과 제목과의 유사도 $Sim1_i$, 문장 위치별 가중치 PW_i, 중심 문장 확률 TP_i, 기준 문장과 인접한 2개 문장의 유사도 $Sim2_i$를 모두 결합한 점수이다. 이렇게 구해진 문장별 중요도 점수를 기준으로 순위를 매겨 중심 문장을 추출한다.

Ⅳ. 실험 및 결과

본 장에서는 실험에 사용된 텍스트를 수집하는 과정과 실험에서 사용한 데이터 셋에 대하여 설명하고 본 논문에서 제안한 중심 문장 추출 방법의 효용성을 입증하기 위해 분류성능평가지표를 활용하여 정확도(Accuracy), 재현율(Recall), 정밀도(Precision)를 측정하여 성능을 평가하고 중심 문장 추출 성능은 실제 추출 정확도와 ROUGE-1 스코어의 정밀도(Precision)로 평가한다.

본 연구에서 실험에 사용한 환경은 [표 4-1]과 같다.

구분	내용	
실행 환경	Google Colaboratory	
	RAM	16GB
	DISK	100GB
	GPU	Tesla K80
딥러닝 프레임워크	Tensorflow, PyTorch	
사용 언어	python3	

[표 4-1] 실험 환경

A. 데이터 수집

본 논문에서 제안한 중심 문장 추출 방법을 실험하기 위해 국립국어원의 말뭉치 데이터를 사용하였다. 사용된 말뭉치 데이터 중 '신문 기사 말뭉치'는 뉴스 기사 id별 기사 본문으로 363개의 json 파일로 구성되어있고, '문서 요약 말뭉치'는 '신문 기사 말뭉치'에서 임의로 선택한 기사에 대한 중심 문장(topic)과 요약문(summary)으로 json 형태로 구성되어있다. [표 4-2]는 json 형태로 서로 분리된 데이터를 하나로 통합하는 과정의 파이썬(Python) 코드이다.

```
//요약 말뭉치
with open('./말뭉치/NIKL_SC.json', encoding="utf-8") as json_file:
    json_data = json.load(json_file)

data = json_data['data']
# len(data) #4389
news_ids = []
titles = []
topic_sentences = []
summary_sentences = []

for i in range(len(data)):
    if i not in [20, 82]: #누락 기사 제외
        news_ids.append(data[i]['document_id'])
        titles.append(data[i]['head'])
        topic_sentences.append(data[i]['topic_sentences'])
        summary_sentences.append(data[i]['summary_sentences'])
contents = []
ids = []
for news_id in news_ids:
    fname =  './말뭉치/' + news_id.split('.')[0]
    filename = fname + '.json'
    with open(filename, encoding="utf-8") as json_file:
        json_data = json.load(json_file)
        # 뉴스 기사 id를 기준으로 기사 본문과 요약 문장을 결합
        for doc in json_data['document']:
            if doc['id'] == news_id:
                contents.append(doc['paragraph'])
                ids.append(news_id)
```

[표 4-2] 데이터 셋 구성 코드

B. 데이터 셋

본 절에서는 실험에 사용되는 데이터 셋에 대해 설명한다. 실험을 위해 수집된 데이터는 문장 단위로 분리하기 위해 [표 4-3] 코드를 적용한다. 해당 코드는 한국어 문장 토큰화 도구인 kss 모듈을 임포트하여 기사문 정제 작업 후 문장 단위로 분리하는 작업을 수행하는 코드이다. 그 결과, [표 4-4]와 같이 전체 기사 4,347개를 문장 단위로 분리한 전체 문장 수는 96,332개이며, 이 중 딥러닝 모델 학습 데이터는 22,852개, 추출 성능 비교에 사용된 데이터는 86,919개 이다.

```
import kss
def sentTokenize(s):
    text = cleanText(s)
    sentences = kss.split_sentences(text)
    return sentences
```

[표 4-3] 문장 단위 분리 함수

구분	개수
기사문 개수	4,347개
전체 문장 수	96,332개
추출 성능 실험 대상 문장 수	86,919개
딥러닝 모델 학습 사용 문장 수	22,852개

[표 4-4] 문장 단위 분리 결과

[표 4-5]는 문장 단위로 분리된 것을 임베딩하기 위해 Konlpy 패키지에 있는 Okt 한국어 형태소 분리 모듈을 사용하여 명사 중심으로 토큰화를 진행하는 함수이다.

```
from konlpy.tag import Okt
okt = Okt()
def tokenization(sentences):
    sent_tokens = []
    for sentence in sentences:
        tokens = okt.nouns(sentence)
        sent_tokens.append(tokens)
    return sent_tokens
```

[표 4-5] 문장별 토큰화 과정

[표 4-6]은 사전 학습된 ELMo 한국어 모델을 로드하여 문장 벡터를 구하는 과정을 수행하기 위한 코드이다.

```
# ELMo 한국어 모델 로드
elmo = hub.load("https://tfhub.dev/google/nnlm-ko-dim128-with-normalization/2")
embedding_dim = 128
zero_vector = np.zeros(embedding_dim)
# 문장 벡터
def calculate_sentence_vector(sentence):
    if len(sentence) != 0:
        s = sum(elmo(sentence))
        return s.numpy()
    else:
        return zero_vector
# 기사문 전체에 대해서 문장 벡터를 반환
def sentences_to_vectors(sentences):
        return [calculate_sentence_vector(sentence) for sentence in sentences]
```

[표 4-6] 임베딩 모델 로드 및 문장 임베딩 함수

C. 실험 평가 및 분석

1. 실험 평가 방법

a. 분류성능평가지표

본 절에서는 분류성능평가지표인 정확도(Accuracy), 재현율(Recall), 정밀도(Precision)를 이용하여 중심 문장 판별에 대한 성능을 평가하고 본 연구 결과를 TextRank 알고리즘과 비교 분석한다. [그림 4-1]은 분류성능평가지표로 분류 성능 확인을 위한 혼동행렬(Confusion Matrix)이며 식 (6)은 정확도(Accuracy), 재현율(Recall), 정밀도(Precision)를 계산하는 식이다.

		Predicted	
		Negative	Positive
Class	Negative	TN	FP
	Positive	FN	TP

[그림 4-1] Confusion Matrix

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Recall = \frac{TP}{TP + FN} \qquad (6)$$

$$\mathrm{Pr}ecision = \frac{TP}{TP + FP}$$

b. ROUGE 요약 모델 성능 평가

ROUGE(Recall-Oriented Understudy for Gisting Evaluation)는 Chin-Yew Lin에 의해 고안된 텍스트 자동 요약, 기계 번역 등 자연어 생성 모델의 성능을 평가하기 위한 지표로, 모델이 생성한 요약본(system_summary)을 사람이 미리 만들어 놓은 참조본(reference_summary)과 대조해 성능 점수를 계산한다. 식 (7)은 ROUGE의 정밀도(Precision)와 재현율(Recall)을 구하는 식이다.

$$Recall = \frac{Number\_of\_overlapped\_words}{Total\_words\_in\_reference\_summary}$$

$$\Pr ecision = \frac{Number\_of\_overlapped\_words}{Total\_words\_in\_system\_summary} \quad (7)$$

재현율(Recall)은 참조본(reference_summary)을 구성하는 단어 중 몇 개의 단어가 모델이 생성한 요약본(system_summary)의 단어들과 겹치는지를 보는 점수이고, 정밀도(Precision)는 재현율(Recall)과 반대로 모델이 생성한 요약본(system_summary) 중 참조본(reference_summary)과 겹치는 단어들이 얼마나 많이 존재하는지를 보는 것이다.

ROUGE 성능평가는 기준 단어 수에 따라 ROUGE-N, ROUGE-S, ROUGE-L로 나뉘는데, ROUGE-1은 시스템 요약본과 참조 요약본 간 겹치는 unigram의 수를 보는 지표이며, ROUGE-2는 시스템 요약본과 참조 요약본 간 겹치는 bigram의 수를 보는 지표이다[17]. 본 연구에서는 Okt 토큰화를 기준으로 문장을 토큰화하여 ROUGE-1을 적용하여 재현율을 평가 척도로 사용하였다.

2. 실험 결과 분석

실험 대상 문장 86,919개 중 정답 라벨의 비율을 1:1로 맞추어 최종적으로 중심 문장 분류 학습에 사용된 문장은 22,852개이다. 이 중 80%를 훈련 데이터, 20%를 테스트 데이터로 분리하였을 때 [표 4-7]과 같이 테스트 실험 문장은 4,558개로 중심 문장 2,118개와 일반 문장 2,440개로 구성되었다.

구분	개수
테스트 실험 문장	4,558
중심 문장	2,118
일반 문장	2,440

[표 4-7] 중심 문장 분류 모델 테스트 데이터의 수

[표 4-8]은 [표 4-7] 테스트 실험 문장을 대상으로 LSTM 모델과 BERT 모델로 분류 예측한 결과이다.

구분	LSTM	BERT
중심 문장	2,308	2,265
일반 문장	2,201	2,205

[표 4-8] 모델별 중심 문장 예측 결과

[표 4-9], [표 4-10]는 각각 LSTM 모델, BERT 모델의 중심 문장 분류 예측 결과를 혼동행렬로 작성하여 그에 따른 정확도, 재현율, 정밀도 계산한 것이다. 본 연구에서는 중심 문장을 숫자 1로, 일반 문장을 숫자 0으로 지정하여 실험하였다.

	Predicted Positive	Predicted Negative
True Positive	2038	80
True Negative	239	2201
정확도	$\frac{2038+2201}{2038+239+80+2201}$	93%
재현율	$\frac{2038}{2038+80}$	96.22%
정밀도	$\frac{2038}{2038+239}$	89.5%

[표 4-9] LSTM Confusion Matrix

	Predicted Positive	Predicted Negative
True Positive	2265	38
True Negative	63	2205
정확도	$\frac{2065+2205}{2265+38+63+2205}$	97.79%
재현율	$\frac{2265}{2265+38}$	98.27%
정밀도	$\frac{2265}{2265+63}$	97.33%

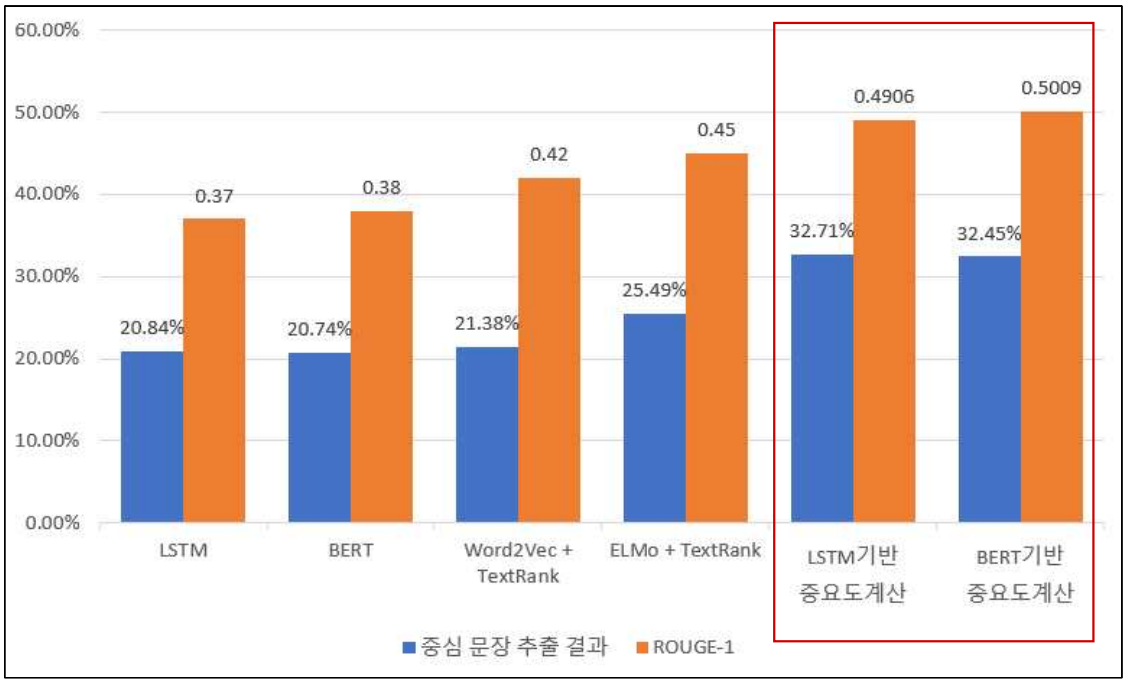
[표 4-10] BERT Confusion Matrix

[표 4-9]와 [표4-10]은 중심 문장 확률(Topic Sentence Probability)을 구하기 위해 적용한 LSTM 기반 중심 문장 판별 방법과 BERT 기반 중심 문장 판별 방법에 대한 분류 성능 지표이다. 두 방법 모두 93% 이상으로 높은 성능을 보인다. 이로써 딥러닝 기반 모델 학습을 통해 중심 문장의 속성을 추출할 수 있음을 확인하였다.

[표 4-11]은 요약 평가 지표로 실제 중심 문장 추출 정확도 및 ROUGE-1 스코어를 계산한 결과이다. 실험 결과 본 논문에서 제안한 딥러닝 기반의 문맥을 고려한 문장 임베딩과 문장 중요도 계산에 필요한 문장 특징을 추출하여 가중치를 적용한 모델이 기존 TextRank 보다 성능이 개선된 것을 볼 수 있다.

요약 평가지표	중심 문장 추출 결과	ROUGE-1
LSTM	20.84%	0.37
BERT	20.74%	0.38
Word2Vec + TextRank	21.38%	0.42
ELMo + TextRank	25.49%	0.45
LSTM기반 중요도 계산	32.71%	0.49
BERT기반 중요도 계산	32.45%	0.50

[표 4-11] 실험 방법별 중심 문장 추출 결과 비교



[그림 4-2] 실험 방법별 중심 문장 추출 결과 비교

제목	LG 휘센 기술 경쟁사에 넘어갈 뻔	비고
원본 기사문	**'LG전자 휘센에어컨에 사용된 금속표면처리기술 등의 첨단기술을 중국 경쟁업체에 유출하려던 국내 벤처기업의 전 대표와 임원 연구원 등 4명이 검찰에 적발됐다.'** (중략) '검찰에 따르면 미국 박사 일본 대학 교수 KIST 선임연구원 등의 경력을 지닌 고 씨는 KIST가 특허기술을 상용화하기 위해 세운 P사의 대표를 맡고 있었지만 평소 성과급에 대한 불만이 컸다.' **'고 씨는 부하 연구원들과 함께 크게 한탕 하기로 하고 지난해 10월부터 올 4월까지 회사를 차례로 그만두면서 에어컨 기술 관련 자료가 담긴 노트북 컴퓨터를 반납하지 않거나 외장메모리에 자료를 저장해 갖고 나오는 방법으로 나노 파우더 등 네 가지 첨단기술을 조직적으로 유출했다.'** **'금속을 나노 크기로 쪼개는 이 기술은 아직 상용화되진 않았지만 세계에서 한국이 유일한 기술보유국일 정도로 미래가치가 큰 첨단기술이다.'** (중략) '하지만 국가정보원과 검찰이 범죄첩보를 입수하면서 이들의 범행은 막을 내렸다.', '이 부장검사는 기술이 중국 업체에 유출됐다면 LG전자가 매출 감소 등으로 1200억 원 가량의 손실을 볼 것으로 추산됐다고 말했다.'	
제안 방법 적용 중심 문장 추출 결과	**'LG전자 휘센 에어컨에 사용된 금속표면처리기술 등의 첨단기술을 중국 경쟁업체에 유출하려던 국내 벤처기업의 전 대표와 임원 연구원 등 4명이 검찰에 적발됐다.',** **'고 씨는 부하 연구원들과 함께 크게 한탕 하기로 하고 지난해 10월부터 올 4월까지 회사를 차례로 그만두면서 에어컨 기술 관련 자료가 담긴 노트북 컴퓨터를 반납하지 않거나 외장메모리에 자료를 저장해 갖고 나오는 방법으로 나노 파우더 등 네 가지 첨단기술을 조직적으로 유출했다.',** **'금속을 나노 크기로 쪼개는 이 기술은 아직 상용화되진 않았지만 세계에서 한국이 유일한 기술보유국일 정도로 미래가치가 큰 첨단기술이다.'**	3/3
비교 실험 TextRank 적용 중심 문장 추출 결과	**'LG전자 휘센 에어컨에 사용된 금속표면처리기술 등의 첨단기술을 중국 경쟁업체에 유출하려던 국내 벤처기업의 전 대표와 임원 연구원 등 4명이 검찰에 적발됐다.'** '이 부장검사는 기술이 중국 업체에 유출됐다면 LG전자가 매출 감소 등으로 1,200억 원가량의 손실을 볼 것으로 추산됐다고 말했다.' **'고 씨는 부하 연구원들과 함께 크게 한탕 하기로 하고 지난해 10월부터 올 4월까지 회사를 차례로 그만두면서 에어컨 기술 관련 자료가 담긴 노트북 컴퓨터를 반납하지 않거나 외장메모리에 자료를 저장해 갖고 나오는 방법으로 나노 파우더 등 네 가지 첨단기술을 조직적으로 유출했다.',**	2/3

[표 4-12] 실험 결과 추출 문장 비교

[표 4-12]는 원본 기사문에 대해 제안한 방법과 비교 실험 방법인 TextRank를 적용하여 중심 문장을 추출한 결과를 비교한 것이다. 원본 기사문에 대해 제시된 중심 문장은 3문장으로 비교 실험인 TextRank 방법을 적용한 경우는 2문장을 동일하게 추출하였으나, 제안한 방법은 3문장을 모두 추출한 것을 알 수 있다.

Ⅴ. 결론 및 향후 연구

본 논문에서는 ELMo, BERT 사전학습 모델 및 LSTM 모델을 활용하여 중심 문장 여부를 판별하는 확률값과 문장의 중요도에 영향을 주는 특징을 결합하여 중심 문장을 추출하는 방법을 제안하였다.

제안하는 방법은 문장 임베딩을 하기 위해 사전 학습된 한국어 ELMo 모델을 사용하여 단어의 의미뿐만 아니라 동음이의어 문제까지 고려한다. 그리고 BERT와 LSTM 모델을 활용하여 문맥의 흐름 통해 문장 자체가 중심 문장일 확률을 구할 수 있다. 이와 함께 문서 내에서의 각 문장의 중요도에 영향을 미치는 요소인, 제목과의 유사도, 다른 문장과의 유사도, 문장별 위치 등을 고려하여 문장별 스코어를 계산하여 중심 문장을 추출하였다. 제안한 방법에 대한 실험 결과는 중심 문장 분류성능은 중심 문장 2,118개, 일반 문장 2,440개 중에서 중심 문장이라고 예측한 문장은 2,308개, 중심 문장이 아니라고 예측한 문장은 2,201개이다. 이에 따라 분류성능평가지표에 따르면 정확도는 93%, 재현율은 96.22%, 정밀도는 89.5%로 높은 분석 결과가 나온 것으로 LSTM 및 BERT 모델의 분류 예측값을 중심 문장일 확률이라고 간주할 수 있다. 최종적으로 중심 문장 추출 성능을 ROUGE-1 정밀도와 4,000개의 기사문에서 실제 중심 문장을 추출한 결과는 ROUGE-1 스코어는 0.50, 실제 추출 결과는 32.45%가 나왔다.

본 연구에서 제안한 방법을 기존 TextRank 알고리즘과 비교 분석한 결과 제안하는 방법의 중심 문장 추출 성능이 10% 정도 개선된 것을 확인할 수 있었다.

본 논문의 향후 연구로는 사전 학습된 언어 모델로 판별한 중심 문장 결과를 중심 문장 추출기에 문서 전체를 입력하여 자동으로 중심 문장을 추출하는 방법을 연구하고자 한다. 또한 이를 기반으로 비지도 학습 기반의 중심 문장 추출 방법 및 뉴스 기사뿐만 아니라 학습 콘텐츠에도 이를 적용하여 교육과정 내 관련 단원에서 활용할 수 있는 프로그램 개발에 적용하고자 한다.

참고 문헌

[1] 한국경제 신문 기사, https://www.hankyung.com/it/article/2019091912151

[2] 네이버, https://m.help.naver.com/support/contents/contentsView.help?contentsNo=8126

[3] 고영중, 박진우, 서정연. “문장 중요도를 이용한 자동 문서 범주화”, 정보과학회논문지 : 소프트웨어 및 응용, 제 29권, 제 6호, pp. 417-424, 2002.

[4] 김희찬. “의미적으로 확장된 문장 간 유사도를 이용한 한국어 텍스트 자동 요약”, 국내석사학위논문 숭실대학교 대학원, 2015.

[5] 김지희. “중심내용 찾기 활동을 통한 설명문 읽기의 효과 연구”, 국내석사학위논문 한국교원대학교 교육대학원, 2011.

[6] 박서희. “BERT Transfer Learning을 활용한 스토리 텍스트 감정 인식”, 국내석사학위논문 성균관대학교 일반대학원, 2020.

[7] 이종권. “문장임베딩과 딥러닝기법을 활용한 관세 품목분류문서의 자동 HS분류 연구”, 국내석사학위논문 한밭대학교 창업경영대학원, 2020.

[8] Peters M. E. et al. “Deep contextualized word representations. arXiv 2018”, arXiv preprint arXiv:1802.05365 (1802).

[9] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

[10] Page L., Brin S., Motwani R., & Winogra ,T., “The PageRank citation ranking : Bringing order to the web”, 2009.

[11] Mihalcea R., & Tarau P., “TextRank : Bringing order into text”, In Proceedings of EMNLP, vol. 4, no. 4, pp. 275, July, 2004.

[12] 홍진표, 차정원. “TextRank 알고리즘을 이용한 한국어 중요 문장 추출”, 한국정보과학회 학술발표논문집, 제 36권, 제 1C호, pp. 311-314, 2009.

[13] 김원우. “중요 문장 추출 및 추상 요약을 통한 cQA시스템 질문 요약 성능 개선”, 국내석사학위논문 광운대학교 대학원, 2019.

[14] Y. Kim, “Convolutional neural networks for sentence - 56 - classification”, Proc. of the 2014 Conference on EMNLP, pp. 1746-1751, 2014.

[15] 전재원. “딥 러닝 기반의 2단계 한국어 문서 요약”, 국내석사학위논문 강원대학교

대학원, 2020.

[16] 정풀잎, 안현철. “문장 위치를 고려한 고객 리뷰 감성 분석 모형”, 인터넷전자상거래연구, 제 19권, 제 1호, pp. 167-186, 2019.

[17] ROUGE:, https://en.wikipedia.org/wiki/ROUGE_(metric)