# 인공신경망을 활용한 뉴스 토픽 분류

텍스트 분류(다중 분류) - Reuters 뉴스 기사 토픽 분류

현지예
융합소프트웨어학부
명지대학교
서대문구, 서울
congping2@google.com

*Abstract*—**1986 년에 공개된 로이터(Reuter) 데이터셋은 짧은 뉴스 기사와 토픽의 집합으로 이루어져 있다. 본 연구는 46 가지 토픽으로 라벨이 달린 11,228 개의 로이터 뉴스로 이루어진 데이터셋을 통해 뉴스 기사의 토픽을 분류하는 모델을 생성하고, 모델의 정확도와 손실을 예측해볼 예정이다. 뉴스는 총 46 가지의 토픽으로 이루어져 있기 때문에 이는 텍스트 분류 중 다중 분류에 속하며, 각 데이터가 정확히 하나의 범주로 분류되기 때문에 단일 레이블 다중 분류 문제라고 할 수 있다.**

*Keywords — text classification, topic classification, neural networks, deep learning, Reuters*

## I. INTRODUCTION

텍스트의 의미론적 내용 또는 주제를 분류하는 것은 자연 언어 처리, 정보 검색, 인공지능, 머신러닝에서 보다 더 중요한 문제들 중 하나이다. 그 중 신문 기사는 특히 그러한 분류를 배울 수 있는 좋은 기회를 제공한다고 할 수 있다. 기사의 의미적 내용은 일반적으로 일관성이 있고, 라벨이 부착된 뉴스 기사는 쉽게 접근할 수 있기 때문이다. 또한 뉴스 기사 토픽을 특정 응용과 연구를 위해 분류하는 것은 상당한 매력이 있다. 뉴스 기사 토픽 분류는 온라인 뉴스의 리포지토리에 대한 기사 자동태깅과 주제별 뉴스 출처 집계(예: 구글 뉴스)를 가능하게 할 뿐만 아니라 뉴스 추천 시스템의 기초를 제공할 수 있다. 좀 더 광범위하게, 뉴스와 미디어의 사회적, 정치적 영향을 고려할 때, 뉴스의 패턴과 편견을 발견하기 위해 프로그래밍 방식으로 뉴스 데이터를 분석하는 데 좋은 영향을 끼칠 수 있다. 복잡한 요인은 뉴스 기사가 종종 여러 주제 레이블에 걸쳐 있다는 것이다. 인간은 기사와 관련된 여러 라벨을 인식하고 올바르게 제공할 수 있지만, 과연 기계 학습 시스템에서도 유사한 결과를 얻을 수 있을까 [4]? 46 가지의 여러 토픽 중 가장 적합한 토픽을 찾기 위해 정확도를 이용하여 표현할 수 있는 모델을 만들어 이를 실현해보고자 한다.

## II. 연구 과정

### A. *데이터셋 다운로드*

로이터(Reuters) 데이터셋은 케라스 데이터셋(Keras datasets)에 포함되어 있다. 구글 코랩(Colab)에서 로이터 데이터셋을 다운로드하고, 단어는 10000 개로 제한한다. 8982 개의 train 데이터와 2246 개의 test 데이터가 있다.

### B. *데이터 탐색*

데이터셋의 샘플은 전처리된 정수 배열이다. 이 정수는 뉴스 기사에 나오는 단어를 나타낸다. 라벨(label)은 토픽의 인덱스로써 정수 0 에서 45 까지 총 46 개로 이루어져 있다.

### C. *벡터화*

One-hot encoding 을 통해 데이터를 벡터로 변환하는 벡터화를 진행한다. 학습에 사용되는 데이터셋의 input 데이터는 뉴스 기사에 들어가 있는 해당 단어의 인덱스를 1.0 으로 변경시킨다.

### D. *모델 생성*

데이터셋의 단어 수를 10000 개로 제한했기 때문에 신경망의 입력 차원 수도 10000 으로 설정한다. 출력 클래스의 개수가 토픽의 개수인 46 개이기 때문에 마지막 층은 출력 클래스에 대한 확률 분포를 나타내는 Softmax 활성화 함수를 사용한다. 중간 층의 규모가 마지막 층인 46 보다 작으면 유용한 정보를 잃게 되는 정보 병목 현상이 발생할 수 있다. 따라서 중간층을 46 보다 충분히 큰 값(각각 128, 64, 64)을 가진 3 개의 Dense layer 로 쌓고, Relu

활성화 함수를 사용한다. 또한 과적합(Overfitting)을 막기 위해 각 층 사이마다 Dropout 을 0.3 만큼 적용한다.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               1280128
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                4160
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 46)                2990
=================================================================
Total params: 1,295,534
Trainable params: 1,295,534
Non-trainable params: 0
_________________________________________________________________
```

### *E. 모델 컴파일*

모델을 컴파일할 때는 옵티마이저(optimizer)와 손실 함수(loss function)가 필요하다. optimizer 는 가장 빠르고 효과가 좋은 adam 을 사용한다. 손실 함수는 출력층이 여러 개이고 모델이 확률을 출력하므로 다중 분류에서 주로 사용되는 categorical_crossentropy 를 사용한다. 이 함수는 두 확률 분포 사이의 거리를 측정한다. 여기서는 정답인 타깃 분포와 예측 분포 사이의 거리이다. 두 분포 사이의 거리를 최소화하면 진짜 레이블에 가능한 가까운 출력을 내도록 모델을 훈련하게 된다.

### *F. 검증 데이터 설정*

검증을 위해 train 데이터에서 1000 개의 샘플을 분리하여 validation 데이터로 구성한다.

### *G. 모델 학습*

512 개의 샘플로 이루어진 미니배치(mini-batch)에서 10 번의 에폭(epoch) 동안 훈련한다. 훈련하는 동안 10000 개의 검증 세트에서 모델의 정확도와 손실을 모니터링한다.

### *H. 모델 평가*

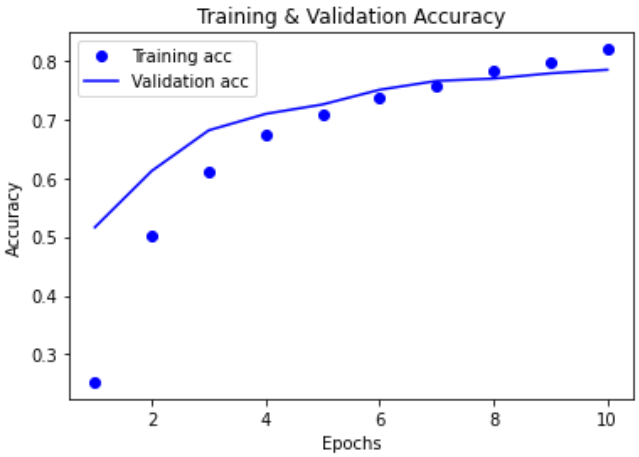
학습한 모델의 성능을 확인하는 것으로, test 데이터를 사용하여 모델을 평가한다. 정확도와 손실, 두 개의 값이 반환된다. Overfitting 이 일어나지 않으면서 정확도가 높고 손실이 낮은 것이 좋은 모델이라고 할 수 있다.

## III. 연구 결과 및 분석

생성한 모델로 학습을 진행하고 모델을 평가해본 결과, 약 1.15 의 손실과 75%의 정확도를 보여주었다.

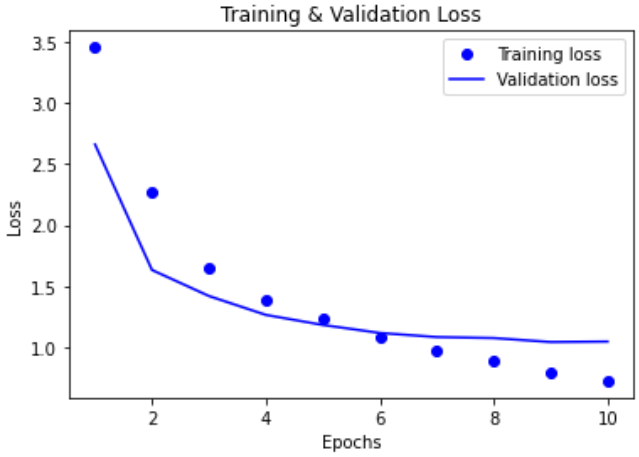
학습이 제대로 진행되고 있는지, Overfitting 이 일어나지는 않는지 확인해보기 위해 학습 결과를 그래프로 시각화하여 나타내 본다.

### *A. 정확도(Accuracy)*




training accuracy 는 약 25%에서 82%까지 부드러운 곡선의 형태로 올라가는 것을 볼 수 있고, validation accuracy 는 약 51%에서 78%까지 올라가는 것을 볼 수 있다. 훈련 데이터와 검증 데이터의 격차가 크지 않고 비슷한 양상을 보이기 때문에 과적합이 일어나지 않고 있다는 것을 알 수 있다.

### *B. 손실(Loss)*




training loss 는 약 3.5 에서 0.72 까지 부드러운 곡선의 형태로 내려가는 것을 볼 수 있고, validation loss 는 약 2.7 에서 1.0 까지 내려가는 것을 볼 수 있다. 훈련 데이터와

검증 데이터의 격차가 크지 않고 비슷한 양상을 보이기 때문에 과적합이 일어나지 않고 있다는 것을 알 수 있다.

### *C. 새로운 데이터 예측*

모델을 이용하여 각 뉴스기사에 대한 토픽을 예측해본다. 모델 인스턴스의 predict 메서드는 46 개의 토픽에 대한 확률 분포를 반환한다. 이를 통해 각 뉴스별로 46 개의 토픽에 해당하는 확률 중 가장 큰 값에 접근하여, 어느 정도의 확률로 예측하고 있는지 살펴본다.

predictions[7]의 인덱스 중 3 번째 인덱스가 가장 큰 값을 가지고 있다는 것을 확인하였다. 해당 인덱스의 값을 출력했더니 약 99%의 확률로 모델이 예측한 것을 확인할 수 있었다. 이를 통해 predictions[7]에 해당하는 뉴스 기사는 3 번째 인덱스인 토픽에 99% 확률로 예측할 수 있을 정도로 분명하게 주제가 드러나는 내용일 것임을 유추할 수 있다.

## IV. 결론

참고한 연구들과 차별성을 두기 위해 Dense layer 와 Dropout 기능을 추가하고, 하이퍼파라미터와 에폭을 조절하는 등 여러 시도를 해보았다. 그 결과 Dropout 을 추가하고 에폭을 조절함으로써 Overfitting 을 줄일 수 있었다.

이 모델의 정확도는 80% 이하로 크게 높지 않았고 손실 또한 약 1.15 로 매우 낮지도 않았지만, 하이퍼파라미터와 레이어의 다양한 구성으로 더 개선해 나간다면 보다 더 높은 정확도와 낮은 손실을 보여주는 괜찮은 모델을 만들 수 있을 것으로 보인다.

또한 텍스트 분류 중 단일 레이블 다중 분류 문제를 뉴스 기사 토픽을 분류하는데 적용함으로써 특정 응용과 연구(뉴스 기사 자동태깅, 뉴스 추천 시스템 등)를 위해 이러한 모델이 다양하게 쓰일 수 있을 것으로 기대한다.

텍스트의 의미론적 내용 또는 주제를 분류하는 것이 인간만이 할 수 있는 어려운 일이 아니라, 자연 언어 처리, 정보 검색, 인공지능, 머신러닝 분야에서 보다 더 많이 쓰이고 발전하여 우리 일상에 더 많이 자리 잡을 것으로 보인다.

## REFERENCES


[1] 사이토 고키, 밑바닥부터 시작하는 딥러닝: 파이썬으로 익히는 딥러닝 이론과 구현, 한빛미디어, 2017

[2] Schwartz, Richard, et al. "A maximum likelihood model for topic classification of broadcast news." Fifth European Conference on Speech Communication and Technology. 1997.

[3] Suh, Yirey, et al. "A comparison of oversampling methods on imbalanced topic classification of Korean news articles." Journal of Cognitive Science 18.4 (2017): 391-437.

[4] CHASE, Zach, Nicolas Genain, and Orren Karniol-Tambour. "Learning Multi-Label Topic Classification of News Articles." (2014).

[5] https://www.tensorflow.org/tutorials/keras/text_classification?hl=ko

[6] https://colab.research.google.com/github/tensorflow/docs-l10n/blob/master/site/ko/tutorials/keras/text_classification_with_hub.ipynb#scrollTo=L4EqVWg4-llM

[7] https://tensorflow.blog/%EC%BC%80%EB%9D%BC%EC%8A%A4-%EB%94%A5%EB%9F%AC%EB%8B%9D/3-5-%EB%89%B4%EC%8A%A4-%EA%B8%B0%EC%82%AC-%EB%B6%84%EB%A5%98-%EB%8B%A4%EC%A4%91-%EB%B6%84%EB%A5%98-%EB%AC%B8%EC%A0%9C/

[8] https://codingcrews.github.io/2019/01/14/reuter_multi_classification/

[9] https://wdprogrammer.tistory.com/23