공학석사학위논문

문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델

Text Classification Model using RNN-LSTM Two-Step with Sentences-Reflecting Learning Results.

충 북 대 학 교 대 학 원

빅데이터 협동과정 빅데이터 전공

최 우 석

2021 년 2월

공학석사학위논문

문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델

Text Classification Model using RNN-LSTM Two-Step with Sentences-Reflecting Learning Results.

지도교수 최 상 현

빅데이터 협동과정 빅데이터 전공

최 우 석

이 논문을 공학석사학위 논문으로 제출함.

2021 년 2월

본 논문을 최우석의 공학석사학위 논문으로 인정함.

심 사 위 원 장 류 관 희 ㊞

심 사 위 원 최 상 현 ㊞

심 사 위 원 이 정 환 ㊞

충 북 대 학 교 대 학 원

2021 년 2월

차 례

Text Classification Model using RNN-LSTM Two-Step with Sentences-Reflecting Learning Results.

Choi, Woo-Seok

Department of BigData

Graduate School, Chungbuk National University

Cheongju, Korea

Supervised by Professor Choi, Sang-Hyun

Abstract

Text classification is a task where text like a word, a sentence, and a paragraph is classified by category, and it is one of the unstructured data analysis.

Many researchers have attempted to solve the text classification problem using deep learning. In particular, LSTM, that is designed for the past events to influence the future outcomes, was suitable for approaching text classification problems. However, LSTM has a limitation where the performance decreased as time steps number, the number of average words used, increased. This is because people also write meaningless sentences that don't address the purpose of the article, apart from the problems of long-term dependencies.

To solve these problems, this research proposed 'text classification model using RNN-LSTM two-step with sentences-reflecting learning results'. This model improved the classification accuracy of long documents by proceeding sentence-level learning with RNN, and then reflecting the result, and then applying it to LSTM again.

Key words: Deep Learning, RNN, LSTM, Text Mining, Text Classification

* A thesis for the degree of Master in February 2021.

표 차례

그림 차례

Ⅰ. 서론

1. 연구의 배경 및 목적

텍스트 분류(Text Classification)란, 단어 또는 문장, 지문 등 텍스트가 어느 카테고리에 해당하는지를 분류하는 작업을 의미하며, 비정형 데이터 분석 중 하나이다(권철민 2019). 텍스트 분류는 입력받은 텍스트가 긍정적인지 또는 부정적인지 분류하는 감성 분석, 텍스트를 미리 지정한 카테고리로 분류하는 카테고리 분석, 텍스트가 내포하고 있는 의도를 파악하는 의도 분석 등으로 구분할 수 있다(Mouthami, K. et al. 2013).

인터넷의 발달은 계속해서 가속화되고 있으며, 스마트폰의 보급, SNS의 활성화, 사물인터넷의 확대로 데이터의 양은 급격히 증가하고 있다(김동완 2013). 특히 페이스북, 트위터, 카카오 등 SNS에서 수집되는 텍스트 데이터는 마케팅 분야에서의 핵심이 될 정도로 중요성이 높아지고 있으며, 이에 따라 텍스트 분석에 대한 연구도 활발하게 진행되고 있다(김정숙 2012).

과거에는 텍스트로부터 카테고리 또는 감성, 의도를 분류하기 위하여 목적에 맞는 사전을 구축하는 방식을 많이 활용하였다. 하지만, 사전을 구축하는 작업은 매우 어려우며 긴 시간이 소요된다는 단점이 존재한다. 게다가 영어의 경우에는 감성사전, 불용어사전 등 체계적이고 다양한 목적에 맞는 사전들이 이미 존재하지만, 한국어는 아직까지 텍스트 분류를 위한 사전이 미흡한 상황이다(길호현 2018). 그 때문에 연구자들은 데이터마이닝(DataMining) 기법으로 많은 양의 데이터를 학습하여 텍스트 사전을 자동으로 구축하려는 방안을 연구하고 있으며(장경애 등 2015), 또한 최근에는 딥러닝(Deep Learning)을 활용하여 별도의 사전 구축 작업 없이 텍스트 분류 문제를 해결하고자 하는 연구도 활발하게 진행되고 있다.

텍스트 데이터는 본질적으로 연속성(Sequence)을 갖고 있다. 즉, 하나의 문서는 여러 개의 문장으로 이루어져 있으며, 하나의 문장은 여러 개의 단어로 구성된다. 게다가 글의 목적에 따라 사용되는 단어가 달라지며, 똑같은 단어라도 사용되는 순서에 따라 글의 의도가 달라질 수 있다(김양훈 등 2016). 이러한 특징은 과거의 사건이 미래의 결과에 영향을 줄 수 있게끔 설계된 순환신경망(Recurrent Neural Network)에 적합하였으며, 실제로 국내·외에서 RNN과 LSTM을 활용한 텍스트 분류 연구가 많이 진행되었다.

하지만, LSTM 알고리즘으로 텍스트를 분류하는 것에도 한계가 존재한다. 일반적으로 LSTM은 RNN에서 발생하는 장기의존성 문제를 해결하기 위해 고안된 알고리즘이다. 하지만, 기대와 다르게 LSTM을 텍스트 분류에 적용하고자 할 때 Time Steps 수, 즉 문서에 사용된 평균 단어의 개수가 높아질수록 성능이 떨어지는 문제가 발생한다(Du, J. et al. 2020). 이는 장기의존성 문제와는 별도로 사람들이 글을 쓸 때, 글의 목적만을 다루지 않으며 의미 없는 문장이 같이 쓰이기 때문이다.

본 연구에서는 이러한 문제점을 보완하기 위하여 '문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델'을 고안하였다. 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델은 텍스트에서 의미 없는 문장을 자동으로 제외함으로써, 의미 있는 문장만을 가지고 텍스트를 분류할 사용할 수 있도록 설계한 텍스트 분류 모델이다.

2. 연구의 구성

본 연구의 구성은 다음과 같다.

제 1장 서론에서는 텍스트 분류가 무엇인지 정의를 내리고, 텍스트 분류가 발전하게 된 배경과 흐름 그리고 본 연구를 진행하게 된 목적을 기술하였다.

제 2장 이론적 배경에서는 지금까지 진행되어온 기계학습 기반의 텍스트 분류 연구와 딥러닝 기반의 텍스트 분류 연구에 대한 동향을 살펴보았으며, 또한 데이터마이닝에 대한 이론을 기술하였다.

제 3장 연구 방법에서는 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델을 설계하기 위해 사용한 데이터 전처리 방법, 그리고 RNN과 LSTM 방법론을 정리하였다.

제 4장 데이터 설명 및 전처리에서는 수집한 텍스트 데이터에 대한 설명과 기초분석을 기술하였으며, 카테고리 재분류, 문장 분리, 텍스트 처리, 형태소 분석, 불용어 처리, 토큰화, 학습셋 구축 등 텍스트 데이터에 대한 7가지 전처리 과정에 대해 상세히 기술하였다.

제 5장 연구 결과에서는 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델에 대한 전체적인 프로세스를 설명하고, 각 단계의 과정과 성능 결과, 그리고 본 연구의 우수성을 기술하였다.

제 6장 결론 및 한계에서는 본 연구에서 진행한 텍스트 분류 결과에 대한 전반적인 요약과 연구의 의의 및 한계점을 제시하였으며 향후 연구를 제안하였다.

II. 이론적 배경

1. 선행 연구

(1) 기계학습(Machine Learning) 기반의 텍스트 분류

과거에는 전문가의 의견에 따라 수작업으로 텍스트 분류를 할 수 있었으나, 인터넷의 발달로 생성되는 문서의 양이 기하급수적으로 늘어남에 따라 더 이상 전문가의 주관만으로는 객관적인 분류가 어려워졌다. 이에 연구자들은 기계학습 기반의 빠르고 정확한 텍스트 분류 방법 고안하였다(김판준 2018).

김영욱 등(2018)은 유사한 의미를 가진 단어들을 군집화하는 알고리즘인 토픽모델링(Topic Modeling)을 활용하여 후보 키워드를 선정한 후, 문서에 존재하는 단어 간의 연관 규칙을 찾는 연관성규칙분석(Apriori)으로 핵심단어를 추출하여 문서를 분류하는 방법을 제시하였다.

나성희 등(2016)은 스피어만 상관성 분석과 특징가중치를 결합한 카테고리 분류 모델을 제안하였다. 이때 사용한 특징가중치란, 단문 텍스트에서 나타나는 특징을 정의하고 각 특징별 많이 등장하는 단어에 대해 부여한 가중치를 의미한다.

하지만, 텍스트 분류를 할 때 문서에 포함된 단어의 출연 여부나 빈도를 원-핫 벡터(One-Hot vector)로 표현한 후 텍스트 분류를 진행하는 BOW(Bag-of-Word) 방법으로는 단어의 순서가 제거되어 맥락을 반영할 수 없다는 한계가 존재하였으며, 이를 해결하기 위하여 Word2Vec과 같이 고차원으로 단어를 벡터화하여 맥락까지 고려할 수 있는 연구가 발전하였다(박성수・이건창 2018).

백두현 등(2020)은 TF-IDF와 Doc2Vec으로 단어를 벡터화한 한 후, 확률적 경사하강법(stochastic gradient descent)과 로지스틱회귀(logistic regression), SVM(support vector machine) 등 3가지 알고리즘을 활용하여 각기 다른 세 가지 정신질환에 대한 진단을 자동 분류하는 연구를 진행하였으며, 고차원으로 텍스트를 벡터화하는 방법이 텍스트 분류의 성능을 높일 수 있음을 시사하였다.

(2) 딥러닝(Deep Learning) 기반의 텍스트 분류

인공지능과 딥러닝(Deep Learning)에 대한 관심이 높아짐에 따라, 텍스트 분야에서도 딥러닝을 통해 분류 성능을 높이고자 하는 연구가 활발히 진행되고 있으며, 특히, 과거의 사건이 미래의 결과에 영향을 줄 수 있게끔 설계된 RNN과 LSTM 등 순환신경망을 활용한 텍스트 분류 연구가 많이 진행되었다.

Wang 등(2018)은 Word2Vec을 사용하여 문서 내의 단어를 고차원으로 벡터화한 후, 이를 LSTM(Long Short Term Memory)의 입력값으로 활용하여 텍스트 분류를 진행하였다.

박상민 등(2018)은 특정 도메인에 대한 감성사전을 빠르게 구축하기 위해 Bi-LSTM(Bidrectional Long Short Term Memory) 기반의 범용 한국어 감성사전 구축 방안을 제시하였다. 해당 연구는 연구하고자 하는 도메인에 맞는 감성사전을 자동으로 구축할 수 있는 방안을 고안하였다는 데에 의의가 있다.

최근에는 CNN(Convolutional neural network) 기반의 심층신경망 구조가 이미지 분류뿐만 아니라 자연어 처리 분야 중 하나인 문서 분류에서도 좋은 성능을 발휘한다는 것이 입증되면서 단어를 벡터(vector)로 표현하는 방법인 Word2Vec과 CNN을 이용한 문장 분류 방법이 연구가 많이 진행되었다(김정미・이주홍 2017).

이현상 등(2018)은 악성댓글의 짧고 변칙적인 특성상 문장의 시퀀스보다 비선형적인 특성을 제대로 학습할 수 있는 CNN 알고리즘을 적용하여 텍스트 분류 모델을 설계하였으며, RNN 알고리즘보다 성능이 좋음을 증명하였다.

다만, 김도우 등(2017)은 Word2Vec과 CNN 기반의 한국어 문서 분류를 진행함으로써 CNN을 이용한 텍스트 분류는 짧은 문장에 한하여 효과적이며, 길이가 길고 복잡한 텍스트에서는 성능이 저하되는 한계가 있음을 검증하였다.

Liu 등(2018)은 전체 문서에서 무작위로 여러 개의 블록으로 나누고 각각의 블록마다 CNN 적용하여 특징을 추출한 후, 다시 RAM(Recurrent attention model)를 적용함으로써 긴 문서에서의 텍스트 분류 성능을 높이고자 하였으

며, 박효연과 김경재(2019)는 CNN과 LSTM을 조합하여 두 개의 알고리즘에 대한 장점을 결합함으로써 텍스트 분류의 성능을 높이고자 하였다.

언어에서는 어휘적 중의성이란 특성이 존재한다. 어휘적 중의성이란, 동음이의어, 다의어와 같이 한 개의 단어가 2개 이상의 의미로 해석이 가능한 경우를 의미한다. 문서의 길이가 길어지게 되면 어휘적 중의성은 더욱더 많아지게 되는데 이러한 특성은 텍스트를 분류하고자 할 때, 정확도를 떨어뜨리는 고질적인 원인이다(기호연・신경식 2020). 본 연구에서는 RNN으로 문장단위의 학습을 진행한 후, 그 결과를 반영하여 다시 LSTM에 적용함으로써 위와 같은 문제점을 보완할 방안을 고안하였다.

아래 <표 1>은 선행연구를 정리한 표이다.

<표 1> 선행연구 요약

구분	저자	요약
머신러닝	나성희 외 2016	스피어만 분석과 특징가중치 기반의 단문 텍스트 분류 모델 설계
	김영욱 외 2018	토픽모델링과 연관규칙을 통한 동적 문서 분류 진행
	박성수 외 2018	Word2Vec과 앙상블 기반의 효율적 한국어 감성 분류 진행
	백두현 외 2020	Doc2Vec과 3가지 머신러닝 기법으로 정신질환 분류 진행
딥러닝	김정미.이주홍 2017	Word2Vec과 LSTM을 결합하여 긴 문서에서의 텍스트 분류 성능을 높이고자 함
	김도우 등 2017	Word2Vec과 CNN을 결합하여 한국어 텍스트 분류 진행
	JH Wang 외 2018	LSTM 기반의 짧은 문장에서의 감성 분류 모델 설계
	L Liu 외 2018	긴 문서에서 높은 분류 정확도를 발휘할 수 있는 CNN-RAM 결합 모델 설계
	박상민 등 2018	Bi-LSTM 기반의 범용 한국어 감성사전(KNU) 구축
	이현상 등 2018	CNN기반의 악성댓글 분류 모델 설계
	박효연.김경재 2019	CNN과 LSTM을 결합한 영화리뷰 감성 분류 모델 설계
	기호연.신경식 2020	어휘적 중의성 문제를 해결하기 위한 양방향 LSTM 기반의 텍스트 분류 모델 설계

2. 데이터마이닝(Data Mining)

데이터마이닝(Data Mining)이란, 대규모 데이터에서 가치 있는 정보를 추출하는 과정을 의미한다. 광물을 찾아낸다는 단어인 'Mining'을 사용하게 된 이유는 마치 광산에서 석탄을 얻는 것처럼 데이터에서 정보를 추출하는 과정이 숨겨진 가치를 발견해낸다는 특징을 갖고 있기 때문이다(Berry and Linoff 2004).

Berry와 Linoff(2004)는 데이터마이닝을 "의미 있는 패턴과 룰을 발견하기 위해 자동화되거나 반자동화된 방법으로 대용량의 데이터를 수집하고 분석하는 과정"으로 정의하였으며, Gartner Group(2014)에서는 "통계적·수학적 기술뿐만 아니라 패턴인식 등 규칙 기반 기술을 이용하여 데이터를 조사함으로써 의미를 도출하는 과정"으로 정의하였다.

<표 2>는 통계학과 데이터마이닝의 특성을 비교한 표이다.

<표 2> 통계학과 데이터마이닝의 비교

전통적인 통계학	데이터마이닝
• 현실에 적용하기에 부적합한 가정 → 정규분포, 선형성, 등분산성 등 • 제안된 가설에 대한 검증이 주 목표 • 선형성에 기반을 두고 있음	• 잡음 데이터에 대한 가정이 없음 • 미래를 예측하는 데 주 목적 • 비선형성에 기반을 두고 있음 • 예측 성과가 통계기법보다 우수한 것으로 많은 실증 연구에서 검증되었음

데이터마이닝과 통계학은 데이터를 가지고 분석한다는 관점에서 유사하나 목적에 있어서 차이가 있다. 기존의 통계학에서는 모형이나 가설을 세우고 이를 검증하는 과정에 초점을 맞추고 있으나, 데이터마이닝은 특정한 가정이 없으며 미래 결과값 예측의 성능을 높이는 것 자체를 목적으로 하고 있다(Shmueli, G. et al. 2011).

데이터마이닝은 데이터의 구성에 따라 지도학습(Supervised Learning)과 비지도학습(Unsupervised Learning)으로 나눌 수 있다. 지도학습이란, 훈련용 데이터에 라벨링, 즉 정답이 포함된 데이터를 가지고 학습하는 방법을 의미한다. 분류(Classification)와 회귀(Regression)가 대표적인 지도학습 작업이다. 반면 비지도학습은 훈련용 데이터에 라벨링이 되어있지 않으며, 데이터의 분포나 특성만을 가지고서 학습하는 방법을 의미한다. 본질적으로 군집분석(Cluster Analysis)의 동의어이다(Han, J. et al. 2011).

비정형 데이터를 통해 가치를 창출하는 작업도 데이터마이닝에 해당하며, 텍스트에서 의미를 추출하는 방법을 별도로 텍스트 마이닝(TextMining)이라고 부르기도 한다. 텍스트 마이닝은 카테고리 분석, 감성 분석, 의도 분석 등 목적에 따라 다양한 방법으로 분류할 수 있다(송민 2017). 본 연구에서 진행하는 텍스트 분류(Text Classification)란, 문서의 내용을 기준으로 하나 혹은 그 이상의 범주(Category)에 할당하는 작업을 의미한다. 이는 데이터마이닝 관점에서 지도학습에 해당한다.

<그림 1>은 텍스트마이닝의 종류를 정리한 그림이다.

<그림 1> 텍스트마이닝의 종류

III. 연구 방법

1. 텍스트 처리 방법(Text Preprocessing Method)

(1) 형태소(morpheme) 분석

일반적인 수치 데이터와는 달리 텍스트 데이터는 아무런 처리도 하지 않은 상태로는 계산 할 수 없다는 특징이 있다. 따라서 텍스트 데이터를 수치 데이터로 변환해야 하는데, 수치 데이터로 변환하기 위한 가장 첫 번째 단계는 형태소를 분석하는 것이다. 언어학에서의 형태소란 의미가 있는 최소의 단위로 정의할 수 있으며 문법적·관계적인 뜻을 나타내는 단어 또는 단어의 부분을 일컫는다(국립국어원, 한국어기초사전). 또한 형태소 분석이란 형태소보다 큰 언어 단위인 어절, 혹은 문장을 최소 의미 단위인 형태소로 분리하는 과정을 의미하며, 형태소를 분석하는 것이 아니라 형태소로 분석하는 것이 핵심이다(강승식 2003). 본 연구에서는 형태소 분석을 통해 명사를 추출하였다.

(2) 불용어(Stopword) 처리

불용어(Stopword)란 인터넷 검색 시 검색 용어로 사용하지 않는 단어 또는 관사, 전치사, 조사, 접속사 등 검색 색인 단어로 의미가 없는 단어이다(국립국어원, 한국어기초사전). 실제로 텍스트 분석의 효과를 높이기 위해서는 가진 데이터에서 유의미한 단어만을 선별하여야 하는데 이 과정에서 불용어를 제거하는 것도 좋은 방법이 될 수 있다(최다빈 등 2020).

본 연구에서는 동사형의 '한다', '이다' 같이 형태소 분석에서 제대로 제외되지 않은 단어, 관사, 전치사, 조사 등의 의미 없는 단어, 자료의 특성상 빈발하게 사용되는 단어(ex: 질문, 회사, 사람인 등), 오타를 불용어라고 정의하였으며, TF-IDF 알고리즘을 활용하여 제거하였다.

TF-IDF(Term Frequency - Inverse Document Frequency)는 텍스트 마이닝에서 이용하는 가중치 모델로, 여러 개의 문서가 존재할 때 특정 단어가 해당 문서 내에서 얼마나 중요한 것인지를 나타내는 통계적 수치이다. 문서의 핵심어를 추출하거나, 검색 엔진에서 검색 결과의 순위를 결정하는 등의 용도로 사용할 수 있다(Qaiser, S. and Ali, R. 2018).

<표 3>과 같이 TF(term frequency)는 특정한 단어가 문서 내에 얼마나 자주 등장하는지 나타내는 값이고, DF(document frequency)는 특정한 단어가 문서에 포함했는지 나타내는 값이며, IDF는 DF의 역수이다(Han, J. et al. 2011). 즉, TF-IDF는 TF와 IDF를 곱하여 구할 수 있다.

<표 3> TF-IDF 가중치 모델

TF(term frequency)	$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$ $n_{i,j}$:단어 ti가 문서 dj에서 출현한 횟수 $\sum_k n_{k,j}$: 문서 dj에서 모든 단어가 출현한 횟수
DF(document frequency)	$idf_i = \log\frac{\lvert D\rvert}{\lvert d_j \rvert t_{j\in} d_j\rvert}$ $\lvert D\rvert$: 문서집합에 포함되어 있는 문서의 수 $\lvert d_j \rvert t_j \in d_j\rvert$: 단어 t_j가 등장하는 문서의 수
TF-IDF	$TFIDF_{i,j} = tf_{i,j} \times idf_i$

TF-IDF는 하나의 문서 내에서 특정 단어 빈도가 높을수록, 그리고 전체 문서에서 특정 단어를 사용한 문서의 수가 낮을수록 TF-IDF 값이 높아진다. 따라서 TF-IDF를 활용하면 흔하게 나타나는 단어를 선별하는 효과를 얻을 수 있다(Qaiser, S. and Ali, R. 2018).

(3) 텍스트 벡터화(Text Vectorization)

텍스트 또는 이미지와 같은 비정형 데이터를 별도의 절차 없이 분석에 활용하는 것은 매우 어려운 일이다. 따라서 텍스트 데이터를 가지고 데이터 분석을 하기 위해서는 컴퓨터가 텍스트를 이해할 수 있도록 수치화해주는 과정이 필요하다. 이때, 텍스트 데이터를 수치데이터로 변환하는 작업을 텍스트 벡터화(Text Vectorization)라고 부른다.

다만, 텍스트를 곧바로 벡터화할 수 없으며, 벡터화하기 위해서는 텍스트를 최소 단위로 분리하는 토큰화(Tokenization) 작업이 필요하다. 토큰화는 데이터를 토큰(Token)이라 불리는 단위로 변환하는 작업을 일컬으며, 여기서 토큰(Token)이란 문법적으로 더 이상 분리할 수 없는 단위를 의미한다. 일반적으로 텍스트 전처리를 진행할 때에는 토큰의 단위를 단어(Word)로 설정하여 텍스트를 토큰화한다(유원준 2020).

<표 4> 텍스트 벡터화(Text Vectorization) 방법

방법	설명
One-Hot encoding	• N개의 단어를 각각 N차원의 벡터로 표현하는 방법 • 단어에 해당되는 차원에 1을 넣고 나머지는 0인 희소벡터 • 단어의 특정한 관계나 의미를 포함하지 않음
TF-IDF	• 특정 단어가 문서 내에 등장하는 빈도와 그 단어가 문서 전체 집합에서 등장하는 빈도를 고려하여 벡터화 하는 방법 • 단어의 빈도에 역문서 빈도를 가중치로 주어 계산하는 개념
Word Embedding	• 단어를 밀집 벡터(dense vector)의 형태로 표현하는 방법 • 단어들은 N차원의 벡터로 표현되며 실수 값을 갖음 • 대표적으로 단어 간 유사도를 계산하는 Word2Vec가 있음

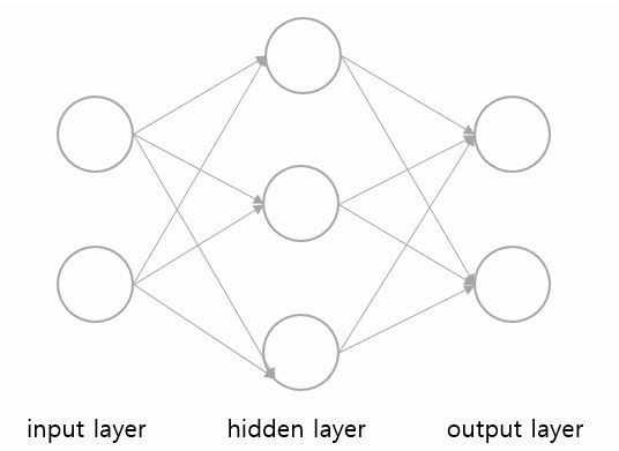
<표 4>는 대표적인 텍스트 벡터화 방법 3가지를 보여주고 있다. 본 연구에서는 원-핫 인코딩(One-Hot Encoding)을 활용하여 벡터화하였다.

2. 딥러닝(Deep Learning)

(1) 인공신경망(artificial neural network)

신경망(Neural Network)이란 인간의 뇌가 가지는 생물학적 특성 중 뉴런의 연결 구조를 가리키며, 이러한 신경망을 수학적으로 추상화해 일정 단위의 인위적인 네트워크로 표현한 것을 인공신경망(Artificial Neural Network)이라고 부른다. 즉, 뇌를 형성하는 뉴런의 집합체를 수학적으로 표현한 알고리즘이다(Bengio, Y. et al. 2013).

<그림 2> 인공신경망의 예



인공신경망은 <그림 2>와 같이 기본적으로 입력층(Input Layer), 은닉층(Hidden Layer), 출력층(Output Layer) 3개의 층으로 구분할 수 있다. 입력층은 예측값을 도출하기 위한 데이터를 입력하는 층이며, 출력층은 인공신경망 프로세스를 통해 나온 결과값을 반환하는 층이다. 마지막으로 은닉층은 입력층과 출력층을 잇는 역할을 하며 모든 입력 노드로부터 입력값을 받아 가중합을 계산하고, 이 값을 전이함수에 적용하여 출력층으로 전달하는 층으로, 인공신경망의 핵심이라고 할 수 있다(Saito Goki(이복연 옮김) 2017).

입력층과 출력층 사이에 여러 개의 은닉층으로 이루어진 인공신경망을 심층신경망(Deep Neural Network, DNN)이라 부른다. 심층신경망은 일반적인 인공신경망과 마찬가지로 복잡한 비선형 관계(Non-linear relationship)들을 모델링할 수 있으며, 오차의 역전파(Backpropagation) 알고리즘을 통해 학습한다. 이때, 가중치들은 아래의 <식 1>을 이용한 확률적 경사 하강법(Stochastic gradient descent)을 통하여 조정할 수 있다(Wakui Yoshiyuki and Wakui Sadami(박광수 옮김) 2018).

<식 1> $$\Delta w_{ij}(t+1) = \Delta w_{ij}(t) + \eta \frac{\partial C}{\partial w_{ij}}$$

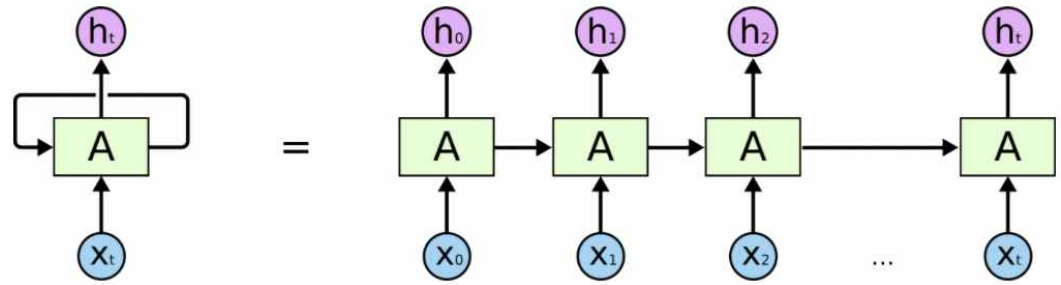
<식 1>에서 C는 비용함수(Cost function)을 의미하며 η는 학습률(Learning rate)을 의미한다. 일반적으로 비용함수는 학습하고자 하는 데이터의 형태와 활성화함수(Activation function)에 의해서 결정된다. 본 연구에서와같이 다중 카테고리 분류 문제(Multiclass classification problem)에서 지도 학습을 수행할 때에는 일반적으로 활성화함수로 Softmax함수를 사용하며, 비용함수는 교차 엔트로피 함수(Cross entropy function)로 결정된다(Hinton, G. et al 2012).

딥러닝은 인공신경망의 구조를 어떻게 설정하느냐에 따라 합성곱신경망(Convolutional Neural Network, CNN), 순환신경망(Recurrent Neural Network, RNN) 등 다양한 알고리즘으로 분류할 수 있으며, 분석 목적에 따라 적합한 알고리즘을 선정하여야 한다. 본 연구에서는 텍스트 분류 문제를 해결하기 위하여 연속성(Sequence)을 학습에 반영할 수 있는 순환신경망을 활용하여 연구를 진행하고자 한다.

(2) 순환신경망(RNN, Recurrent Neural Network)

일반적인 인공신경망의 은닉층에는 순서가 반영되지 않고 단순히 뉴런만 배치된 구조지만, 순환신경망(RNN)은 과거의 사건이 미래의 결과에 영향을 줄 수 있는 순환 구조(Directed cycle)로 구성되어있다. 따라서 데이터 중에서도 연속성(Sequence)을 갖는 시계열 데이터, 문자, 음성 등 다양한 분야에서 RNN이 활용되고 있다(Yusuke Sugomori(김범준 옮김) 2017). 본 연구에서도 텍스트 즉, 문자를 처리하기 위한 방법으로 RNN을 활용한다.

<그림 3> RNN의 구조



*그림출처 : Olah, C.(2015). Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

RNN에서 Recurrent는 이전의 시점에서 어떤 정보가 추가로 오는 것을 의미하며, 간단하게 설명하자면 <그림 3>과 같이 직전 데이터(t-1)와 현재 데이터(t) 간의 상관관계를 고려하여 다음의 데이터(t+1)를 예측하고자 하는 알고리즘이라고 볼 수 있다. 즉, 현재 데이터뿐만 아니라 과거 데이터까지 고려하여 결과를 예측한다는 특징을 갖는다.

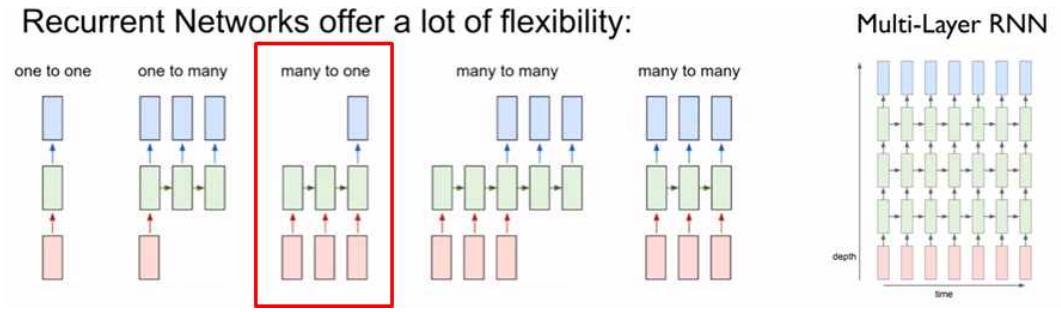
<그림 3>을 자세히 살펴보면, 매번 들어오는 입력을 X로 표현하고 있으며, 바깥으로 나가는 출력을 h로 표현하고 있다. RNN에서의 핵심은 바로 상태(State)를 갖는다는 것인데 h를 가리킨다. 다시 말해 h는 RNN에서의 출력으로 이동하는 값인 동시에 다음 시점으로 전달하는 상태 값이다.

<식 2>

$$h_t = fw(h_{t-1} + x_t)$$
$$y_t = W_{hy}h_t$$

<식 2>는 RNN을 간단하게 수식화한 것이다. <식 2>를 보면 t에 대해 두 가지 계산을 하는 것을 확인할 수 있다. h_t는 현재 상태를 의미하고 이전 상태(h_{t-1})와 입력값(x_t)을 사용하여 계산한다. 그리고 y_t는 예측값을 의미하고 가중치(W)와 현재 상태(h_t)를 곱하여 계산한다(Zaremba, W. et al. 2014).

<그림 4> RNN의 설계구조



*그림출처 : Andrej Karpathy(2015), The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/

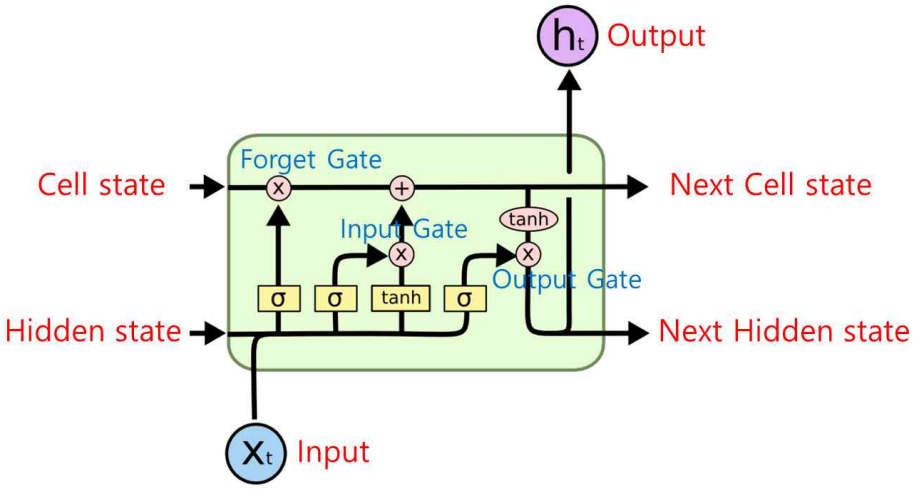
<그림 4>는 설계에 따라 달라지는 RNN의 구조를 시각화한 그림이다. RNN은 크게 Input과 Output에 의해서 구조가 바뀌게 되는데, 크게 1대다(one-to-many), 다대1(many-to-one), 다대다(many-to-many), 다대다(many-to-many) 형태로 분류할 수 있다. 본 연구에서 진행하는 텍스트 분류 모델은 하나의 문서에 사용된 단어들이 Input으로 들어간 후, 최종 카테고리 1개를 출력하는 다대1(many-to-one) 구조이다. 또한 둘 이상의 RNN을 층을 쌓아 복잡한 모델을 만들 수도 있는데, 이러한 모델을 <그림 4>에서 맨 오른

쪽에 있는 Multi-Layer RNN이라고 부른다. 다만, Multi-Layer RNN은 과적합에 대한 우려가 있어 본 연구에서는 single-Layer RNN으로 설계하였다.

(3) LSTM(Long Short-Term Memory)

RNN은 시간을 많이 거슬러 올라갈수록(Long term) 경사를 소실하는 문제가 발생한다. 이를 장기의존성 문제(the problems of long-term dependencies)라고 부르며, 초기값에 따라서 과거 데이터를 계속 곱할수록 작아지는 현상 때문에 생기는 문제이다. 이러한 문제점을 해결하기 위해 고안된 알고리즘이 LSTM(Long Short-Term Memory)이다. LSTM은 직전의 데이터뿐만 아니라 거시적으로 과거의 데이터까지 고려하여 미래를 예측할 수 있게끔 설계되었다(Hochreiter, S., and Schmidhuber, J. 1997).

<그림 5> LSTM 네트워크 구조

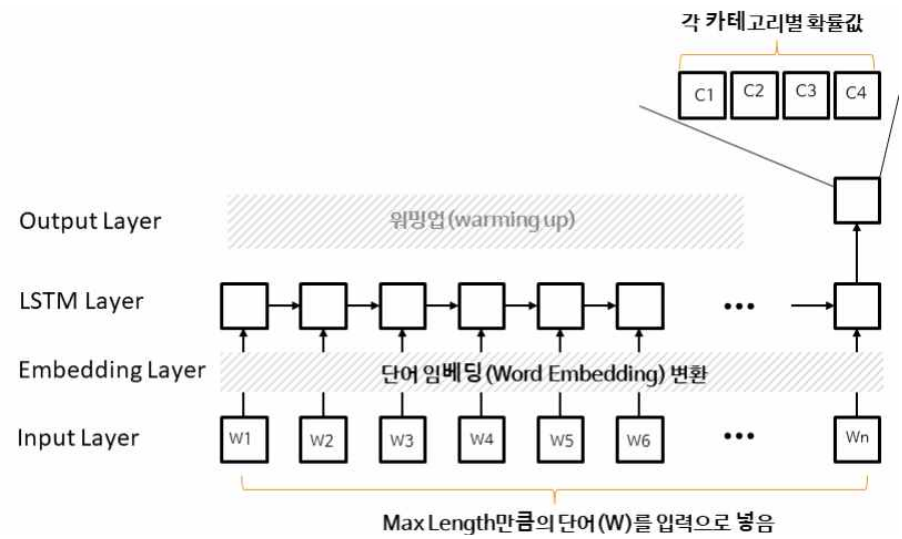


*그림출처 : Andrej Karpathy(2015), The Unreasonable Effectiveness of Recurrent Neural Networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

<그림 5>는 LSTM의 네트워크 구조를 시각화한 그림이다. LSTM의 네트워

크 구조는 크게 6개의 파라미터와 3개의 게이트로 구성된다. Input은 X_t로 표시하며 입력값을 의미한다. Output은 h_t로 표시하며 출력값을 나타내며 다음 hidden state 값과 항상 동일하다. Cell state는 오차가 사라지지 않고 전체 LSTM을 관통하도록 설계한 값이다. 이 Cell state가 LSTM의 핵심이라고 볼 수 있으며 3개의 gate에 의해 제어된다. Input Gate, Forget Gate, Output Gate는 정보들이 어느 시점에서 정보를 버리거나 유지하여 선택적으로 흘러갈 수 있게 조절하는 역할을 한다(Hochreiter, S., and Schmidhuber, J., 1997).

<그림 6> LSTM 프로세스



<그림 6>은 텍스트 분류를 위한 LSTM의 프로세스를 시각화한 그림이다. 층은 Input Layer, Output Layer, Embedding Layer, LSTM Layer 등 4개로 이루어져 있으며 각 Layer와 파라미터에 대한 자세한 설명은 4장 분석 결과에서 다루고자 한다.

Ⅳ. 데이터 설명 및 전처리

1. 데이터 설명

(1) 데이터 수집

<표 5> 수집한 데이터에 대한 개요

분류	특징
데이터출처	사람인(saramin)의 취준진담 커뮤니티
수집기간	2019년 11월 6일 ~ 2020년 9월 14일(약 10개월)
크기	5,894개
카테고리	'경영진', '근무환경', '급여', '면접/자소서', '이직/퇴사', '자기개발', '조직문화', '직무', '기타' 등 9개

본 연구에서는 취업포털사이트 '사람인(saramin)의 취준진담 커뮤니티'에서 2019년 11월 6일 ~ 2020년 9월 14일(약 10개월) 사이에 올라온 취업질문 게시글을 수집하였다. 수집한 데이터는 총 5,894개이며 해당 커뮤니티는 '경영진', '근무환경', '급여', '면접/자소서', '이직/퇴사', '자기개발', '조직문화', '직무', '기타' 등 9개의 카테고리로 게시글을 구분하고 있다. 다만 카테고리 설정은 글쓴이가 직접 카테고리를 설정하여야 하며 또한, 단 하나의 카테고리만 설정할 수 있기 때문에 실제 글의 내용과 설정된 카테고리가 일치하지 않은 경우가 많다는 특징이 있다.

그럼에도 불구하고 해당 데이터를 '문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델'의 성능을 확인하기 위한 데이터로 선정한 이유는 다음과 같다. 첫째, 온라인 커뮤니티 또는 SNS 등 User에 의해 텍스트 데이터가 생성되어 깔끔하지 않은 상황에서도 사용할 수 있는 모델을 연구하고자 하였다. 둘째, 카테고리 간 구분이 애매한 상황에서의 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델이 어느 정도까지 성능을 발휘하는지 확인하고자 하였다.

(2) 데이터 기초분석

본 연구자는 '문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델' 설계를 진행하기에 앞서 '사람인(saramin)의 취준진담 커뮤니티'의 특성을 살펴보기 위하여 기초분석을 진행하였다.

<표 6> 월 단위 게시글 수

년도	월	개수	비율
2019	11	260	4%
	12	299	5%
2020	1	1,034	**18%**
	2	1,023	**17%**
	3	790	**13%**
	4	632	11%
	5	561	10%
	6	362	6%
	7	403	7%
	8	357	6%
	9	172	3%

<그림 7> 월 단위 게시글 수

우선 2019년 11월에서 2020년 10월까지 시계열에 따라 게시글의 수를 확인해본 결과, <표 6>, <그림 7>과 같이 1월, 2월, 3월에 각각 1,034개(18%), 1,023개(17%), 790개(13%) 등 가장 활발하게 게시글이 올라왔으며, 연말로 갈수록 게시글이 줄어드는 특징을 확인할 수 있었다. 이는 취업포털사이트의 특성상 채용이 활발한 상반기에 글이 많이 올라오고 있다고 추측할 수 있다.

'사람인(saramin)의 취준진담 커뮤니티'에 올라온 게시글의 공백을 제외한 길이는 평균 110.1, 중앙값 71로 나타났으며 이를 통해 게시글 길이의 분포가 부적편포(Negatively skewed)를 띄고 있음을 확인할 수 있다. 또한 표준편차가 134.9로 매우 높게 나왔는데 게시글마다 사용된 텍스트의 길이가 많이 다르다는 것을 의미한다.

마지막으로, 클래스의 불균형 문제(Class imbalance)가 존재하는지 확인하기 위해 카테고리별 데이터를 개수를 파악하였다.

<표 7> 카테고리별 게시글 수

Category	Count
직무	**1,697**
급여	1,200
근무환경	967
면접/자소서	742
기타	**450**
이직/퇴사	381
조직문화	368
자기개발	111
경영진	**26**
합계	5,892

카테고리별 데이터의 개수를 파악한 결과, '직무', '급여', '근무환경', '면접/자소서', '기타', '이직/퇴사', '조직문화', '자기개발', '경영진' 순으로 게시글의 수가 많았으며 <표 7>과 같이 카테고리별 데이터의 개수가 불균형한 것을 확인할 수 있었다. 게시글이 가장 많은 카테고리('직무')와 적은 카테고리('경영진') 간의 차이는 1,671개로 상당히 높았으며 실제로 데이터 분석을 진행하기 전에 카테고리를 재설정할 필요가 있었다.

또한, 카테고리 중 '기타'는 450개로 꽤 많은 게시글이 해당하였으나 원본 데이터를 탐색해본 결과 다양한 주제의 글이 포함되어 있었기에 분류기의 성능을 저하시킬 수 있다고 판단하여 실제 모델링에서는 제외한 후, 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델을 설계하였다.

2. 데이터 전처리

(1) 카테고리 재분류(Category Reclassification)

본 연구에서 활용하는 '사람인(saramin)의 취준진담 커뮤니티' 데이터는 9개의 카테고리로 분류를 하고 있으나, 카테고리별 내용의 구분이 명확하지 않고 클래스 불균형 문제가 존재하기 때문에 카테고리를 재분류 하였다.

<표 8> 카테고리별 자주 등장한 단어

Category	Keyword	Count
경영진	기업문화, 상사, 분위기	26
조직문화	호칭, 연차, 휴가, 워라밸, 분위기, 회식, 야근, 복지	368
근무환경	휴무, 업무복장, 업무강도, 정규직전환	967
면접/자소서	스펙, 취업준비, 기업, 지원, 면접	742
자기개발	스펙, 성적, 학업, 취업준비, 자소서	111
이직/퇴사	이력서, 이직, 퇴직, 스펙, 성적, 취업준비	381
급여	연봉, 성과급, 직급, (구체적인수치)	1,200
직무	우대학과, 자격증, 직무, 역할, 기술, 경험, 업무, 연구	1,697

<표 8>은 실제 데이터셋에서 카테고리별 자주 등장한 단어를 확인한 결과이다. <표 8>의 Keyword를 보면 '경영진', '조직문화', '근무환경'은 기업의 문화, 환경, 분위기, 복지 등 기업의 환경과 문화와 관련된 단어가 많이 등장한 것을 확인할 수 있었다. 또한, '근무환경', '면접/자소서', '자기개발'은 스펙, 취업준비, 면접 등 취업 및 취업준비와 관련된 단어가 많이 등장한 것을 확인할 수 있었다. 이 외에 '급여' 카테고리는 명확하게 다른 카테고리와 구분되는 것

을 확인할 수 있었으며, 마지막으로 '직무' 카테고리는 '면접/자소서', '자기개발'과 유사한 속성을 보였으나 '직무' 자체의 게시글 수(Count)가 높아 재분류에서 제외하였다.

<표 9> 카테고리 재분류 결과

Categories	New Category	Count
경영진, 조직문화, 근무환경	환경문화	1,361
면접/자소서, 자기개발, 이직/퇴사	취업준비	1,243
급여	급여	1,200
직무	직무	1,697

본 연구에서는 기존의 '경영진', '조직문화', '근무환경'을 '환경문화'로 통합하였으며, '면접/자소서', '자기개발', '이직/퇴사'를 '취업준비'로 통합하였다. '급여'와 '직무'는 카테고리 재분류에서 제외하였으며 '기타'는 하나의 카테고리로 합치기에 어려움이 있어 사용데이터에서 제외하였다.

최종적으로 선정한 카테고리는 <표 9>와 같다. 카테고리를 재분류한 결과, '환경문화(1,361개)', '취업준비(1,243개)', '급여(1,200개)', '직무(1,697개)' 등 클래스 불균형 문제가 해결된 것을 확인할 수 있다.

(2) 문장 분리(Sentence Separation)

본 연구에서 제안하는 RNN-LSTM 모델은 분석 프로세스를 2단계로 구분한다. 우선 하나의 게시글을 문장 단위로 분리하고 나서 문장별 카테고리를 예측한다. 그다음, 문장 예측 결과값을 최종 게시글 예측에 필요한 가중치로 반영하여 분류 성능을 높이고자 한다. 따라서, 수집한 게시글 데이터를 문장 단위로 분리하는 과정이 필요하다.

<표 10> 문장 구분을 위한 구분자 선정

구분자(Separator)
마침표(.), 물음표(?), 느낌표(!), 물결표(~), 울음표시(ㅠ, ㅜ), 웃음표시(ㅋ, ㅎ) 등

문장 단위로 분리하기 위해 '사람인(saramin)의 취준진담 커뮤니티'에 작성된 텍스트 특성을 파악하였다. 해당 커뮤니티는 취업을 희망하는 사람들이 주로 글을 작성하였으며, 대부분 취업과 관련된 정보를 요청하는 내용이었다. 또한, 정보를 요청한다는 특성상 띄어쓰기와 맞춤법 상태가 양호하였다. 그리고 문장을 구분하기 위해서 마침표(.), 물음표(?), 느낌표(!), 물결표(~), 울음표시(ㅠ, ㅜ), 웃음표시(ㅋ, ㅎ)를 주로 사용하는 것을 확인할 수 있었다. 따라서 <표 10>에 나열한 8개 기호 및 자음을 구분자(Separator)로 선정하였다. 다만, 구분자가 두 번 이상 동시에 사용하는 경우에 문장을 잘못 분리할 수 있어서 2개 이상의 구분자를 1개로 변환하는 작업도 같이 진행하였다.

<표 11> 종결어미의 종류 및 형태

종결어미 종류	형태	예시
평서형 어미	-다, -니다, -요, -죠 등	그가 간다
감탄형 어미	-는구나, -는구려 등	그가 가는구나
의문형 어미	-느냐, -는가, -니, -요 등	그가 가느냐
명령형 어미	-어라, -거라, -너라 등	빨리 먹어라
청유형 어미	-자, -세 등	우리 같이 가자

이 외에도 본 연구에서는 구분자 외에 문장을 구분하기 위하여 종결어미를 활용하였다. 종결어미(sentence-closing ending)란 문장을 끝맺게 하는 형태소로, <표 11>과 같이 문장의 종결 형식에 따라 평서형, 의문형, 명령형, 청유형, 감탄형 종결 어미로 구분된다(국립국어원, 한국어기초사전). '사람인(saramin)의 취준진담 커뮤니티'에서는 대부분 평서형 어미와 의문형 어미를 사용하고 있는 것을 확인할 수 있었으며, 게시글을 문장 단위로 분리하는 데에 이용하고자 하였다.

실제로 수집한 데이터에서 문장을 분리하기 위해 해당 커뮤니티에서 자주 사용되는 '니다', '어요', '아요', '데요', '해요', '네요', '겠죠', '까요', '나요', '려요', '서요' 등 11개의 종결어미를 선정하였으며, 문장 단위로 구분해주기 위해 종결어미의 뒤에 마침표(.)를 추가해주었다. 앞서 설명한 종결어미와 구분자를 통해 5,492개의 게시글을 문장 단위로 분리하여 총 18,779개의 문장을 생성하였다.

(3) 텍스트 전처리(text preprocessing)

다음 과정으로 형태소를 분석하여 명사를 추출하고 텍스트 분석을 진행하기에 앞서 텍스트 전처리를 진행해줄 필요가 있다. 일반적으로 텍스트를 전처리하기 위해서는 분석하고자 하는 데이터의 특성과 연구의 목적에 부합된 작업을 선정해야 한다. 본 연구에서는 한국어로 작성된 텍스트의 주제를 분류하는 것이 목적이기 때문에 타 언어로 작성된 문자를 제거하였으며, 의미상 필요하지 않은 자음과 모음으로만 구성된 문자도 제거해주었다. 또한, 명사를 추출하는 과정에서 등장하는 길이가 1인 글자는 대부분 2개 이상의 뜻이 있기 때문에 다양한 해석이 가능하여 텍스트 분석에 방해가 되므로 제거해주었다.

마지막으로 형태소 분석 시에 발생하는 오류를 해결하기 위해 특수문자를 제거해주었다. 정규표현식을 활용한 특수문자 제거방식은 제거하고자 하는 모든 특수문자를 연구자가 직접 리스트를 만들어야 하는 어려움이 있다. 이에 본 연구에서는 정규표현식을 사용하는 대신, Python의 isalnum()함수를 사용하여 한 글자씩 불러와 해당 문자가 영어, 한글 혹은 숫자인지 확인하는 방식으로 처리해주었다.

(4) 형태소 분석(Morpheme Analysis)

Python에서는 한국어의 형태소 분석을 돕기 위한 패키지로 KoNLPy를 제공하고 있다. 텍스트를 형태소로 분리하는 방법으로는 ① 단어-품사 사전을 구축하여 이를 이용해 단어를 품사로 분리하는 방법과, ② 사전을 구축하지 않고 모델을 학습 시켜 구별해내는 방법이 있는데 KoNLPy에서는 단어-품사 사전을 구축하여 형태소를 분리하는 방법을 제공하고 있으며, <표 12>와 같다.

<표 12> KoNLPy의 형태소 분석 방법

방법	특징
Hannanum	• 복합명사를 추출하는 데에 효과적 • 띄어쓰기가 제대로 쓰이지 않은 텍스트에 약함
Kkma	• 최소명사 단위로의 분리에 좋음 • 복합명사를 추출하는 데에 약함
Komoran	• 띄어쓰기가 제대로 쓰이지 않은 텍스트에도 효과적 • 다른 방법에 비해 명사추출 성능이 낮음
Okt	• 오타를 제대로 인식하지 못함

본 연구에서는 텍스트 분류가 목적이기 때문에 복합명사를 정확하게 인식하기보다는 사용된 명사를 제대로 추출할 방법을 선택할 필요가 있었다. 즉, 복합명사를 두 개의 명사로 분리된 상태로 형태소를 분리하여도 문제가 없기에 KoNLPy에서 제공하는 방법 중에서 Kkma를 선정하였다. <표 13>은 2개의 성능을 비교한 표이다.

<표 13> Hannanum과 Kkma의 성능 비교

방법	형태소 분석 결과
실제 데이터	"웹서비스기획이 실제로하는 직무랑합격스펙 알려주세요."
Hannanum	“웹서비스기획”, “직무랑합격스펙”
Kkma	“웹”, “서비스”, “기획”, “직무”, “합격”, “스펙”

(5) 불용어 처리(Stopword Removal)

Python의 KoNLPy를 활용하면 간편하게 형태소 분석을 진행할 수 있으나, 완벽하게 형태소를 분리해주는 것은 아니다. 실제로 형태소 분석 결과를 보면 동사가 제대로 지워지지 않아 '입니', '있다', '하기' 등의 단어가 많이 출현된 것을 확인할 수 있었으며, 이 외에도 명사가 아닌 다른 형태소도 많이 포함된 것을 확인할 수 있었다. 또한, '질문', '문의', '기업' 등의 단어는 '사람인의 취준 진담' 커뮤니티의 특성으로 인해 대부분의 게시글에서 사용되는데 이렇게 자주 사용되는 단어도 제거해야 할 필요가 있다.

본 연구자는 TF-IDF 알고리즘을 활용하여 위와 같은 불용어, 오타, 의미없는 단어를 제거하고자 하였다. TF-IDF 알고리즘을 활용하여 불용어를 없애기 위해서는 ① TF-IDF값을 계산하고, ② 단어 수 × 2의 Matrix로 변환, 마지막으로 ③ 의미없게 쓰인 비율(No-Meaning Rate)을 계산해주어야 한다.

① TF-IDF 가중치 계산

우선 명사를 추출한 텍스트 데이터를 가지고 TF-IDF 가중치 값을 계산하기 위해 Python의 sklearn패키지를 이용하였다. sklearn은 TF-IDF를 계산해주는 함수인 TfidfVectorizer()를 제공한다.

<표 14>는 실제 데이터에 TfidfVectorizer()를 적용한 결과이다.

<표 14> TF-IDF 계산 결과

문장번호	가격	가계	가공	…	흥미	희망직무	희망퇴직
1	0	0	0	…	0	0	0
2	0	0	0.097	…	0	0	0
3	0	0	0	…	0.2956		
4	0	0	0	…	0	0.1590	0
…							
18,778	0	0	0	…	0.1061	0	0.1061
18,779	0.1590	0	0	…	0	0	0

실제로 사용한 데이터는 18,779개의 문장과 13,576개의 단어로, 18,779× 13,576의 크기의 Matrix가 생성되었다.

② 단어 수 × 2의 Matrix 변환

Matirx에 표시된 가중치 값은 해당 단어가 그 문서에서 얼마나 중요한지 수치로 표시한 값으로, 0에 가까울수록 의미 없음을, 1에 가까울수록 중요한 단어임을 나타낸다. 또한, 같은 단어일지라도 게시글에 따라 계산된 값이 다르게 나온다는 특징이 있다. 따라서 본 연구에서는 가중치 값에 'H=0.1'라는 기준을 세워 H를 넘으면 '해당 문서에서 의미 있게 사용됨', H를 넘지 못하면 '해당 문서에서 의미가 없음'으로 판단하여, H보다 큰 값을 0으로, 작은 값을 1로 대체하였다. 그리고 <그림 8>처럼 단어별 빈도수를 더한 결과인 13,576(단어의 개수) × 2(단어, 누적 빈도)의 크기인 Matrix로 변환하였다. 여기에서 누적 빈도는 18,779개의 게시글에서 특정 단어가 의미 없게 사용된 횟수를 계산한 값이다.

<그림 8> TF-IDF 결과를 변환하는 과정

	단어1	단어2	단어3	.....	단어n
문서1					
문서2					
문서3			행렬의 값은 모두 0 또는 1		
⋮					
⋮					
문서m					
결과	$\sum_{k=1}^{m}$ 문서m	$\sum_{k=1}^{m}$ 문서m	$\sum_{k=1}^{m}$ 문서m	$\sum_{k=1}^{m}$ 문서m	$\sum_{k=1}^{m}$ 문서m

→

명사	누적빈도
단어1	$\sum_{k=1}^{m}$ 문서m
단어2	$\sum_{k=1}^{m}$ 문서m
⋮	
⋮	
단어n	$\sum_{k=1}^{m}$ 문서m

③ 의미 없이 쓰인 비율(No-Meaning Rate) 계산

다만, 의미 없게 사용된 횟수를 가지고 불용어 사전을 구축하기에는 문제가 존재한다. 그 이유는 실제빈도가 높으면 의미 없이 사용된 누적빈도 자체도 높아지기 때문이다. 즉, 자주 쓰이지 않는 불용어 또는 오타의 경우 실제 빈도수 자체가 낮기 때문에 불용어를 제대로 제외할 수 없다. 따라서, <그림 9>처럼 의미 없이 사용된 누적빈도를 실제빈도로 나눠주어 No-Meaning Rate를 구해주었다.

<그림 9> No-Meaning Rate 계산 과정

명사	의미없게 사용된 빈도의 합 (A)	실제빈도 (B)	No-Meaning Rate
단어1	$\sum_{k=1}^{m}$ 문서m	freq_단어1	A/B
단어2	$\sum_{k=1}^{m}$ 문서m	freq_단어2	A/B
⋮			
⋮			
단어n	$\sum_{k=1}^{m}$ 문서m	freq_단어n	A/B

No-Meaning Rate는 TF-IDF 가중치 값을 구한 나온 의미 없게 사용된 누적빈도를 실제 누적빈도로 나눈 비율로, '특정 단어 별 의미 없게 사용된 비율'을 나타낸다. 즉, No-Meaning Rate는 불용어, 오타, 의미 없는 단어들에 대한 목록을 제시해주며, 최종적으로 연구자는 위의 목록에 해당하는 단어를 제외한 단어들의 실제빈도를 사용하여 데이터를 분석할 수 있다. 이러한 방식으로 본 연구에서는 총 7,257개의 불용어를 선정하고, 불용어 사전(Stopword List)을 구축하였다.

(6) 토큰화(Tokenization)

텍스트 데이터를 분석하기 위해서는 수치로 변환해주는 작업이 필요하다. 텍스트를 수치로 변환하는 방법을 토큰화(Tokenization)라고 부른다. 토큰화 과정에서 일반적으로 특수문자 제거여부, 줄임말과 띄어쓰기 구분여부, 토큰의 기준 등 고려해야 할 사항들이 많으나, 본 연구에서는 이미 텍스트 데이터에 대한 전처리가 진행되었기 때문에 큰 문제 없이 바로 토큰화 작업을 진행하였다.

단어를 토큰의 기준으로 잡았으며, 이미 형태소처리가 되어 있기 때문에 Python의 딥러닝 패키지인 Keras에서 제공하는 Tokenizer함수를 이용하여 쉽게 변환할 수 있었다. 다만, 사용된 모든 단어를 토큰화시키기에는 토큰의 수가 너무 많아지기 때문에, 단어 빈도분석을 진행하여 빈도가 높은 순으로 5,000개의 단어만 토큰화 작업을 진행하였다. <표 15>는 토큰화 결과 예시이다.

<표 15> 단어 토큰화 결과 sample

단어	토큰화 결과
삼성, 삼성전자, 전자, 인적성, 시험, 준비	285, 961, 187, 2797, 175, 17
제도, 현대, 현대백화점, 백화점, 퇴근시간, 시간, 컴퓨터, 자동, 시행	287, 226, 5435, 1620, 660, 23, 369, 2462, 1355
홈플러스, 4년, 공채, 신입, 연봉	1358, 112, 373, 16, 5
생산직, 근무, 환경, 업무, 강도, 지원, 공정, 배치	85, 2, 7, 4, 33, 10, 337, 1783

본 연구에서는 원-핫 인코딩(One-Hot-Encoding)을 이용하여 수치 데이터로 변환하였다. 원-핫 인코딩이란 모든 단어에 고유한 정수 인덱스를 부여하는 방법으로, 정수 인덱스 i를 단어의 총 개수 N인 이진 벡터로 변환하는 과정이다. 다만, 일반적인 원-핫 인코딩 구조를 LSTM의 Input으로 넣는 방식은 학습 시간이 길어지는 현상이 있기 때문에 이를 해결하기 위하여 각 단어에 명시적으로 인덱스를 할당하는 방식으로 수치화시켰다.

(7)학습셋(Training Set) 구축

본 연구에서 제안하는 '문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델'을 설계하기 위해서는 학습셋을 구축하여야 한다. 왜냐하면 실제로 수집한 데이터는 게시글 단위로 카테고리가 설정되어 있기 때문에 문장 단위로 분리한 상태에서는 카테고리, 즉 정답이 없는 상태이다. 일반적으로 텍스트 분류는 지도학습의 한 방법이므로 정답을 사전에 매칭시켜주어야 하며 RNN을 통한 텍스트 분류에서는 대게 하나의 카테고리 마다 1,000개의 학습데이터가 있으면 분류하는 데에 괜찮은 성능을 발휘한다.

<표 16> 문장 카테고리 선정 및 학습셋 구축

문장 카테고리	학습셋 개수
취업준비	1,000
환경문화	1,000
직무	1,000
급여	1,000
의미없음	1,000

*'의미없음'은 의미 없는 문장을 제거하기 위해 생성함

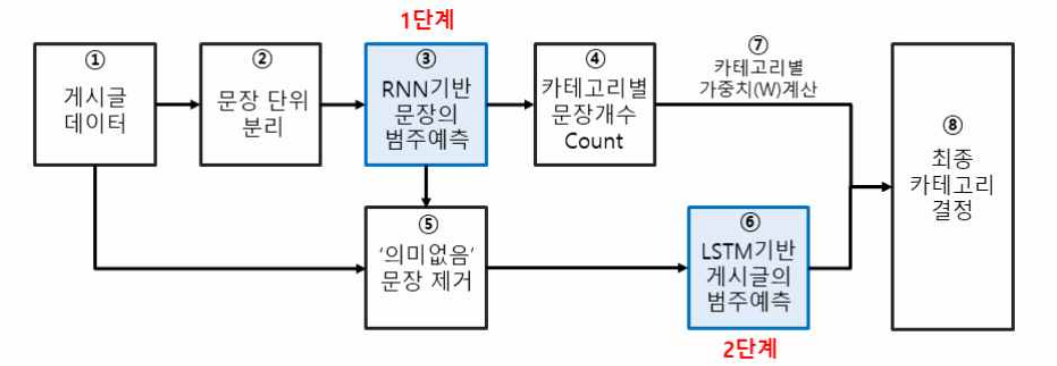
문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델에서의 핵심은 의미 없는 문장을 제거하여 학습의 속도를 높이고 모델의 성능을 높이는 데에 있다. 따라서 <표 16>과 같이 기존의 '취업준비', '환경문화', '직무', '급여' 등 4개의 카테고리와 의미 없는 문장을 제거하기 위한 '의미없음' 카테고리 등 5개의 카테고리를 문장의 정답으로 선정하였다. 그다음 연구자가 카테고리별 1,000개의 문장을 직접 분류하였으며 총 5,000개의 정답이 있는 문장 데이터를 구축하였다.

Ⅴ. 분석 결과

1. 전체 프로세스

문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델의 전체 프로세스는 <그림 10>과 같다.

<그림 10> RNN-LSTM 2단계 텍스트 분류 모델 전체 프로세스



제안하고자 하는 RNN-LSTM 모델은 우선 ① 게시글 데이터를 수집하여 ② 문장 단위로 분리한 후에 ③ RNN 기반의 문장 범주를 예측한다(1단계). 이때, 문장의 범주는 전처리 단계에서 만들었던 학습셋으로 모델을 학습한다.

그다음, ⑤ 예측된 문장의 카테고리가 '의미없음'에 해당되는 문장을 게시글 데이터에서 제거한다. 동시에 ④ 문장 예측 결과를 바탕으로 카테고리별 문장 개수를 Count한다.

마지막으로 의미 없는 문장이 제거된 게시글 데이터를 Input으로 넣고 ⑥ LSTM 기반의 게시글 범주를 예측한다(2단계). 모델의 마지막 Dense층(출력층)은 Softmax함수로 설정하여 카테고리별 확률값처럼 출력될 수 있게 하며, ⑦ 그 값에 ④에서 도출된 가중치 값을 반영하여 ⑧ 최종 카테고리를 결정한다.

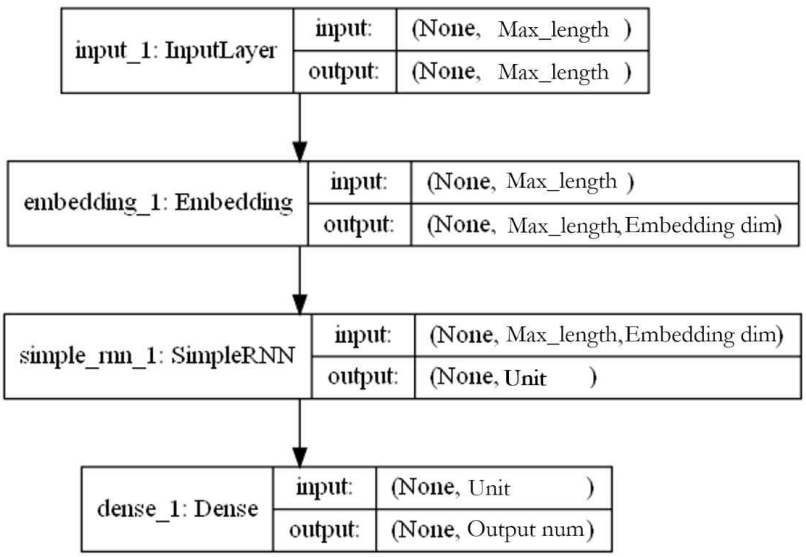
2. 1단계 : RNN 기반의 문장 범주 예측

(1) RNN 모델 설계

본 연구에서는 Python의 Keras 패키지를 활용하여 딥러닝 모델을 설계하였다. 문장 단위의 카테고리 예측 문제는 사용된 단어의 개수가 적기 때문에 장기의존성 문제의 발생 가능성이 낮다고 판단하여 RNN을 사용하였으며, 마찬가지로 RNN을 2개 이상 쌓기(stacked RNN)에는 과적합 문제가 발생할 것으로 판단하여 단순 RNN 층으로 구성하였다.

실제로 설계한 RNN 기반의 문장 범주 예측 모델은 <그림 11>과 같다.

<그림 11> RNN 기반의 문장 범주 예측 모델 도식화



<그림 11>와 같이 Input_layer(입력층), Embedding_layer(단어임베딩층), Simple_RNN_layer(RNN층), Dense_layer(완전연결층) 등 4개의 층으로 RNN 기반의 문장 범주 예측 모델을 구성하였다.

① Input_layer(입력층)

Input_layer은 1×최대입력길이(Max_length) 형식으로 입력하는 것을 볼 수 있으며, 여기서 최대입력길이란 RNN의 sequence를 의미한다. 즉, 문장의 카테고리를 예측하기 위하여 최대 몇 개의 단어까지 입력값으로 사용할지 선정해야 한다.

② Embedding_layer(단어임베딩층)

Embedding_layer는 RNN의 입력형식에 맞추기 위한 형식 변환 층이다. 일반적인 RNN의 입력 형식은 (batch_size, timesteps, input_dim)의 3차원으로 구성해야 하는데, RNN을 활용한 텍스트 분석에서는 토큰화된 텍스트 값을 Embedding Vector로 변환하여 형식을 맞춰준다.

③ Simple_RNN_layer(RNN층)

Simple_RNN_layer는 가장 핵심층으로, RNN이 실행되는 층이다. RNN의 Output으로 Unit값을 지정해야 하는데, 이는 은닉층의 크기를 의미한다. 즉, Unit은 하이퍼 파라미터이기 때문에 연구자가 값을 조정하며 최적값을 도출해야 한다. 또한, RNN에서 Unit은 출력층으로 보내는 값의 크기와도 동일하다.

④ Dense layer(완전연결층)

Dense layer은 출력층으로도 부르며, 본 연구에서는 텍스트 분류 모델이기 때문에, Output에 문장의 카테고리 개수인 5로 지정하였다.

(2) 하이퍼 파라미터의 변화에 따른 성능 비교

최적의 RNN 기반 문장 카테고리 예측 모델을 설계하기 위해서는 딥러닝의 하이퍼 파라미터를 조작하며 성능을 평가하여야 한다. 실제로 연구자가 조작할 수 있는 RNN의 하이퍼 파라미터는 Train-Test ratio(학습데이터 비율), Max length(최대 입력 길이), Unit number(유닛 수), Drop out ratio(드롭아웃 비율), Embedding dim(단어 임베딩 차원), Epochs(에포크), Activation function(활성화 함수), Loss function(손실 함수), Optimizer function(최적화 함수), Validation_split ratio(유효성 검사 비율) 등 다양하게 존재한다.

<표 17> 하이퍼 파라미터 설정

하이퍼 파라미터	설정 값
Output node	5(Category num)
Train-Test ratio	0.7
Validation_split ratio	0.2
Drop out ratio	0.2
Activation function	Softmax
Loss function	Categorical_crossentropy
Optimizer function	Adam
Epochs	300

본 연구에서는 <표 17>과 같이 8개의 하이퍼 파라미터값을 고정한 후, Max length, Embedding dim, Unit 3개의 하이퍼 파라미터를 조작하면서 성능을 비교하고자 한다.

또한 Epochs은 300으로 설정한 후, RNN 모델의 학습을 진행할 때, 이전 Epoch 때와 비교해서 오차가 증가했다면 해당 Epoch을 기록하고 모델을 저장하는 방식으로 구현하였다.

① Max length의 변경

일반적으로 RNN에서의 Input은 Sequence Data Type이며, 텍스트 분석에 적용하는 경우, 단어의 개수로 설정한다. 하지만, 실제 데이터에서는 문장마다, 게시글마다 사용된 단어의 개수가 모두 다르기 때문에 곧바로 RNN의 Input으로 넣을 수가 없다. 즉, RNN의 입력 길이를 설정해주어야 하는데 Max length 값이 입력하는 단어의 개수를 조절하는 하이퍼 파라미터값이다.

불용어를 제거한 후 문장마다 사용된 단어 개수를 확인해본 결과, 평균 5.46개가 사용되었으며, 중앙값은 4.9로 계산되었다. 따라서 본 연구에서는 Max length를 4, 6, 8, 10으로 값을 변경하며 모델의 성능을 확인해보았다.

실제 학습에서 적용할 때에는, 문장에서의 사용된 단어의 개수가 Max length보다 부족한 경우 앞부분에 부족한 만큼 0을 채워 길이를 맞춰주었으며, 반대로 사용된 단어의 개수가 Max length보다 초과하는 경우 뒷부분의 단어를 제외해 길이를 맞춰주었다.

<표 18> Max length가 6일 때, 입력 데이터 변환 예시

입력 데이터 변환 전	입력 데이터 변환 후	설명
285, 961, 187, 2797, 175, 17	285, 961, 187, 2797, 175, 17	그대로 사용
287, 226, 5435, 1620, 660, 23, ~~369, 2462, 1355~~	287, 226, 5435, 1620, 660, 23	뒷부분 제거
1358, 373, 16, 5	0, 0, 1358, 373, 16, 5	앞부분 0추가

<표 18>은 Max length에 따른 입력 데이터 변환 예시이다. 표 안의 숫자는 토큰화된 단어이며, 사용된 단어의 개수에 따라 입력 데이터를 어떻게 변환하는지 보여주고 있다.

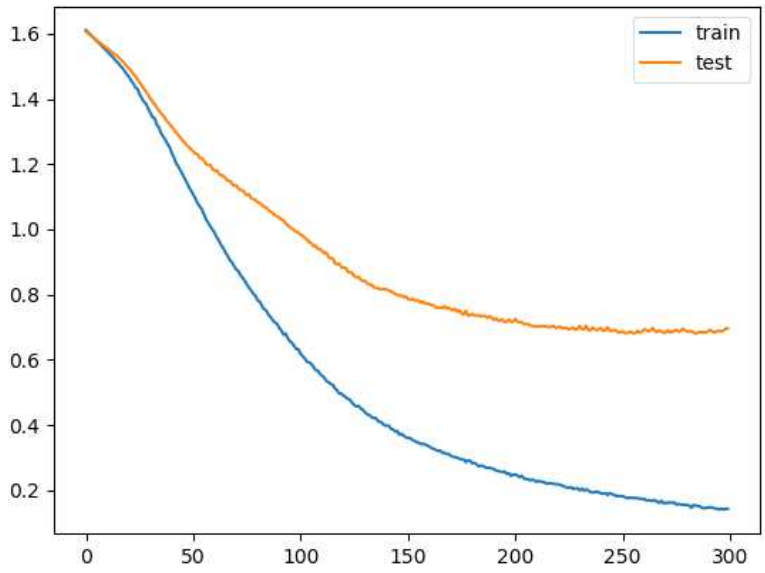
<표 19> Max length의 변경에 따른 정확도(Accuracy) 비교

Max length	Embedding dim	Unit	Classification Accuracy (%)					
			급여	직무	취업 준비	환경 문화	의미 없음	all
4	10	32	0.74	0.66	0.75	0.75	0.82	0.74
6	10	32	0.78	0.69	0.76	0.77	0.85	0.76
8	10	32	0.82	0.67	0.75	0.74	0.85	0.77
10	10	32	0.80	0.78	0.79	0.75	0.86	0.79
12	10	32	0.78	0.77	0.79	0.75	0.84	0.78
14	10	32	0.78	0.73	0.79	0.74	0.85	0.78

<표 19>는 Max length의 변경에 따른 정확도를 비교한 결과이다. Max length는 4부터 2씩 증가하여 14까지 총 6개의 모델을 비교하였으며, Embedding dim과 Unit은 일단 각각 10, 32로 고정한 후 결과값을 확인하였다. 정확도(Accuracy)는 '급여', '직무', '취업준비', '환경문화', '의미없음' 카테고리 각각의 분류 정확도와 전체의 분류 정확도를 표기하였다. 그리고 이때 계산된 정확도는 검증 데이터(Validation Data)에서의 정확도를 의미한다.

전체의 분류 정확도(all)를 확인해보면 Max length가 4일 때 74%로 가장 낮았으며 Max length가 10일 때 79%로 가장 높았다. 그리고 Max length를 12, 14로 높였을 때 정확도가 둘 다 78%로 Max length가 10일 때보다 오히려 정확도가 낮아진 것을 확인할 수 있다. 이는 사용된 단어 개수의 평균이 5.46개로 낮기 때문에 Max length를 10 이상으로 키웠을 때, 오히려 학습에 부정적인 영향을 주는 것으로 판단할 수 있다. 따라서 RNN 기반의 문장 범주 예측 모델에서 Max length 값은 10으로 결정하였다.

<그림 12> Max length가 10일 때의 Loss 그래프
(x축 : Epoch, y축 : Loss)



<그림 12>는 Max length를 10으로 설정한 후, Epoch이 증가함에 따라 Loss가 얼마나 감소하는지를 나타내는 그래프이다. 그래프의 x축과 y축은 각각 Epoch와 Loss이다. 그리고 파란색 선은 Train Data에 대한 Loss값의 변화를 의미하고 있으며 주황색 선은 Test Data에 대한 Loss값의 변화를 의미한다.

그래프를 살펴보면 Train Data는 Epoch이 증가함에 따라 계속해서 Loss가 감소하는 것을 확인할 수 있지만, Test Data는 Epoch이 200을 넘어가는 순간 선의 기울기가 수평에 가까워짐을 확인할 수 있다.

실제로 Epoch을 300 이상으로 설정해본 결과, Test Data에 대한 Loss값이 오히려 증가하는 것을 확인할 수 있었는데 이는 Train Data에 과적합(Overfitting)된 결과라고 볼 수 있다.

② Embedding dim의 변경

다음은 Embedding dim의 변경에 따른 오차 및 성능의 비교를 진행하였다. 임베딩 차원의 변경은 토큰화된 단어를 N차원의 벡터공간으로 변환하는 과정을 의미하며, 값이 높아질수록 모델이 복잡해지고 성능이 좋아지지만 차원이 과도하게 높아지면 과적합 문제가 발생할 수 있다.

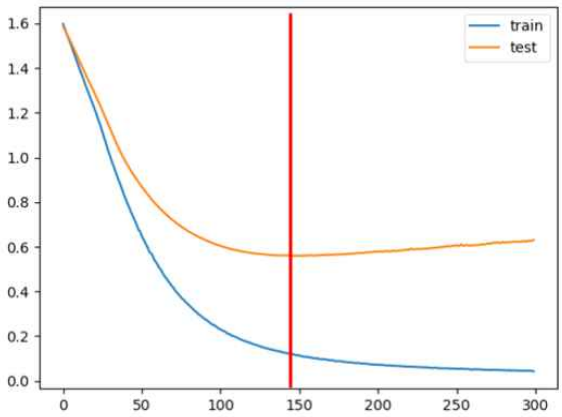
본 연구에서는 Max length와 Unit을 각각 10, 32로 설정한 후에 Embedding dim 값을 5, 10, 15, 30 ,50으로 변경하며 모델의 성능을 확인하였다. 모델의 성능을 확인한 결과는 <표 16>과 같다.

<표 20> Embedding dim의 변경에 따른 정확도(Accuracy) 비교

Max length	Embedding dim	Unit	Classification Accuracy (%)					
			급여	직무	취업 준비	환경 문화	의미 없음	all
10	5	32	0.81	0.65	0.78	0.76	0.84	0.76
10	10	32	0.79	0.73	0.83	0.76	0.86	0.79
10	15	32	0.86	0.76	0.85	0.84	0.85	0.83
10	30	32	0.91	0.76	0.86	0.84	0.84	0.84
10	50	32	0.88	0.75	0.81	0.82	0.85	0.82

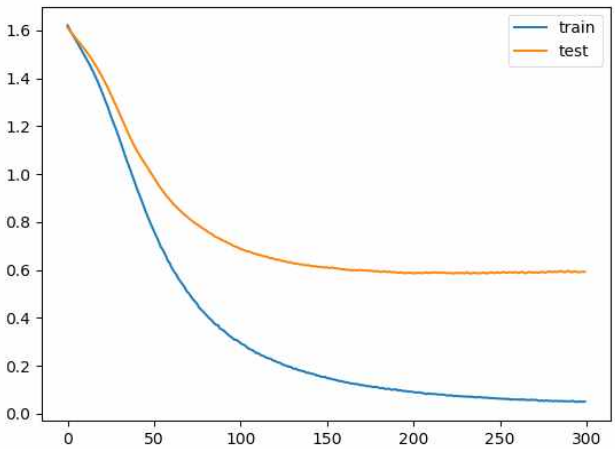
<표 20>의 전체의 분류 정확도(all)를 확인해보면 Embedding dim이 5일 때 76%로 가장 낮았으며 Embedding dim이 30일 때 84%로 가장 높았다. 그리고 Embedding dim을 50로 높였을 때 정확도가 82%로 Embedding dim이 30일 때보다 오히려 정확도가 낮아진 것을 확인할 수 있는데, 이는 과적합 문제가 발생하였기 때문으로 판단된다. 실제로 <그림 13>을 보면 Epoch의 크기가 145를 넘어갈 때 Test Data의 Loss가 더 증가하는 것을 확인할 수 있다. 이는 Epoch이 증가함에 따라 오히려 성능이 떨어지고 있음을 의미한다.

<그림 13> Embedding dim이 50일 때의 Loss 그래프
(x축 : Epoch, y축 : Loss)



따라서 RNN 기반의 문장 범주 예측 모델에서 Embedding dim 값은 30으로 결정하였다.

<그림 14> Embedding dim이 30일 때의 Loss 그래프



<그림 14>는 Embedding dim이 30일 때의 Loss 그래프를 확인한 결과이다. 그래프를 살펴보면 Train Data는 Epoch이 증가함에 따라 계속해서 Loss가 감소하는 것을 확인할 수 있지만, Test Data는 Epoch이 170을 넘어가는 순간 선의 기울기가 수평에 가까워짐을 확인할 수 있다.

③ Unit의 변경

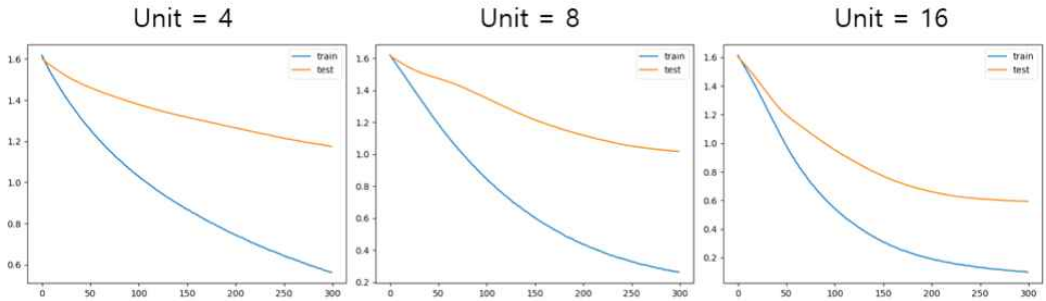
마지막으로 Unit의 변경에 따른 오차 및 성능의 비교를 진행하였다. Unit은 RNN 신경망에 존재하는 뉴런의 개수를 의미한다. 본 연구에서는 Max length와 Embedding dim을 각각 10, 30로 설정한 후에 Unit의 값을 4, 8, 16, 32, 64, 128로 변경하며 모델의 성능을 확인하였다. 모델의 성능을 확인한 결과는 <표 21>과 같다.

<표 21> Unit의 변경에 따른 정확도(Accuracy) 비교

Max length	Embedding dim	Unit	Classification Accuracy (%)					
			급여	직무	취업 준비	환경 문화	의미 없음	all
10	30	**4**	0.56	0.53	0.52	0.74	0.72	0.60
10	30	**8**	0.82	0.57	0.61	0.73	0.83	0.69
10	30	**16**	0.73	0.72	0.74	0.85	0.87	0.81
10	30	32	0.91	0.76	0.86	0.84	0.84	0.84
10	30	**64**	0.88	0.76	0.84	0.86	0.85	0.84
10	30	**128**	0.87	0.77	0.87	0.83	0.85	0.84

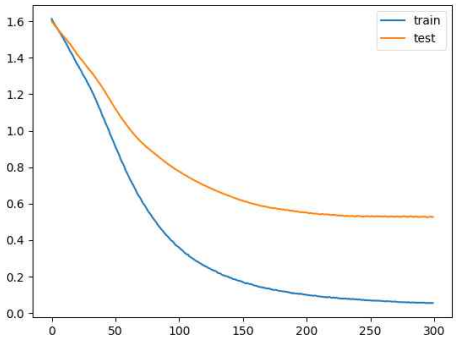
<표 21>의 전체의 분류 정확도(all)를 확인해보면 Unit이 4일 때 60%로 가장 낮았으며 Unit이 32, 64, 128일 때 모두 84%로 가장 높았다. 일반적으로 Unit은 인공신경망에서 뉴런의 개수를 의미하므로 값이 커질수록 모델이 매우 복잡해진다. 따라서 정확도가 가장 높으면서 Unit의 수가 낮은 32를 최적의 값으로 판단하였다.

<그림 15> Unit이 4, 8, 16일 때의 Loss 그래프



<그림 15>는 Unit의 수가 4, 8, 16일 때의 Loss 그래프를 출력한 결과이다. Unit 4와 8에 해당하는 그래프의 선은 Epoch가 300에 도달해도 Loss의 기울기가 0으로 수렴하지 않는 것을 확인할 수 있다. 이는 Epoch가 300일 때에도 학습이 완벽하게 이뤄지지 않았음을 의미하며, 실제로 <표 21>과 같이 정확도가 각각 60%, 69%로 매우 낮게 나타났다. 반면 Unit이 16인 경우 Epoch이 300에 가까워질수록 기울기가 0에 가까워진 것을 확인할 수 있으며, Unit 4, 8보다 상대적으로 학습이 제대로 이루어졌다고 판단할 수 있다.

<그림 16> Unit이 32일 때의 Loss 그래프



<그림 16>은 Unit이 32일 때의 Loss 그래프이며, Test Data는 Epoch이 240을 넘어가는 순간 선의 기울기가 수평에 가까워짐을 확인할 수 있다.

최종적으로 RNN 기반의 문장 카테고리 예측 모델의 Max length, Embedding dim, Unit값은 각각 10, 30, 32로 결정하였으며 RNN 구조는 <표 22>와 같다.

<표 22> RNN 기반의 문장 카테고리 예측 모델의 구조

Layer (type)	Output Shape	Param num
input_1 (InputLayer)	(None, 10)	0
embedding_1 (Embedding)	(None, 10, 30)	165,000
simple_rnn_1 (SimpleRNN)	(None, 32)	2,016
dense_1 (Dense)	(None, 5)	165

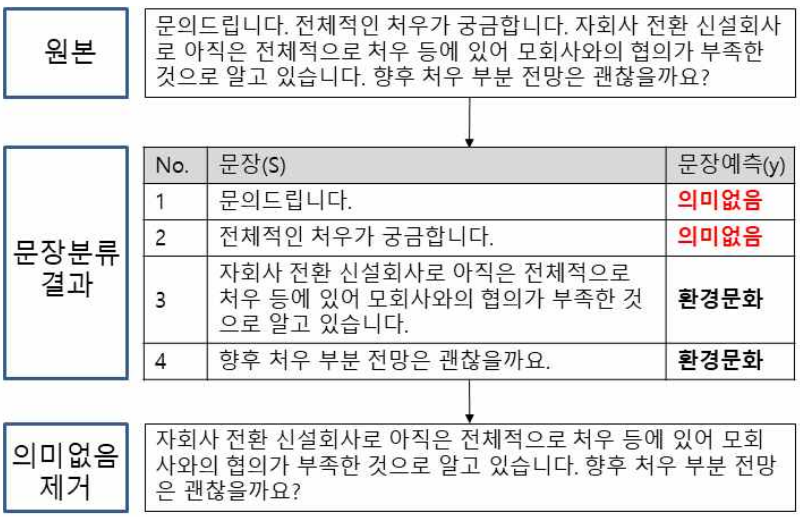
(3) RNN 기반의 문장 단위 범주 예측 모델 활용

본 연구에서는 18,779개의 문장 데이터 중에서 5,000개의 문장을 가지고 RNN 기반의 문장 카테고리 예측 모델을 설계하였으며, 모델을 활용하여 카테고리가 정해지지 않은 13,779개의 문장의 카테고리를 분류해주었다. 그다음, 문장의 카테고리 분류 결과를 가지고 ① '의미없음' 문장제거 및 ② 카테고리별 가중치(W) 계산 2가지 작업을 진행하였다.

① 게시글에서 '의미없음' 문장제거

텍스트 정보를 가지고 해당 문서의 카테고리를 분류할 때, 일반적으로 하나의 게시글에서 사용된 텍스트의 양이 커질수록 카테고리 분류 성능이 낮아지게 된다. 그 이유는 텍스트 안에 다양한 주제가 포함되어있을 가능성이 커지기 때문이다. 또한 사람들은 글을 작성할 때, 처음부터 끝까지 핵심적인 내용만을 작성하지 않는다. 그렇기에 글의 핵심, 즉 목적과 상관없는 의미 없는 문장을 제외한 후 학습을 진행한다면 카테고리 분류 성능이 개선될 수 있다.

<그림 17> '의미없음' 문장 제거 처리결과 예시



지금까지 1단계 과정 즉, 게시글을 문장 단위로 분리하고 각 문장의 카테고리를 분류한 이유는 의미 없는 문장을 찾아 제외하기 위함이다. 따라서 RNN 기반의 문장 범주 예측 모델을 통해 18,779개의 문장에 대해 모두 카테고리를 분류한 후, 의미 없는 문장을 제외하고 다시 게시글 데이터로 취합하였다. <그림 17>은 실제로 게시글 데이터에서 문장의 카테고리를 분류하고 '의미없음' 문장을 제거하는 과정을 보여준다.

의미 없는 문장이 제외된 게시글 데이터는 2단계, LSTM 기반의 게시글 범주 예측 모델의 Input 데이터로 이용된다.

② 카테고리별 가중치(W) 계산

본 연구에서는 1단계 RNN 기반의 문장의 카테고리 분류 결과를 활용하여 2단계 LSTM 기반의 게시글의 범주 예측에 이용하기 위한 '카테고리별 가중치(W)를 계산한다.

<표 23> 게시글마다의 카테고리별 문장의 개수 Count 결과

게시글 번호	카테고리별 문장의 개수 Count			
	급여	취업준비	환경문화	직무
1	0	0	0	2
2	1	0	1	0
3	4	2	0	1
4	1	4	0	0
		⋮		
5,492	0	0	0	5

1단계 과정이 끝나면 모든 문장은 카테고리값이 정해져 있다. 이를 2단계 과정에 활용하기 위하여 우선 <표 23>처럼 5,492개의 게시글 모두 카테고리별 문장의 개수를 더해주었다. 하지만, 단순히 카테고리별 문장의 개수를 더해준 값을 가중치(W)로 쓰기에는 게시글마다 사용된 문장의 개수가 다르기 때문에 일반화할

수 없다는 한계가 있다. 따라서 본 연구에서는 Softmax 함수를 적용하여 카테고리별 가중치(W)를 만들고자 하였다. Softmax란, 입력받은 값을 0~1 사이로 정규화하여 출력값들의 총합이 항상 1이 되도록 해주는 함수이며, <식 3>과 같다.

<식 3>

$$f(\vec{x})_i = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \quad \text{for } i = 1, \ldots, K$$

<표 24> Softmax 함수를 적용하여 만든 카테고리별 가중치 값(W)

게시글 번호	카테고리별 가중치 값(W)			
	급여(w1)	취업준비(w2)	환경문화(w3)	직무(w4)
1	0%	0%	0%	100%
2	50%	0%	50%	0%
3	50%	33.4%	0%	16.7%
4	20%	80%	0%	0%
		⋮		
5,492	0%	0%	0%	100%

<표 24>는 실제로 Softmax 함수를 적용하여 만든 카테고리별 가중치 값(W)이다. Softmax함수를 적용함으로써 실제로 모든 값이 0에서 1사이에 정규화된 것을 확인할 수 있다. 이렇게 도출한 값은 2단계, LSTM 기반의 게시글 범주 예측 모델의 결과값에 한 번 더 곱해주어 최종 카테고리 예측에 활용한다.

<식 4>

$$Category(y) = Max\left(\begin{bmatrix} Output1 \\ Output2 \\ Output3 \\ Output4 \end{bmatrix} + \begin{bmatrix} Output1 \\ Output2 \\ Output3 \\ Output4 \end{bmatrix} \times \begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \end{bmatrix} \times \alpha \right)$$

Category(y) : 카테고리 선정 값

Max( ) : 최댓값 출력 함수

Output : LSTM Dense Layer의 Output 값(카테고리 개수만큼 존재)

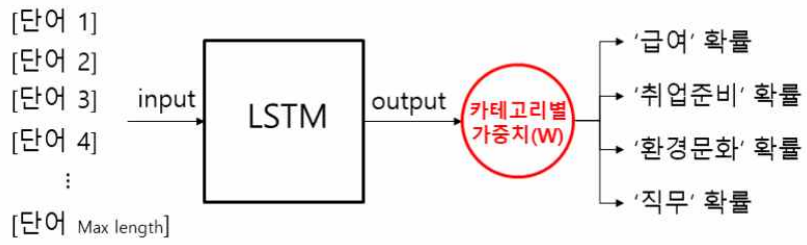
w : 카테고리별 가중치 값

α : w 조절 하이퍼 파라미터(α >0)

<식 4>는 본 연구에서 LSTM 기반의 게시글 범주 예측 모델(2단계)의 결과값에 카테고리별 가중치 값을 반영하는 식을 정리한 것이다. <식 4>에서 Output은 LSTM의 마지막 출력의 값이며, 카테고리별 확률값으로 도출된다. W는 본 연구에서 계산한 카테고리별 가중치 값이다. 마지막으로 α는 w의 영향력을 조절하는 하이퍼 파라미터로 0 이상의 값을 갖는다. 만약 α가 0이라면 w를 반영하지 않은 것과 같으며, α를 높일수록 w의 영향력이 높여줄 수 있다.

<식 4>를 자세히 살펴보면 Output과 w를 곱한 결과를 Output에 더하는 것을 확인할 수 있는데, 이를 통해 Output 결과값에 w만큼의 영향력을 높여주는 기능을 수행할 수 있다.

<그림 18> 카테고리별 가중치(W) 적용 프로세스 설명



<그림 18>은 2단계, LSTM 기반의 게시글 범주 예측 모델의 결과에 카테고리별 가중치를 계산하는 프로세스를 보여주고 있다. LSTM 모델에 게시글 데이터가 Input으로 들어가면, 카테고리별 확률이 Output으로 출력된다. 그다음, Output으로 나온 확률값에 카테고리별 가중치를 곱해주어 최종 카테고리를 결정한다.

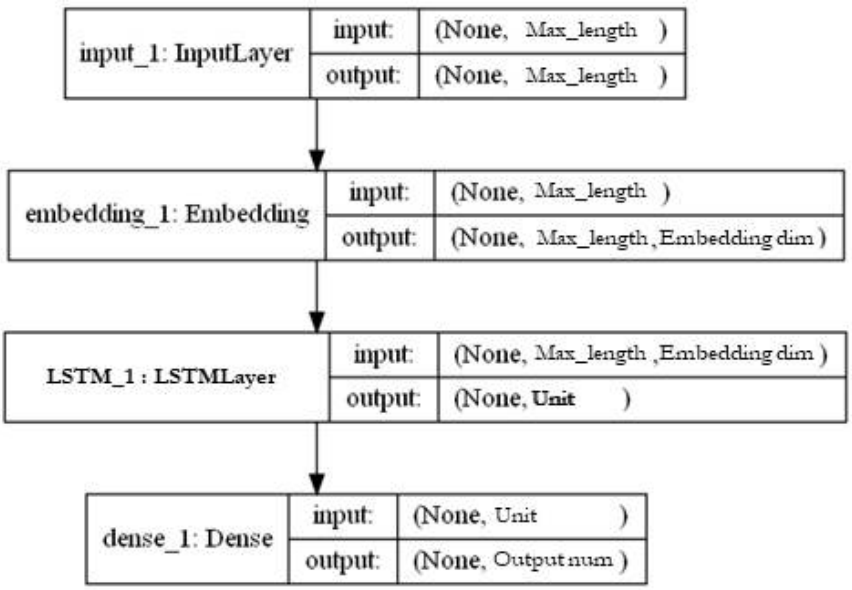
정리하자면, 본 연구에서 정의하는 '카테고리별 가중치 값의 역할'은 문장 단위로 확인했을 때, 많이 나타난 카테고리에 (+)가중치를 부여하는 것이다. 이를 통해 긴 글을 학습시킬 때, 성능이 저하되는 문제를 보완할 수 있을 것으로 기대한다.

3. 2단계 : LSTM 기반의 게시글 범주 예측

(1) LSTM 모델의 설계

이번 장에서 설명하는 2단계, LSTM 기반의 게시글 범주 예측 모델이 본 연구의 최종 목적인 게시글마다 카테고리를 분류해주는 단계이다. 1단계 문장 단위와 달리 게시글 단위는 사용된 단어의 개수가 많아지기 때문에 장기의존성 문제를 보완할 수 있는 LSTM 알고리즘을 이용하였다.

<그림 19> LSTM 기반의 게시글 범주 예측 모델 도식화



<그림 19>를 보면 모델의 전체적인 모습은 RNN과 똑같은 것을 확인할 수 있다. 다만, 3번째 Layer에 RNN 대신 LSTM이 들어간 것 확인할 수 있다. 이번 장에서 언급하는 LSTM 모델은 모두 <그림 19>의 프로세스로 설계하였으며 Input Data 및 Output 가중치 부여 등 변화를 주어 성능을 비교하였다.

(2) 모델 소개

본 연구에서는 '사람인(saramin)의 취준진담 커뮤니티'에 올라온 5,492개 게시글의 카테고리를 분류하기 위하여 총 4가지 모델을 설계하고 성능을 비교하였다.

<표 25> 게시글 카테고리 분류 모델 정의

구분	분석 알고리즘	Input Data 및 특징
Model_1	단순 LSTM 모델	- 게시글 데이터 전체
Model_2	문장단위 RNN 모델	- 문장예측 결과만 이용 - 단순 최대 값으로 게시글 분류
Model_3	RNN-LSTM 모델 ①	- '의미없음 '문장을 제외한 게시글 데이터
Model_4	**RNN-LSTM 모델 ②**	**- '의미없음' 문장을 제외한 게시글 데이터 - LSTM의 Output에 카테고리 가중치 반영**

<표 25>는 게시글 카테고리 분류를 위한 4가지 모델을 정의한 표이다.

Model_1은 게시글 데이터를 별다른 가공 없이 LSTM의 Input Data로 활용한 모델로, 다른 모델과 성능을 비교하기 위해 설계한 단순 LSTM 모델이다.

Model_2는 1단계에서 문장의 카테고리를 분류한 결과를 가지고, 게시글마다 가장 많이 나타난 문장의 카테고리를 해당 게시글의 최종 카테고리로 분류한 모델이다. 즉, 단순히 문장 카테고리의 합이 가장 높은 카테고리가 게시글의 최종 카테고리로 결정되는 모델로, LSTM을 활용하지 않고 단순 Max()함수를 활용한 모델이다.

Model_3은 1단계, 문장의 카테고리를 분류한 결과에서 '의미없음' 문장을 게시글 데이터에서 삭제한 후 나머지 게시글 데이터를 LSTM의 Input Data로 활용하는 RNN-LSTM 모델이다. 의미 없는 문장을 삭제함으로써 Input Data

가 단순 LSTM 모델보다 줄어드는 특징을 갖고 있다. 입력 데이터의 길이가 길수록 성능이 저하되는 문제를 해결하기 위해 본 연구에서 고안한 방법이며, 의미 없는 문장을 제거함으로써 학습의 정확성 및 효율성을 높일 수 있을 것으로 기대한다.

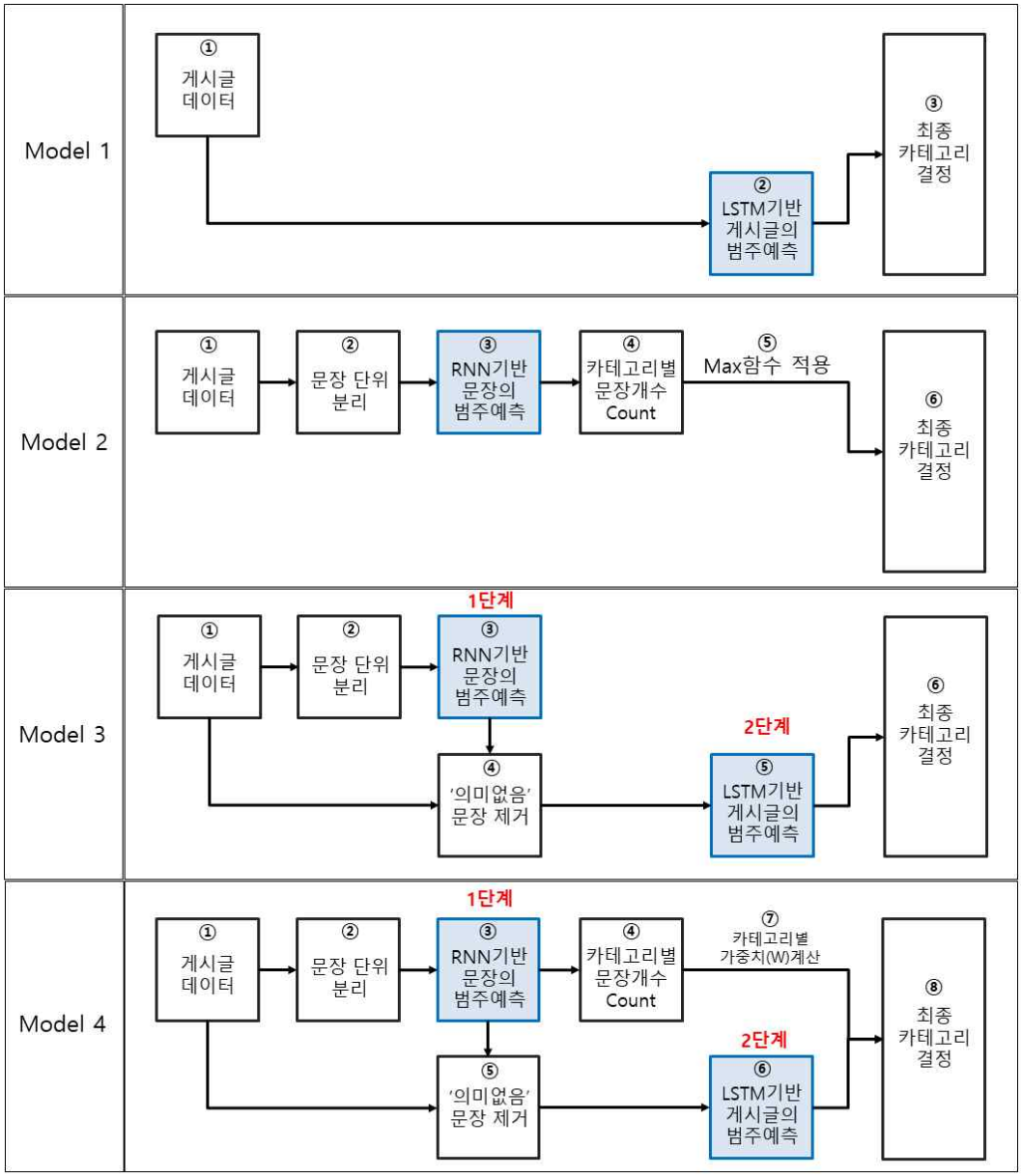
마지막으로 Model_4는 Model_3과 마찬가지로 '의미없음' 문장을 게시글 데이터에서 삭제한 후 나머지 게시글 데이터를 LSTM의 Input Data로 활용하는 RNN-LSTM 모델이다. 하지만 Model_3와 달리 LSTM의 Output 결과에 카테고리별 가중치(W)를 한 번 더 계산하여 최종 카테고리를 예측한다. 여기서 카테고리별 가중치(W)는 4.2.1에서 구한 가중치 값으로 문장의 카테고리 분류 결과를 반영시켜주는 역할을 한다.

본 연구는 4가지 Model의 카테고리 분류 성능을 확인함으로써 의미 없는 문장을 제거하였을 때, 그리고 문장의 카테고리 분류 결과를 반영하였을 때 카테고리 분류의 성능이 높아지는지에 대해 검증하고자 하였다.

또한 각 모델의 성능을 객관적으로 비교하기 위하여 Epochs, Embedding dim, unit 등 하이퍼 파라미터를 동일한 값으로 맞추어 모델을 설계해주었다. 모델의 성능은 정오분류표(confusion matrix)를 통해 정확도를 비교하는 방식으로 검증하였다. 모든 Model에서 Epochs과 Embedding dim, Unit은 각각 300, 30, 32로 동일하게 설정하였다. 다만, Max length의 경우 모델마다 사용된 단어의 길이가 다르므로 값을 바꾸어가며 모델별 최적값을 설정해주었다.

<그림 20>은 Model 별 분석 프로세스를 비교한 그림이다.

<그림 20> Model 별 분석 프로세스 비교



(3) 모델의 성능 평가

① Model_1

Model_1은 게시글 데이터를 별다른 가공 없이 LSTM의 Input Data로 활용하였다. Model_1에서는 Max length가 30일 때 분류 성능이 가장 좋았으며, 최종적으로 Max length 값을 30으로 두고 모델을 구현하였다. 학습-테스트 비율(Train-Test ratio)를 0.3으로 설정하였으며, 성능의 확인은 Test Data에 대한 분류 정확도를 확인하였다. <표 26>은 Model_1의 학습 결과를 정오분류표로 작성한 표이다.

<표 26> Model_1의 confusion matrix

Model_1 confusion matrix		실제 카테고리(Y)			
		급여	직무	취업준비	환경문화
예측한 카테고리 ($\hat{Y}$)	급여	244	54	39	61
	직무	29	296	66	47
	취업준비	19	83	257	39
	환경문화	38	61	34	281
Classification Accuracy(%)		73.9%	59.9%	64.9%	65.7%
Total Accuracy(%)					**65.4%**

<표 26>은 Model_1의 성능을 확인하기 위한 정오분류표이다. 정오분류표를 보면 전체 정확도는 65.4%로 나타난 것을 확인할 수 있다. '급여' 카테고리를 73.9%로 분류 정확도가 높았으나 '직무' 카테고리가 59.9%로 매우 낮은 것을 확인할 수 있다. 실제 '직무'지만 '급여', '취업준비', '환경문화'로 예측한 건이 각각 54, 83, 61건으로 다른 카테고리보다 높은 것을 확인할 수 있다. 이는 '직무' 카테고리가 명확한 특징이 없기 때문에 타 카테고리로 오분류하는 확률이 높다고 추측된다.

② Model_2

Model 2는 1단계에서 문장의 카테고리를 분류한 결과를 가지고, 게시글마다 가장 많이 나타난 문장의 카테고리를 해당 게시글의 최종 카테고리로 분류한 모델이다. 즉 게시글 데이터를 가지고 LSTM을 돌린 모델이 아니라, 단순히 문장 카테고리의 합이 가장 높은 카테고리를 게시글의 최종 카테고리로 결정한 결과이다. 다만, 카테고리의 합이 같은 경우가 발생하면 먼저 등장한 문장의 카테고리로 게시글의 카테고리를 결정하였다. <표 27>은 Model_2의 정오분류표이다.

<표 27> Model_2의 confusion matrix

Model_2 confusion matrix		실제 카테고리(Y)			
		급여	직무	취업준비	환경문화
예측한 카테고리 ($\hat{Y}$)	급여	252	65	29	60
	직무	28	284	64	40
	취업준비	15	93	268	35
	환경문화	35	52	35	293
Classification Accuracy(%)		76.4%	57.5%	67.7%	68.5%
Total Accuracy(%)					**66.6%**

<표 27>을 보면 전체 분류정확도가 66.6%인 것을 확인할 수 있다. 이는 Input Data에 별다른 가공 없이 LSTM을 돌린 Model_1보다 1.5% 정확도가 높은 결과이며, 문장의 카테고리만을 가지고 예측한 Model_2가 게시글 전체를 LSTM의 모델에 넣고 예측한 Model_1과 유사하거나 좀 더 성능이 좋다고 볼 수 있다.

③ Model_3

Model_3은 1단계, 문장의 카테고리를 분류한 결과에서 '의미없음' 문장을 게시글 데이터에서 삭제한 후 나머지 게시글 데이터를 LSTM의 Input Data로 활용하는 모델이다. Model_3에서는 Max length가 20일 때 분류 성능이 가장 좋았으며, 최종적으로 Max length 값을 20으로 두고 모델을 구현하였다.

<표 28> Model_3의 confusion matrix

Model_3 confusion matrix		실제 카테고리(Y)			
		급여	직무	취업준비	환경문화
예측한 카테고리 ($\hat{Y}$)	급여	269	49	24	52
	직무	14	321	51	38
	취업준비	22	84	301	31
	환경문화	25	40	20	304
Classification Accuracy(%)		81.5%	65.0%	76.0%	71.1%
Total Accuracy(%)					**72.5%**

<표 28>을 보면 전체 분류정확도가 72.5%로 이전의 Model보다 높게 상승한 것을 확인할 수 있다. 실제로 '급여' 카테고리의 분류 정확도는 81.5%로 매우 높은 것을 확인할 수 있었으며, '취업준비' 또한 76%로 높았다. 다만, '직무' 카테고리의 분류정확도는 65%로 다른 카테고리보다는 낮은 정확도를 보였다. 하지만, Model_1과 비교하여 모든 카테고리의 분류정확도가 최소 5%이상 높아졌으며, 전체 정확도는 7.1%상승하였다.

Model_1보다 Model_3이 Input Data가 작아지고, Max length가 줄어들어 모델의 전체 복잡도가 줄어들었음에도 모델의 성능이 향상되었다. 이는 의미 없는 문장을 제거하는 것이 카테고리 분류 성능을 높일 수 있다는 것을 의미한다.

④ Model_4

마지막으로 Model_4는 Model_3에서 카테고리별 가중치(W)를 추가로 계산하는 모델이다. LSTM 모델의 마지막 층은 Softmax 함수에 의해 각 카테고리일 확률값 4개가 출력된다. 일반적으로 4개의 출력값 중에 가장 높은 카테고리를 최종 카테고리로 결정하지만, Model_4에서는 출력값에 결과에 카테고리별 가중치(W)를 한 번 더 계산해주어 최종 카테고리를 예측한다.

<표 29> Model_4의 confusion matrix

Model_4 confusion matrix		실제 카테고리(Y)			
		급여	직무	취업준비	환경문화
예측한 카테고리 ($\hat{Y}$)	급여	274	46	24	43
	직무	18	328	49	40
	취업준비	18	79	303	33
	환경문화	20	41	20	311
Classification Accuracy(%)		83.0%	66.4%	76.5%	72.7%
Total Accuracy(%)					**73.8%**

<표 29>는 Model_4의 정오분류표이다. <표 29>를 보면 전체 분류정확도가 73.8%로 Model_3보다 1.3%가 높아졌으며, Model_1보다 8.4%가 높아진 것을 확인할 수 있다. Model_4에서 적용한 카테고리별 가중치(W)는 하나의 게시글 안에서 많이 등장한 문장의 카테고리에 (+)가중치를 부여하는 역할을 한다. Model_4의 결과를 통해 게시글 하나의 카테고리를 예측할 때, 곧바로 게시글을 LSTM 모델의 Input으로 넣는 것보다는 1단계, 문장의 카테고리를 분류하고, 2단계 의미 없는 문장을 제거한 후 LSTM의 Input으로 사용하는 것이 정확도를 높일 수 있다는 것을 확인할 수 있었다.

(4) 결과

<표 30> 각 모델별 분류 성능 비교

구분	분석 알고리즘	Input Data 및 특징	분류정확도
Model_1	단순 LSTM	- 게시글 데이터 전체	65.4%
Model_2	문장단위 RNN	- 문장예측 결과만 이용 - 단순 최대 값으로 게시글 분류	66.6%
Model_3	RNN-LSTM ①	- '의미없음 '문장을 제외한 게시글 데이터	72.5%
Model_4	RNN-LSTM ②	- '의미없음' 문장을 제외한 게시글 데이터 - LSTM의 Output에 카테고리 가중치 반영	73.8%

<표 30>은 각 모델의 분류 성능을 정리한 표이다. 4가지의 LSTM 기반의 게시글 범주 예측 모델의 성능을 확인해본 결과, Model_1과 Model_2가 각각 65.4%, 66.6%로 성능이 비슷한 것을 확인할 수 있었으며, Model_4의 분류정확도가 73.8%로 가장 높게 나왔다. 단순 LSTM 모델(Model_1)보다 본 연구에서 제안하는 모델(Model_4)의 분류정확도가 8.4%가량 높아진 것을 확인할 수 있다.

Model_4에서의 핵심은 두 가지로 요약할 수 있다. 첫 번째는 문서 내에서 의미 없는 문장을 제거해주는 것이고, 두 번째는 각 문장의 카테고리를 파악하고 많이 등장한 카테고리에 가중치를 부여해주는 것이다. 일반적으로 사람들은 글을 쓸 때, 항상 핵심만 담지 않고 의미 없는 문장을 사용하는 경향이 있다. 이러한 의미 없는 문장은 자동으로 텍스트 분류를 진행할 때에 방해가 될 수 있어서 사전에 의미 없는 문장을 없애주는 과정이 필요하다. 또한, 사람들이 글을 작성할 때, 글의 목적과 관련된 내용이 글의 전체 비중에서 높은 비율을 차지한다. 이러한 이유로 문장 단위로 카테고리를 분류한 결과를 문서의 카테고리 분류에 반영함으로써 성능을 높일 수 있다.

Ⅵ. 결론 및 한계

1. 연구 요약

본 연구에서는 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델을 통해 '사람인(saramin)의 취준진담 커뮤니티'에 올라온 5,492개 게시글의 카테고리를 분류하고 성능을 확인하였다. 본 연구 모델의 첫 번째 단계는 게시글을 문장 단위로 분류하고, 문장 단위의 카테고리를 분류하는 작업이다. 이 과정을 통해 게시글에서 의미 없는 문장을 제거해주었으며, 또한 카테고리별 가중치(W)를 계산한다. 두 번째 단계는 의미 없는 문장이 제거된 게시글을 LSTM의 Input Data로 활용하여 Output을 뽑은 후에 카테고리별 가중치(W)를 추가로 계산하여 최종 카테고리를 분류한다.

실제로 본 연구자는 5,492개의 게시글 데이터를 18,779개의 문장 데이터로 분리한 후 각 문장의 카테고리를 분류하는 RNN 기반의 문장 단위 범주 예측 모델을 설계하였다. 본 연구에서는 Max length, Embedding dim, Unit 등 3개의 하이퍼 파라미터를 조작하며 성능을 확인하였으며 각각 10, 30, 32일 때 모델의 성능이 가장 좋았다.

그다음, RNN 기반의 문장 단위 범주 예측 모델의 결과를 활용하여 게시글 데이터에서 의미 없는 문장을 제거해주었으며, 또한 카테고리별 가중치(W)를 계산해주었다.

마지막으로 본 연구의 핵심인 RNN-LSTM 2단계 텍스트 분류 모델의 성능을 확인하기 위해 <표 26>과 같이 4개의 모델을 설계하고 각 모델의 분류정확도를 확인하였다. 각 모델의 성능을 확인한 결과, 게시글 데이터를 별다른 가공 없이 LSTM의 Input Data로 활용한 Model_1이 65.4%로 성능이 가장 낮았으며, 본 연구에서 제안하는 RNN-LSTM 2단계 텍스트 분류 모델(Model_4)의 정확도가 73.8%로 일반 LSTM 모델보다 8.4% 성능이 향상된 것을 확인할 수 있었다.

2. 시사점 및 한계

일반적으로 LSTM은 RNN에서 발생하는 장기의존성 문제를 해결하기 위해 고안된 알고리즘이다. 하지만, 기대와 다르게 LSTM을 텍스트 분류에 적용하고자 할 때, Time Steps 수, 즉 문서에 사용된 평균 단어의 개수가 높아질수록 성능이 떨어지는 문제가 발생한다. 이러한 이유는 장기의존성 문제와는 별개로 사람들이 글을 쓸 때, 글의 목적만을 다루지 않으며 의미 없는 문장이 같이 쓰이기 때문이다. 본 연구에서는 이러한 문제점을 보완하기 위하여 문장 단위의 학습 결과를 반영한 RNN-LSTM 2단계 텍스트 분류 모델을 고안하였다. RNN-LSTM 2단계 텍스트 분류 모델은 의미 없는 문장을 제거한 후에 학습을 진행하며, 또한 문장 단위의 카테고리 가중치를 계산하기 때문에 문서에 사용된 평균 단어의 수가 높아지더라도 괜찮은 성능을 발휘할 수 있다.

또한 본 연구에서 사용한 '사람인(saramin)의 취준진담 커뮤니티'는 글쓴이가 직접 카테고리를 설정하며, 선택할 수 있는 카테고리 간의 구분이 명확하지 않다는 특징이 있었다. 이러한 특징으로 일반적인 LSTM 모델의 분류 성능이 65.4%로 나타났다. 하지만 본 연구모델을 사용한 결과, 8.4% 높은 73.8%의 분류 성능을 보였다. 따라서 RNN-LSTM 2단계 텍스트 분류모델은 카테고리 간의 구분이 명확하지 않은 상황에서의 방안이 될 수 있다.

본 연구자는 '사람인(saramin)의 취준진담 커뮤니티' 데이터 5,492개만을 가지고 LSTM 모델을 학습하고 성능을 평가하였으나, 이는 딥러닝을 적용하고 성능을 판단하기에는 데이터의 수가 부족하다는 한계가 있었다. 따라서 향후 연구에서는 SNS, 뉴스 데이터 등 대용량의 데이터를 수집하고 본 연구 모델을 적용해 모델의 타당성을 검증하고 연구를 발전시키고자 한다.

참고문헌

<국내문헌>

강승식(2003), 한국어 형태소 분석과 정보검색. 홍릉과학출판사.

국립국어원, 한국어기초사전, https://krdict.korean.go.kr/mainAction.

권철민(2019), 파이썬 머신러닝 완벽 가이드, 위키북스.

기호연・신경식(2020), "양방향 LSTM을 적용한 단어의미 중의성 해소 감정분석," 한국빅데이터학회지, 5(1), 197-208.

길호현(2018), "텍스트마이닝을 위한 한국어 불용어 목록 연구," 우리말글학회 78(-), 1-25.

김도우・구명완(2017), "Doc2Vec과 Word2Vec을 활용안 Convolutional Neural Network 기반 안국어 신문 기사 분류," 정보과학회논문지, 44(7), 742-747.

김동완(2013), "빅데이터의 분야별 활용사례," 經營論叢, 34(-), 39-52.

김양훈・황용근・강태관・정교민(2016), "LSTM 언어모델 기반 안국어 문장 생성," 한국통신학회논문지, 41(5), 592-601.

김영욱・김기현・이홍철(2018), "텍스트 마이닝과 연관성 규칙을 이용안 동적 문서 분류 방법 연구," 한국컴퓨터정보학회논문지, 23(10), 103-109.

김정미・이주홍(2017), "Word2Vec을 활용안 RNN기반의 문서 분류에 관안 연구," 한국지능시스템학회논문지, 27(6), 560-565.

김정숙(2012), "빅데이터 활용과 관련기술 고찰," 한국콘텐츠학회지, 10(1), 34-40.

김판준(2018), "기계학습에 기초안 국내 학술지 논문의 자동분류에 관안 연구," 정보관리학회지, 35(2), 37-62.

나성희・김정인・이은지・김판구(2016), "SNS가 가지는 특징정보를 활용한 단문텍스트 카테고리 분류방법에 관안 연구," 한국정보기술학회논문지, 14(6),

159-165.

박상민・나철원・최민성・이다희・온병원(2018), "Bi-LSTM 기반의 한국어 감성사전 구축 방안," 지능정보연구, 24(4), 219-240.

박성수・이건창(2018), "Word2Vec과 앙상블 분류기를 사용한 효율적 한국어 감성 분류 방안," 한국디지털콘텐츠학회논문지, 19(1), 133-140.

박호연・김경재(2019), "CNN-LSTM 조합모델을 이용한 영화리뷰 감성분석," 지능정보연구, 25(4), 141-154.

백두현・황민규・이민지・우성일・한상우・이연정・황재욱(2020), "텍스트 분류 기반 기계학습의 정신과 진단 예측 적용," 생물정신의학, 27(1), 18-26.

송민(2017), 텍스트마이닝 Text Mining, 서울, 도서출판청람.

유원준(2020), 딥러닝을 이용한 자연어 처리 입문, 위키독스.

이현상・이희준・오세환(2020), "딥러닝 기술을 활용한 악성댓글 분류: Highway Network 기반 CNN 모델링 연구," 한국경영학회, 343-351.

장경애・박상현・김우제(2015), "인터넷 감정기호를 이용한 긍정/부정 말뭉치 구축 및 감정분류 자동화," 정보과학회논문지, 42(4), 512-521.

최다빈・최우석・최상현・이정환(2020), "충북지역 대학생의 중소기업 취업에 대안 인식조사: 텍스트마이닝을 기반으로," 중소기업연구, 42(4), 235-250.

Saito Goki(이복연 옮김)(2017), Deep Learning from Scratch, 한빛미디어.

Wakui Yoshiyuki and Wakui Sadami(박광수 옮김)(2018), 처음 배우는 딥러닝 수학, 한빛미디어.

Yusuke Sugomori(김범준 옮김)(2017), 정석으로 배우는 딥러닝, 위키북스.

<국외문헌>

Andrej Karpathy(2015), *The Unreasonable Effectiveness of Recurrent Neural Networks*, http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Berry, M. J., and Linoff, G. S. (2004). *Data mining techniques: for marketing, sales, and customer relationship management*. John Wiley & Sons

Du, J., Huang,Y., and Moilanen, K.(2020), *"Pointing to Select: A Fast Pointer-LSTM for Long Text Classification,"* Proceedings of the 28th International Conference on Computational Linguistics.

Shmueli, G., Nitin R. Patel and Peter C. Bruce(2011), *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. John Wiley & Sons

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N. and Kingsbury, B.(2012), *"Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,"* IEEE Signal processing magazine, *29(6)*, 82-97.

Hochreiter, S., and Schmidhuber, J. (1997), *"Long short-term memory,"* Neural computation, *9(8)*, 1735-1780.

Han, J., Pei, J., and Kamber, M. (2011), *Data Mining: Concepts and Techniques*.

Liu, L., Liu, K., Cong, Z., Zhao, J., Ji, Y. and He, J.(2018), *"Long length document classification by local convolutional feature aggregation,"* Algorithms, *11(8)*, 109.

Mouthami, K., Devi, K. N. and Bhaskaran, V. M.(2013), *"Sentiment analysis and classification based on textual reviews,"* ICICES, IEEE.

Olah, C. (2015), *Understanding LSTM NetwRelevance orks*, http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Qaiser, S. and Ali, R. (2018), *"Text Mining: Use of TF-IDF to Examine the Relevance of Words to Document,"* International Journal of Computer Applications, *181(1)*, 25-29.

Wang, J. H., Liu, T. W., Luo, X. and Wang, L.(2018), *"An LSTM approach to short text sentiment classification with word embeddings,"* In Proceedings of the 30th conference on computational linguistics and speech processin, 214-223.

Bengio, Y., Courville, A. and Vincent, P. (2013), *"Representation Learning: A Review and New Perspectives,"* IEEE Trans. PAMI, special issue Learning Deep Architectures.

Zaremba, W., Sutskever, I. and Vinyals, O.(2017), *"Recurrent neural network regularization,"* arXiv preprint, 1409-2329.