*The Korean Journal of Applied Statistics*
2022, Vol. 35, No. 1, 19–33

DOI: https://doi.org/10.5351/KJAS.2022.35.1.019
Print ISSN1225-066X / Online ISSN 2383-5818

# Semantic analysis via application of deep learning using Naver movie review data

Sojin Kim[a], Jongwoo Song[1,a]

[a]Department of Statistics, Ewha Womans University

**Abstract**

With the explosive growth of social media, its abundant text-based data generated by web users has become an important source for data analysis. For example, we often witness online movie reviews from the 'Naver Movie' affecting the general public to decide whether they should watch the movie or not. This study has conducted analysis on the Naver Movie's text-based review data to predict the actual ratings. After examining the distribution of movie ratings, we performed semantics analysis using Korean Natural Language Processing. This research sought to find the best review rating prediction model by comparing machine learning and deep learning models. We also compared various regression and classification models in 2-class and multi-class cases. Lastly we explained the causes of review misclassification related to movie review data characteristics.

Keywords: semantic analysis, natural language processing (NLP), LSTM, recurrent neural network (RNN), movie review

## 1. 서론

인터넷의 성장과 SNS의 등장으로 텍스트 데이터의 양이 방대해지고 그 중요성 또한 대두되고 있다. 인터넷 이용자들이 온라인에 남긴 글을 보고 우리는 정보를 빠르게 얻을 수 있으며 직접 경험해보지 않아도 특정 상품이나 서비스에 대한 평가를 읽고 파악할 수 있다. 영상 컨텐츠에 대한 리뷰의 중요성 또한 강조되고 있는데 특히 사용자들이 이용할 컨텐츠를 결정할 때 해당 컨텐츠의 평점이나 리뷰를 보고 참고하기 때문이다. 네이버의 영화 탭에 들어가면 현재 상영하는 영화에 대한 평점이나 리뷰를 확인할 수 있다. 이 때 리뷰에서 간혹 '별점 테러' 혹은 '리뷰 알바'라는 단어를 보게 되는 것도 그만큼 영화 관객들이 영화의 평점과 리뷰를 참고해 해당 영화를 볼 것인지 결정하는 경우가 많다는 것을 시사한다.

영화의 평점과 리뷰에 대한 관심이 많아지면서 영화 리뷰의 감성 분석을 수행하는 선행 연구가 많이 진행되어왔다. 초기 텍스트 분석에서는 machine learning을 이용하여 Random Forest, XGBoost, Naïve Bayes 등의 분석을 수행했다. Kharde와 Sonawane (2016)는 machine learning을 이용해 50000개의 영화 리뷰에 대한 SVM 분석을 수행하였다. Parmar 등 (2014)은 Random Forest의 hyperparameters tuning을 통해 영화 리뷰의 감성 분석 성능을 높였다. 그리고 Nayak (2016)은 Naïve bayes, Support Vector Machine, Random Forest Classifier를 활용해 Twitter의 영화 리뷰 데이터 감성 분석을 하고 성능을 비교했다. 최근에는 순환 신경망(recurrent neural network, RNN)을 이용해 문장의 구조를 반영하여 감성 분석을 수행하는 경우가 많다. 특히 문장에서 단어의 순서를 고려한 LSTM 모델과 반대 방향으로의 순서까지 고려한 Bidirectional LSTM, 그리고 CNN과 LSTM을 결합한 Hybrid형태의 CNN-LSTM 모델을 활용하고 있다. Lee 등 (2018)은 deep learning을 이용해 한글을

[1] Corresponding author: Department of Statistics, Ewha Womans University, 52, Ewhayeodae-gil, Seodaemun-gu, Seoul 03760, Korea. E-mail:josong@ewha.ac.kr

Published 28 February 2022 / journal homepage: http://kjas.or.kr

음소 단위로 분할하여 IMDB의 영화 리뷰를 RNN, LSTM, GRU 세 가지 모형을 이용해 감성 분석을 시행하였다. Oh 등 (2019)은 bidirectional LSTM 모델을 이용해 한국어 영화리뷰 감성 분석을 수행했고 Park과 Kim (2019)은 CNN과 LSTM을 결합하여 CNN 모형과 LSTM모형을 상호 보완한 CNN-LSTM 모델을 제안했다. 또한 Rehman 등 (2019)도 Hybrid CNN-LSTM model을 제안하여 영화 리뷰 감성 분석의 성능을 개선하였다. 그럼에도 아직 한국어 자연어 처리는 띄어쓰기와 오타에 따라 분석이 잘 되지 않는 어려움이 있다. 온라인 리뷰 특성 상 띄어쓰기와 오타 문제는 불가피하고, 줄임말이나 유행어의 사용이 많을수록 분석은 더욱 어렵다. 본 연구에서는 네이버 영화 리뷰 및 평점 데이터를 가지고 리뷰의 의미를 분석하는 모형을 구축하고 리뷰 텍스트 데이터를 이용하여 평점을 예측해보고자 한다. 2장에서는 연구 데이터의 EDA(탐색적 자료분석)과정과 자연어 전처리 과정, 그리고 모형 구축 과정에 대해 자세히 서술할 것이다. 3장에서는 모형 구축과정에서는 2-Class Classification model에서 시작하여 점차 확대해가며 다양한 모델의 성능을 비교해 최적의 Classification model을 구축하고자 한다. 먼저 가장 간단한 2-Class Classification에서 기존에 사용하던 머신러닝 방법과 RNN모형을 비교할 것이다. 그 후 10-Class부터 데이터를 그룹화한 4-Class와 3-Class까지 Classification 모형을 순차적으로 적용시킬 것이다. 그리고 그 과정에서 Classification과 Regression 모형, 그리고 두 가지를 결합한 모형의 성능을 비교한다. 또한 두 번의 Classification을 단계적으로 수행하는 2-step 방법론을 통해서도 텍스트 리뷰의 의미를 분석할 것이다. 마지막 4장에서는 모델이 예측하지 못한 오분류 데이터를 확인해봄으로써 오분류의 원인을 서술하고 모델의 한계점을 알아보고자 한다.

## **2.** 연구 데이터 **EDA** 및 자연어 처리 과정

### 2.1. 연구 데이터 EDA

#### *2.1.1. 데이터 설명*

본 연구에서 사용한 네이버 영화리뷰 및 평점 데이터는 2019년 7월부터 2020년 6월까지 1년 간 총 12,481편의 영화에 대한 리뷰와 평점에 대한 자료이다. 데이터는 약 100만개의 리뷰 및 평점을 포함하고 있다. 데이터의 형태와 데이터에 포함된 변수는 Table 1과 Table 2와 같다.

Table 1: Examples of data

| Mcode | Ktitle | Rating | Review | Date |
|---|---|---|---|---|
| 189537 | #살아있다 | 10 | 좀비물 좋아하는 총으로 쏘고 …, | 20200630235926 |
| 189537 | #살아있다 | 1 | 세계를 사로잡은 21세기 한국문화…,? | 20200630235911 |
| 12158 | 토이즈 | 10 | 어렸을때 장난감들끼리 싸우는…, | 20200630235853 |
| 27404 | 4월 이야기 | 10 | 일본은 이런 감성영화는 참 잘 만들어 | 20200630235735 |

Table 2: Description of Variables

| Variable | Class | Description |
|---|---|---|
| Mcode | Categorical | 영화고유코드 |
| Ktitle | Character | 영화제목 |
| Rating | Numerical/Categorical | 영화평점 |
| Review | Character | 영화리뷰 |
| Date | Date | 리뷰작성일시 |

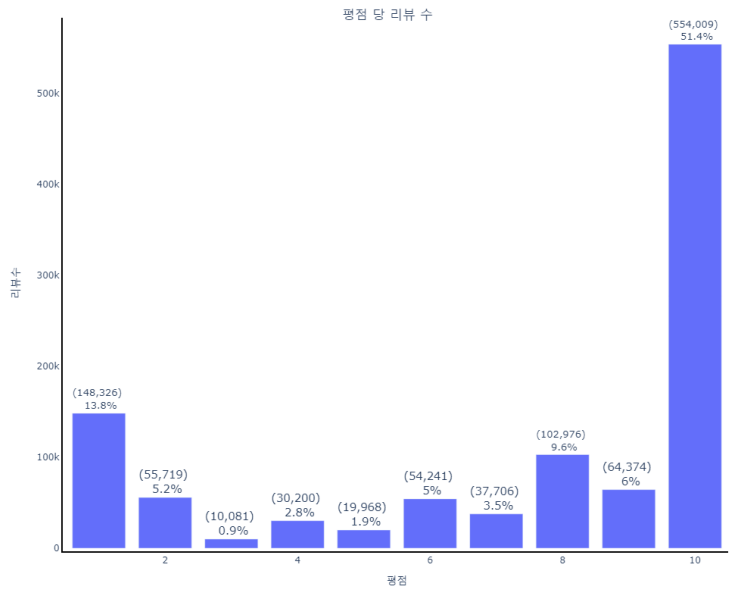


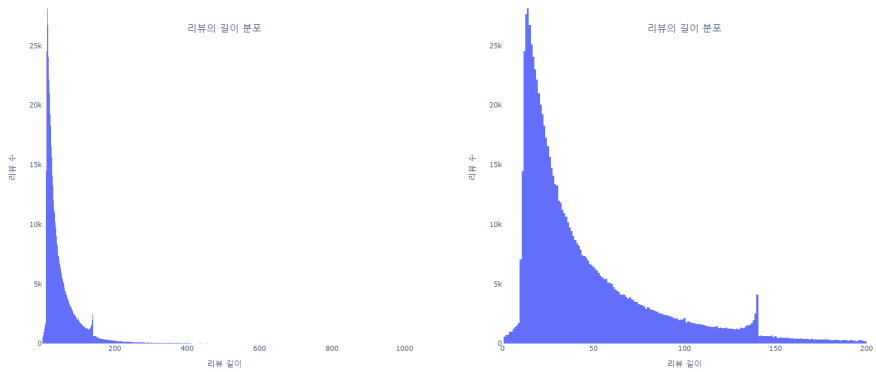

Figure 1: *Number of reviews per movie rating.*

### *2.1.2. 변수 설명*

Rating(평점)변수는 1점부터 10점까지 10개의 class로 구성되어 있으며, 각 평점당 리뷰 수와 비중은 Figure 1과 같다. 해당 데이터의 특이점은 리뷰 수가 가장 많은 평점은 10점이고 그 비중은 50%에 육박한다는 점이다. 다음으로 리뷰 수가 많은 평점은 1점으로 그 비중은 13.8%이다. 즉, 해당 데이터는 양 극단의 값의 빈도가 높은 Bimodal(쌍봉)의 형태를 보인다. 이는 평점이 1점부터 10점까지 있지만 리뷰를 남기는 관객들은 영화에 매우 만족했거나 매우 불만족스러운 상황에 리뷰를 작성하는 경향이 있기 때문이라고 판단했다.
Review(리뷰)변수는 영화의 리뷰가 담긴 텍스트 자료이다. 평균 길이는 약 49이고, 최대 길이는 998이며 리뷰의 길이 분포는 Figure 2의 (a)와 같다. Figure 2의 (b) 그래프에서 리뷰 길이가 200이하 자료들을 자세히 보면 길이가 50 이하인 문장의 길이가 비교적 짧은 자료들이 많은데 이 것은 해당 데이터가 영화의 한 줄 평으로 작성된 데이터이기 때문이다. 또한, (b) 그래프에서 길이 140 부근에서 리뷰의 수가 주변 대비 많은 것을 볼 수 있다. 이것은 네이버 영화에서 직접 리뷰를 입력할 때 140자 정도 작성하면 텍스트 박스가 거의 꽉 차게 되는 것을 보고 리뷰 작성자들이 주어진 텍스트 박스를 넘치지 않게 작성하는 경향이 있다고 해석했다.

### *2.1.3. 영화 별 평점 분포*

다음은 영화 별 평점의 분포를 살펴보았다. 본 연구에서 활용한 데이터에는 텍스트 리뷰없이 평점만 존재하는 자료들이 존재한다. 따라서 영화 별 평점 분포를 확인할 때에는 텍스트 리뷰가 없는 데이터를 포함해 약 108만개의 평점 데이터를 모두 활용할 것이고, 이후 의미 분석을 수행할 때에는 텍스트 리뷰가 없는 데이터는 제외한 후 분석을 수행할 것이다. 총 1,077,600개의 평점 데이터 중 평점이 가장 많은 영화는 58,439개의 평점이 달린 '82년생 김지영'이고, 그 다음으로는 '조커'(36,459개), '백두산'(35,378개), '엑시트'(31,655개), '봉오동 전투'(28,162개), '겨울왕국2'(27,029개)이다. Figure 3에서 영화 '82년생 김지영'의 평점 분포를 보면 1점과




(a) Length distribution (b) Length distribution under 200

Figure 2: *Distribution of the length of movie reviews.*


(a) 82년생 김지영 (b) 조커 (c) 백두산

(d) 엑시트 (e) 봉오동전투 (f) 겨울왕국2

Figure 3: *Distribution of the ratings of each movie.*

10점에 가장 많은 평점이 몰려있는 것을 확인할 수 있다. 반면 평점이 두 번째로 많은 영화 '조커'는 평점 10점의 비중이 매우 높다. 그 외 다른 영화도 평점의 분포가 영화마다 다른 것을 보고 영화의 평균 평점과 분산과의 관계를 확인하고자 했다.

Figure 4 는 평점의 평균과 평점의 분산의 관계를 보기 위한 그래프다. 이 때 평점의 수가 100개 이하인 영화는 평균과 분산 계산 시 왜곡된 값이 산정될 수 있기 때문에 제외했다. 즉, 평점의 수가 100개 이상인 영화의 리뷰

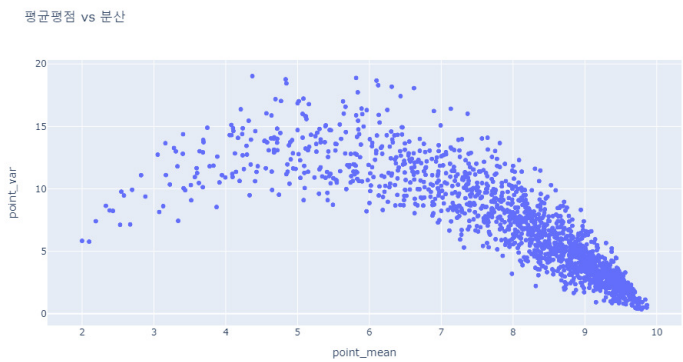



Figure 4: *Variance of rating vs Mean of ratings.*

약 77만개의 데이터를 활용해 그래프를 그린 것이다. 그래프의 $x$축은 영화 별 평점의 평균 값이고, $y$축은 영화 별 평점의 분산을 나타낸다. 그래프를 보면 평균 평점이 5-6일 때 평점의 분산이 크고, 평균 평점 1점, 10점 근처에서는 평점의 분산이 작다. 따라서 평균 평점이 5-6점인 영화는 평점이 5-6점에 몰려있는 것이 아니라 평균 평점이 양 극단에 분포되어 있다는 점을 알 수 있다. 평균 평점이 양 극단에 분포하는 경우는 분산이 크기 때문인데, 영화 '82년생 김지영'이나 '백두산'의 평점 분포가 이러한 경우다. 다음으로는 평균 평점과 영화의 매출액과의 관계를 살펴보았다. 매출액은 Kofic의 박스오피스 매출 데이터를 2019년 7월부터 2020년 6월까지 추출한 값이다. Figure 5 는 누적 관객수가 만 명 이상인 영화들의 평균 평점과 누적 관객수와의 산점도를 그린 것이다. 그래프를 보면 영화의 평균 평점과 누적 관객수는 양의 상관 관계가 있는 듯 하지만 명확한 양의 상관 관계가 있다고 보기는 어렵다. 즉, 평균 평점이 높은 영화가 반드시 누적 관객수가 큰 것은 아니라고 판단했다.

## 2.2. 자연어 처리 과정

텍스트 데이터를 딥러닝 모형에 적용하기 위해서는 자연어 처리 과정이 필요하다. 그 중에서도 한국어를 처리하기 위한 자연어 처리 과정이 요구된다. 자연어 처리는 문장의 Tokenization, Vectorization, 그리고 Padding 과정을 포함한다. 의미 분석에 활용할 데이터의 전처리 과정에서는 약 108만개의 데이터 중 텍스트 리뷰가 존재하는 약 88만개의 데이터만 사용되었다.

### *2.2.1. 전처리(Preprocessing)*

먼저 중복된 텍스트를 모델에 학습하게 하지 않기 위해 중복된 텍스트 데이터를 제외했다. 그리고 텍스트 데이터를 토큰화(Tokenization)하기 전 문장에 있는 한글과 공백을 제외하고는 모두 제거했다. 이 과정에서 느낌표(!), 마침표(.)와 같은 문장 부호는 사라지게 되는데 만약 의미없이 문장 부호만 나열된 리뷰가 있다면 그러한 리뷰는 텍스트가 없는 null값이 되므로 데이터에서 제거해주었다. 따라서 전처리 과정을 거치면 사용할 수 있는 데이터의 수가 크게 줄게 되어 약 86만(855,773)개의 텍스트 데이터만 남게 되었다.

### *2.2.2. 토큰화(Tokenization)*

다음으로 문장을 토큰화(tokenization)하는 작업이 필요하다. Python의 Keras에는 tokenizer 모듈이 있어서 토큰화가 가능하지만 좀 더 정확한 분석을 위해 한국어 자연어처리 모듈인 Konlpy를 사용하였다. Konlpy의 형태소 분석기를 이용하면 문장에서 형태소나 명사 등의 품사로 토큰화가 가능하다. 본 연구에서는 형태소

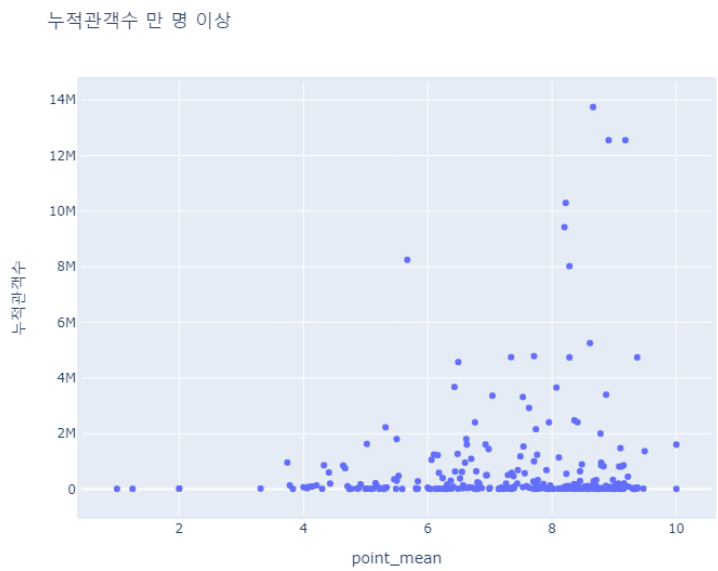



Figure 5: *Cumulative Sales vs Mean of ratings.*

Table 3: Stopwords

| | |
|---|---|
| Stopwords(불용어) | '의','가','이','은','들','는','좀','잘','걍','과','도','를','으로','자','에','와','한','하다', '다', '로' |

분석기 중 하나인 Okt를 이용하여 문장의 최소 단위인 형태소로 분리하였다. 형태소로 분리한 후에는 불용어를 제거해주는 과정이 필요하다. 불용어(Stopwords)는 분석에 큰 의미가 없는 단어로, 주로 조사나 접미사가 불용어에 포함된다. 따라서 형태소로 토큰화한 뒤 Python의 list형태로 저장할 때 불용어 리스트에 포함된 단어를 제외하여 list에 저장해준다. 제외된 불용어 리스트는Table 3과 같다.

### *2.2.3. Vectorization*

다음으로는 형태소로 나뉘어진 문장을 벡터화(Vectorization)하는 과정이 필요하다. 텍스트 데이터를 임베딩하는 방법에는 word2vec, Glove, FastText, BERT 등 여러 방법이 있다. Mikolov (2013) 등이 제안한 Word2vec은 continuous bag of words (CBOW)와 Skip-Gram 2가지의 방식을 통해 단어의 빈도수를 기반으로 자주 같이 등장하는 단어들의 정보를 고려한 임베딩 방법이다. 또한, Pennington (2014) 등이 제안한 Glove에서는 단어 벡터의 내적으로 코사인 유사도를 사용한 Word2vec과 달리 단어 벡터의 내적으로 동시 등장 확률을 사용했다. Facebook에서 개발한 FastText는 하나의 단어 안에 여러 단어가 존재할 수 있음을 고려하여 내부 단어 학습이 가능하고, BERT는 2018년 구글에서 공개한 사전 훈련 모델로 그 성능이 우수한 것으로 알려져 있다. 하지만 본 연구에서는 가장 간단한 방법으로 keras의 preprocessing 모듈을 활용해 index로 벡터화를 진행했다. 이 때 각 단어에 word index, 즉 단어마다 고유의 숫자가 부여되는데 문서에 나오는 빈도가 높은 단어부터 오름차순으로 숫자가 부여된다. 따라서 생성된 word index의 일부분을 선택하여 분석에 사용할 단어의 개수

Table 4: Word Index

| | |
|---|---|
| word index (단어집합) | [‘영화’:1, ‘보다’:2, ‘있다’:3, ‘이다’:4, ‘좋다’:5,‘너무’:6, ‘없다’:7, ‘재밌다’:8, ‘연기’:9, ‘되다’:10, ‘같다’:11,‘적’:12,‘만’:13, ‘진짜’:14, ‘않다’:15, ‘스토리’:16, ‘배우’:17, · · · ] |

Table 5: Decision of Vocabulary size

| | |
|---|---|
| 전체 단어 집합의 크기 | 83,378 |
| 등장 빈도가 1회인 희귀 단어의 수 | 37,133 |
| 단어 집합에서 희귀 단어의 비율 | 44.5% |
| 분석에 활용할 단어 집합의 크기 | 46,245 |

Table 6: An example input and output from preprocessing

| | |
|---|---|
| Raw | 캐스팅 진짜 좋네요. 새로운 배우들의 조합도 좋았습니다. |
| Tokenization | [‘캐스팅’, ‘진짜’, ‘좋다’, ‘새롭다’, ‘배우’, ‘조합’, ‘좋다’] |
| Vectorization | [701, 15, 6, 476, 22, 1459, 6] |
| Padding | [0, 0,· · · , , 0, 701, 15, 6, 476, 22, 1459, 6] |

를 지정할 수 있다. 본 연구에서는 모든 단어를 모델에 넣는 것보다 최소 2회 이상 등장하는 단어만 사용하는 것이 유의미하다고 판단하여 1회만 등장하는 단어는 제거하여 단어 집합을 구성하였다. 1회 등장 단어는 전체 단어 집합의 44.5%로, 전체 83,378개 중 1회 등장하는 단어 37,133개를 제외하기로 했다. 그 결과 전체 단어 집합에서 1회 등장 단어를 제외한 크기인 46,245개를 분석에 활용하고자 했다. 생성된 단어 집합과 단어 집합 크기 결정 과정은 다음의 Table 4와 Table 5에서 확인할 수 있다.

#### *2.2.4. Padding*

마지막으로 각 리뷰의 길이를 일정하게 맞추는 작업을 수행하였다. 이 때 리뷰의 길이는 토큰의 수를 의미한다. 2.2.3의 Vectorization과정을 통해 문장은 각 단어와 매칭된 고유의 숫자로 표현된다. 이 때 문장마다 길이가 다르기 때문에 한 문장을 벡터의 형태로 고려하면 모델에 적용되는 벡터의 길이가 각각 다르게 된다. RNN 모델에 적용하려면 입력 데이터의 크기가 일정해야하기 때문에 지나치게 긴 문장은 자르고, 짧은 문장은 빈 부분을 0으로 채우는 과정이 필요하다. 동일한 길이로 맞춰주기 위해 토큰의 개수 분포를 참고하였는데 토큰 개수에 따라 총 리뷰에서 커버하는 비중을 아래 Figure 6 과 같이 확인한 후 약 98%를 포함할 수 있는 길이 80으로 지정하여 사용했다. Padding 작업이 끝나면 모형에 적용할 수 있는 데이터 형태가 된다.

전처리부터 Tokenization, Vectorization, Padding 과정을 거치면 원래의 데이터인 문장은 Table 6과 같은 과정으로 변형 되는 것을 확인할 수 있다.

## 3. Modeling

모델링을 하기 위해 sklearn의 model selection 모듈을 이용해 train data와 test data로 나누었다. 이 때 train data와 test data의 비율은 3:1로, 각각 86만(855,773)개의 75%, 25%만큼 나누어 훈련용 리뷰 수는 약 64만(641,829)개, 테스트용 리뷰 수는 약 21만(213,944)개로 split했다. 준비된 train data와 test data를 2장에서 서술한 자연어 처리 과정을 통해 모델 입력 형태로 만들어 주었다. 3장 modeling에서는 2-Class와 10-Class의 Classification과 Regression, 그룹화된 4-Class Classification, 그리고 두 번의 단계로 분석하는 2-step 방법까지 수행한 후

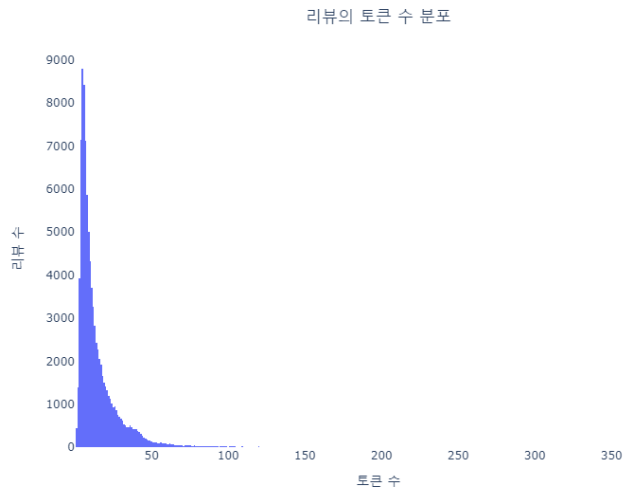



Figure 6: *Distribution of the amount of Tokens.*

모델들을 비교하여 영화 리뷰의 의미를 분석하고자 한다.

## 3.1. 2-Class Classification

### *3.1.1. 1점/10점 Classification*

우선적으로 평점이 1점, 10점인 리뷰 만을 가지고 데이터가 잘 분류되는지 2-Class Classification을 수행하였다. 그 중에서도 기존의 머신러닝 분류 모형과 RNN을 이용한 모형의 분류 결과를 비교 분석해 그 성능을 비교했다. 머신러닝 분류모형으로는 Random Forest, XGBoost, Naïve Bayes 모델, RNN모형으로는 LSTM, Bi-LSTM, Conv1d-LSTM 모형을 사용하였다. 머신러닝 분류 모형과 RNN 모형에 들어가는 데이터셋은 차이가 있다. 머신러닝 분류 모형의 경우 Figure 7의 (a)와 같이 문서에 등장하는 단어들의 빈도를 가지고 count vectorize하여 가장 빈도가 높은 단어 500개를 이용해 word list를 구성하였다. 각 리뷰에서 word list에 있는 단어가 등장하면 1, 그렇지 않으면 0을 부여하는 방법으로 훈련 데이터를 만들어 모델에 적합 시켰다. 반면 RNN 모형에 들어가는 데이터 셋은 자연어 처리 과정을 거친 결과로, Figure 7의 (b)와 같은 형태이다.

XGBoost는 grid search cross validation을 이용해서 최적의 모수를 사용했고, Random Forest는 조율모수에 따른 성능 차이가 크지않아서 디폴트 모형을 사용했다. LSTM model에서는 단어를 벡터로 표현하기 위해 모델에 Embedding과정을 포함하는데, 본 연구에서는 embedding vector의 차원을 100으로 설정하였다. 따라서 Embedding layer에 단어 집합의 크기와 설정한 embedding vector의 차원을 입력해준다. 해당 분류 모형은 평점이 1점과 10점인 리뷰를 분류하는 모형으로 2-Class Classification이기 때문에 마지막 출력 층에서 뉴런의 수는 1을, 활성화 함수로는 sigmoid를 사용한다. 추후 Class의 크기를 크게 할 때에는 활성화 함수를 softmax로 변경해 모델에 적용해 주었다. LSTM model은 layer를 쌓아서 모형을 더 정교하고 복잡하게 만들 수 있다. 따라서 LSTM layer만 단층으로 있는 간단한 모형과 두 층의 layer와 과적합 방지를 위해 Dropout을 적용한 모형을 비교하였다. 그 결과 정확도의 차이가 미미하고 분석에 소요되는 시간을 고려했을 때 간단한 모델을 사용해도 성능에 크게 영향이 없을 것이라고 판단했다. 따라서 이후에 언급하는 LSTM모형은 가장 간단한 형태의 LSTM모형을 기본으로 한다.

| | 영화 | 보다 | 을 | 이다 | 너무 | 있다 | 좋다 | 재밌다 | 없다 | 진짜 | ... | 애니메이션 | 반 | 주 | 기다 | 지키다 | 자기 | 이란 | 편 | 줄 | 대 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 500 columns

(a) machine learning model

```
array([[    0,     0,     0, ...,   532,   105,    51],
       [    0,     0,     0, ...,    53,   292,     2],
       [    0,     0,     0, ...,  1918,   695,   307],
       ...,
       [    0,     0,     0, ...,  2871, 15160,   276],
       [    0,     0,     0, ...,    33,   168,     9],
       [    0,     0,     0, ...,    82,    51,    83]], dtype=int32)
```

(b) RNN model

Figure 7: *Input data of two different models.*

Table 7: Accuracy comparison between machine learning and RNN model

| Model | Machine learning model | | | RNN model | | |
|---|---|---|---|---|---|---|
| | Random Forest | XGBoost | Naive Bayes | LSTM | Bi-LSTM | Conv1d-LSTM |
| 정확도(%) | 70.7% | 74.3% | 74.6% | 92.3% | 92.2% | 91.9% |

머신러닝 분류 모형과 RNN모형을 사용했을 때 정확도는 Table 7과 같다. 머신러닝에서는 XGBoost와 Naïve Bayes 모형의 정확도가 약 74%로 가장 좋았고, RNN 모형에서는 세 모형의 정확도가 약 92%로 성능이 안정적으로 좋았다. 또한 세 모형 모두 머신러닝 모형을 사용했을 때 보다 정확도가 높음을 확인할 수 있다. 따라서 본 연구에서는 RNN모형을 중심으로 분석을 수행할 것이다. 그리고 Bi-LSTM이나 Conv1d-LSTM 모형은 LSTM 모형에 비교했을 때 모델이 복잡한 것에 비해 정확도에 큰 차이가 없었다. 이 것은 연구 데이터가 짧은 문장이 많기 때문이라고 판단하여 이후 Multi-Class Classification 문제에서도 LSTM 모델만 사용하기로 했다.

## 3.2. Multi-Class Classification

이번에는 Multi-Class Classification 분석을 수행하고자 한다. 평점이 1점부터 10점까지 있기 때문에 우선 10-Class Classification을 시도하고, 점차 Class수를 줄여가며 최적의 Classification 모델을 구축하고자 했다.

### *3.2.1. 10-Class Classification*

Multi-Class Classification 문제에서는 10-class classification, regression, 그리고 regression과 classification을 결합한 3가지 형태의 모델을 적용하고 비교하였다.

1) 10-Class Classification model

   10-Class Classification은 2-Class 문제에서 사용한 LSTM모델을 동일하게 사용했다. 다만 마지막에 Dense에는 Class수만큼 입력해주고, multi-Class case이기 때문에 활성화함수는 'softmax'를 사용하였다. 분류 결과 10-class classification의 정확도는 57.1%로 산정되었다. 이 때 전체 자료에서 평점 10점의 비중이 51%인 것과 분류 모델로 예측된 평점의 62%가 10점인 것을 고려하면 정확도가 유의미하게 높다고 판단하기는 어려웠다.

2) Regression 다음으로는 regression을 이용하여 분석하고자 했다. 평점에서 1점은 부정적인 의미를, 10점

Table 8: Accuracy comparison of 4-Class case

| Model | 4-Class | Regression+ |
|---|---|---|
| | Classification | Classification |
| 정확도(%) | 65.4% | 49.8% |

은 긍정에 가까운 의미를 나타내는 등간척도이기 때문에 회귀 분석이 가능하다고 판단하여 수행했지만, regression 결과 RMSE값은 2.17이었다. 이 때 평점의 범위가 1부터 10이라는 점을 고려하면 성능이 좋다고 판단하기 어려웠다.

### *3.2.2. 4-Class Classification*

앞서 class가 10개일 때 1점과 10점에서는 잘 예측하는 반면 나머지 평점에서는 예측이 잘 되지 않았다. 따라서 이번에는 Class수를 줄여 4-Class를 중심으로 앞의 과정을 반복 수행하였다.

1) 4-Class Classification

10개의 Class를 4개로 그룹화하기 위해 50% 비중을 차지하고 있는 10점을 제외한 나머지 평점들을 3점씩 나누었다. 1-3점, 4-6점, 7-9점, 10점으로 그룹화했을 때 비중을 살펴보니 각각 20%, 10%, 20%, 50%로 고르게 분포되었다고 판단하여 4-Class로 4-Class로 그룹화하여 모델에 적합한 결과 분류 정확도는 65.4%였다.

2) Regression + Classification

Regression과 Classification에서의 정확도를 비교하기 위해 앞의 Regression결과를 그룹화하여 Classification 모형처럼 만드는 방법을 사용하였다. 앞서 10-Class Regression결과를 아래와 같이 구간을 나누어 4-class classification case로 바꾼 뒤 정확도를 산정한 결과 정확도는 49.8%였다.

$$\begin{cases} y = 0 & \text{if} \quad \hat{y} < 0.5 \\ y = 1 & \text{if} \quad 0.5 \le \hat{y} < 1.5 \\ y = 2 & \text{if} \quad 1.5 \le \hat{y} < 2.5 \\ y = 3 & \text{if} \quad \hat{y} \ge 2.5 \end{cases} \tag{3.1}$$

1), 2)의 두 가지 분석의 정확도를 비교하기위한 Table 8을 보면 4-Class Classification보다 Regression과 Classification을 결합한 형태의 모형의 정확도가 더 낮았다. 따라서 Regression을 이용한 모형보다 Multi-Class Classification 모형을 사용하는 것이 더 낫다고 판단하여 이 문제에 대해서는 4-Class Classification 모형을 최종 모형으로 선택했다.

### *3.2.3. 3-Class Classification*

4-Class Classification에서 양 극단인 1-3점, 10점은 비교적 잘 분류하는 반면, 중간 클래스인4-6점과 7-9점에서는 분류가 거의 되지 않았다. 따라서 이번에는 class수를 하나 더 줄여 3-Class로 분석을 수행했다. 평점의 비중을 고려하여 이번에는 1-4점(23%), 5-9점(27%), 10점(50%)으로 그룹화하였다. 분류 결과 정확도는 70.1%로 4-Class보다 성능이 개선되었다.

Table 9: Confusion Matrix of 2-step Classification model

| 평점 | | Predicted | | | Sum |
|---|---|---|---|---|---|
| | | 1-4점 | 5-9점 | 10점 | |
| Real | 1-4점 | 19.3% | 1.4% | 2.3% | 22.9% |
| | 5-9점 | 4.4% | 11.3% | 11.5% | 27.2% |
| | 10점 | 3.1% | 2.2% | 44.6% | 49.9% |
| Sum | | 26.7% | 14.9% | 58.4% | 100% |

Table 10: Accuracy Comparison among various classification models

| Classification model | 정확도 |
|---|---|
| 2-Class | 91.0% |
| 3-Class (2-step) | 75.2% |
| 3-Class | 70.1% |
| 4-Class | 62.8% |
| 10-Class | 34.4% |

### 3.3. 2-Step model

이번에는 두 단계에 거쳐 Classification을 수행하는 2-Step 방법을 이용하였다. 앞서 가장 극 단의 두 값인 1점과 10점을 분류하는 모델에서는 정확도가 약 92%였고, multi-Class Classification에서도 1점과 10점에서 분류가 잘 되는 것을 확인하였다. 그 중에서도 10점을 정 분류하는 경우가 많았기 때문에 첫 번째 단계에서 10점과 나머지인 1-9점을 분류하고, 그 다음 두 번째 단계에서 1-9점을 분류하는 2-Step model을 고안하였다. 2-Step model은 Step 1에서 1-9점, 10점을 분류하는 2-Class Classification을 수행한 다음 평점을 1-9점으로 예측한 데이터들만을 가지고 step2에서 다시 3-Class(1-4점, 5-9점, 10점) Classification을 수행하여 분류, 예측하는 모델이다. 따라서 최종적으로 3-Class로 분류가 되기 때문에 앞서 수행한 3-Class Classification 모델과 비교가 가능하다. 2-Step model을 적용한 결과 정확도는 75.2%가 나왔다. 3-Class model의 정확도가 70.1%라는 점을 고려했을 때 2-Step model이 더 성능이 좋은 모형이라고 판단할 수 있었다. 또한 Table 9의 오분류표를 통해 앞서 4-class case보다 중간 클래스가 잘 분류되었음을 확인할 수 있다.
3장에서 우리는 다수의 모형 비교를 통해 Class수가 늘어날수록 정확도가 낮아짐을 확인했다. 해당 결과는 Table 10에서 확인할 수 있다.

## **4.** 오분류의 원인

4장에서는 3장의 modeling에서 오 분류된 데이터를 통해 예측 모델의 한계점에 대해 예시를 들어 자세히 서술하고자 한다. 3장의 결과를 이용해 가장 정확도가 높았던 2-step model에서 잘못 분류한 리뷰들을 일부 추출하여 연구자 본인이 직접 평점을 보지 않고 예측해보았다. 그 결과 정확도는 약 47%로, 사람이 직접 리뷰를 읽고 평점을 예측하는 것 또한 쉽지 않았다. 따라서 4장에서는 리뷰 예시를 통해 오 분류의 원인을 크게 3가지로 나누어 설명할 것이다.

### 4.1. 리뷰만으로 평점 예측이 어려울 때

텍스트 분석은 문장에 포함된 단어를 모델에 학습시키고 평점에 따라 단어와 문장의 긍/부정을 파악하고 의미를 분석한다. 그런데 이 때 리뷰가 긍/부정을 파악할 수 없는 단어로 구성되는 경우 평점 예측을 어려운 것을

Table 11: Examples of the first cause of misclassification

| 모델 예측 | 실제 평점 | 리뷰 |
|---|---|---|
| 1-4 | 10 | 할말을 잃게 만드는 영화 |
| 1-4 | 10 | 소스케가 포뇨에 대한 사랑을 시험하는 관문을 통과하면서 죽음의 문턱에서 마을사람들을 구해낸거 아닌가 |
| 10 | 2 | 그냥 다큐멘토리같아요 |
| 10 | 1 | ㅋㅋㅋ아ㅋㅌㅌ개웃기네 |
| 1-4 | 10 | 이영화를 한마디로 표현 하자면 뚜띠뚜띠 |
| 1-4 | 10 | 내 현실은 이보다 더 똥이다 |

Table 12: Examples of the second cause of misclassification

| 모델 예측 | 실제 평점 | 리뷰 |
|---|---|---|
| 10 | 1-4 | 만약 이 영화를 스포해주는 친구가 있다면 그 친구는 당신의 평생 친구입니다 |
| 10 | 1-4 | 중간에 화장실 편하게 갈수 있음 |
| 10 | 1-4 | 영화가 너무 재미있어서 보다가 잠듬 |

Table 13: Examples of the third cause of misclassification

| 모델 예측 | 실제 평점 | 리뷰 |
|---|---|---|
| 1-4 | 8 | 미안합니다. 보다 잤습니다 |
| 1-4 | 8 | 다 좋은데 그 엿같은 일본의 역겨운 감성. 누가봐도 배드엔딩인데 해피엔딩듯출연진 전원이 억지웃음을 짓고 있다.나랑은 안맞다 |
| 1-4 | 10 | ㅋㅋ나만볼수없지 |
| 10 | 1 | 진심 개재밌어요 |
| 10 | 1 | 볼만해요 ㅎㅎ두번 보세요 |

확인했다. Table 11과 같이 감정 관련 단어가 아닌 영화의 전반적인 스토리 위주의 리뷰이거나 중의적 표현, 혹은 리뷰를 보고도 영화에 대한 감상을 알 수 없을 경우가 그러하다.

## 4.2. 풍자 표현

최근 네이버 영화 리뷰를 보면 비꼬는 말투나 반어적 표현을 사용한 리뷰들이 있다. 이런 경우 예측 모델은 리뷰의 숨은 의미를 파악하지 못하고 잘못 예측하게 되는데, 이런 부분에서는 모델이 잘못 예측할 확률이 높다. 그 예시들은 Table 12와 같다.

## 4.3. Label mismatch

리뷰를 보면 명백하게 부정 적인 감상 임에도 평점이 높게 입력 되어있거나 그 반대의 경우가 있다. 이런 경우를 label mismatch라고 정의하였다. Table 13의 예시는 연구 데이터에서 발견한 label mismatch 사례로, 이러한 데이터가 모델의 학습 데이터로 들어가게 되면 모델이 잘못 학습할 수 있고, 테스트 데이터로 들어가면 오분류된 데이터가 될 것이다. 따라서 모델의 성능을 저하하는 요소가 된다.

## 5. 결론

지금까지 한국어 자연어 처리와 머신러닝과 딥러닝을 이용해 네이버 영화 리뷰 데이터의 평점 예측을 통한 텍스트의 의미 분석을 수행하였다. 3장에서는 가장 극 단의 두 평점을 분류하는 2-Class Classification부터 10-Class, 그리고 평점들을 그룹화 하여 분류한 4-Class와 3-Class Classification 분석 후 정확도를 가지고 각 모델의 성능을 비교하였다. 2-Class 문제에서는 머신러닝과 딥러닝을 비교함으로써 가장 좋은 성능을 보인 LSTM 모형을 선택했고, multi-Class 문제에서는 Classification과 Regression을 결합한 형태보다 단순 Classification의 정확도가 높은 것을 확인했다. 또한 1점과 10점의 2-Class Classification 정확도가 좋았던 결과를 이용해 1-9점과 10점을 우선적으로 분류하고 1-9점으로 예측한 데이터를 가지고 한 번 더 분류 분석을 수행하는 2-step 방법을 고안해냈다. 다수의 2, 3, 4, 10 Class 문제에서 정확도를 비교해본 결과 Class 수가 늘어날수록 정확도가 줄어드는 것을 확인할 수 있었다. 4장에서는 실제 리뷰를 가지고 이러한 오분류의 원인에 대해 서술하였다. 리뷰만으로 평점 예측이 어려운 경우, 비꼬는 듯한 풍자 표현이 포함된 리뷰의 경우, 그리고 평점이 잘못 라벨링된 label mismatch가 그러한 경우이며 특히 Label mismatch인 데이터가 훈련 데이터에 있을 경우 모델 성능을 저하하는 요소가 될 수 있음을 설명했다.

오 분류의 원인 중 첫 번째와 두 번째의 경우 임베딩 방법의 한계로, BERT, ELMo, GPT-2 등 contextual embedding 관련 방법을 활용하면 개선의 여지가 있을 것이다. 본 연구에 사용한 데이터가 한 줄 평의 짧은 문장이라는 한계도 있다. 문장이 짧을수록 가치 판단에 대한 부분도 적은 부분을 차지하기 때문에 짧은 문장의 한계를 고려하면 양 극단의 평점은 잘 예측 가능하지만 그 사이의 평점에서 미묘한 차이를 분류하기는 쉽지 않다. 또한 짧은 문장에서는 모델을 다양화하기 어렵다는 한계점이 있다. 선행 연구에서 CNN과 RNN을 결합한 모델의 성능이 가장 좋았음에도 불구하고 본 연구 데이터에서는 CNN을 적용하기에 문장의 길이가 충분히 길지 않아서 성능의 차이가 크지 않았다. 따라서 만약 데이터가 좀 더 풍부했다면 더 다양한 모델의 적용이 가능했을 것이다. 그럼에도 본 연구에서의 최종 모형을 활용하여 mismatch 데이터를 검거에 활용하는 등 모델을 정교화 하는 것에 기여할 수 있을 것이다. 더 나아가 본 연구 데이터에 고객 식별 번호 데이터를 결합시킨 형태의 자료가 있다면 고객 개인이 남긴 리뷰와 평점을 통해 개인의 영화 취향을 학습함으로써 새로운 작품을 추천하는 개인화 추천 시스템에까지 활용할 수도 있을 것이다.

## 6. Acknowledgement

본 연구에 자료를 제공해주신 네이버와 오브젠에 무한한 감사를 드립니다. 본 연구는 두 회사에서 제공한 데이터없이는 불가능했을 것입니다. 다시 한 번 감사의 말씀을 드립니다.

## References

Kharde V and Sonawane S (2016). Sentiment analysis of Twitter data: A survey of techniques, *International Journal of Computer Applications*, **139**, 5–15.

Lee JJ, Kwon SB, and Ahn SM (2018). Sentiment Analysis Using Deep Learning Model based on Phoneme-level Korean, *Journal of Information Technology Services*, **17**, 79–89.

Mikolov T, Chen K, Corrado GS, and Dean J (2013). Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*.

Nayak A (2016). Comparative study of Naive Bayes, *Support Vector Machine and Random Forest Classifiers in Sentiment Analysis of Twitter feeds*.

Oh Y, Kim M, and Kim W (2019). Korean Movie-review Sentiment Analysis Using Parallel Stacked Bidirectional LSTM Model, *Journal of KIISE*, **46**, 45–49.

Park H and Kim K (2019). Sentiment Analysis of Movie Review Using Integrated CNN-LSTM Model, *Intelligence and Information Systems*, **25**, 141-–154.

Parmar H, Bhanderi S, and Shah G (2014). Sentiment mining of movie reviews using random forest with tuned hyperparameters, *International Conference on Information Science*.

Pennington J, Socher R, and Manning C (2014). GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543.

Rehman AU, Malik AK, and Raza B (2019). A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis, *Multimedia Tools and Applications*, **78**, 26597-–26613.

*Received August 24, 2021; Revised October 18, 2021; Accepted October 28, 2021*

# 네이버 영화 리뷰 데이터를 이용한 의미 분석(semantic analysis)

김소진[a], 송종우[1,a]

[a]이화여자대학교 통계학과

요 약

SNS의 등장으로 인터넷 이용자들이 온라인에 남기는 텍스트의 양이 방대해지고 그 중요성이 강조되고있다. 특히 네이버의 영화 탭에서 볼 수 있는 영화 평점이나 리뷰는 실제로 관객들이 영화를 보기 전 해당 영화를 볼 것인지 결정하는 데 주요 요인이 되기도 한다. 본 연구는 실제 네이버 영화 리뷰 데이터를 가지고 평점을 예측하는 분석을 수행했다. 영화 리뷰 데이터를 분석하기 위해 평점의 분포를 통해 데이터 특성을 살펴보았고, 텍스트의 의미를 분석하기 위해 형태소 분석을 통한 한국어 자연어처리를 수행했다. 또한 평점 예측에 활용할 모델 선택을 위해 2-Class와 multi-Class 문제들에 대해 머신러닝과 딥러닝, 회귀와 분류 분석을 비교했으며, 오분류의 원인을 영화 리뷰 데이터 특성과 연관시켜 서술했다.

주요용어: 의미분석, NLP, LSTM, RNN, 영화 리뷰

[1]교신저자: (03760) 서울특별시 서대문구 이화여대길 52, 이화여자대학교 통계학과. E-mail: josong@ewha.ac.kr