감성 분류를 위한 워드 임베딩 성능 비교

윤혜진*, 구자환*, 김응모*
*성균관대학교 소프트웨어대학
yhjsoar5@gmail.com, jhkoo@skku.edu, ukim@skku.edu

Performance Comparison of Word Embeddings for Sentiment Classification

Hye-Jin Yoon*, Jahwan Koo*, Ung-Mo Kim*
*College of Software, Sungkyunkwan University

요 약

텍스트를 자연어 처리를 위한 모델에 적용할 수 있게 언어적인 특성을 반영해서 단어를 수치화하는 방법 중 단어를 벡터로 표현하여 나타내는 워드 임베딩은 컴퓨터가 인간의 언어를 이해하고 분석 가능한 언어 모델의 필수 요소가 되었다. Word2vec 등 다양한 워드 임베딩 기법이 제안되었고 자연어를 처리할 때에 감성 분류는 중요한 요소이지만 다양한 임베딩 기법에 따른 감성 분류 모델에 대한 성능 비교 연구는 여전히 부족한 실정이다. 본 논문에서는 Emotion-stimulus 데이터를 활용하여 7가지의 감성과 2가지의 감성을 5가지의 임베딩 기법과 3종류의 분류 모델로 감성 분류 학습을 진행하였다. 감성 분류를 위해 Logistic Regression, Decision Tree, Random Forest 모델 등과 같은 보편적으로 많이 사용하는 머신러닝 분류 모델을 사용하였으며, 각각의 결과를 훈련 정확도와 테스트 정확도로 비교하였다. 실험 결과, 7가지 감성 분류 및 2가지 감성 분류 모두 사전훈련된 Word2vec가 대체적으로 우수한 정확도 성능을 보였다.

1. 서론

자연어는 일상생활에서 사용하는 언어를 말하고, 자연어 처리는 자연어를 컴퓨터가 처리하도록 하는 것을 말한다. 자연어를 처리하기 위해서는 사람의 언어를 컴퓨터가 이해하고 처리할 수 있도록 단어를 0과 1로 이루어진 형태로 바꾸어야 하는데, 이것을 벡터화 혹은 워드 임베딩이라 한다.

워드 임베딩은 상대적 유사성이 의미적 유사성과 상관관계가 있는 벡터를 말한다. Neural Probabilistic Language Model[1]에서 Neural Network를 사용하여 단어 벡터 생성을 시도하였고, Efficient Estimation of Word Representation in Vector Space[2]에서 Word2vec 기법을 소개하였으며, GloVe: Global Vectors for Word Representation[3]에서 Word2vec 기법에 카운트 기반의 방법을 가미한 GloVe기법을 소개하였다.

Sentiment Analysis using Word2vec-CNN-BiLSTM Classification[4]에서는 Word2vec, CNN, BiLSTM 모델을 활용하여 텍스트 데이터를 긍정, 부정, 중립의 감성으로 분류하였고, Sentiment Analysis and Classification using Convolutional Neural Networks[5] 또한 Word2vec 및 CNN 모델을 활용하여 텍스트 데이터를 긍정, 부정, 중립의 감성으로 분류하였다. 이와 같이 워드 임베딩을 활용하여 감성 분류를 시도한 여러 연구들이 있지만, 긍정, 부정, 중립의 3가지 감성 혹은 긍정, 부정의 2가지 감성으로 분류한 연구가 대다수다.

본 논문에서는 다양한 임베딩을 통해 문장의 감성 분류를 시도하였다. CountVectorizer, TfidfVectorizer, Word2vec의 CBOW(Continuous Bag-of-Words Model), Word2vec의 Skip-gram(Continuous Skip-gram Model), Pretrained-Word2vec 등 총 5가지 방법으로 문장을 벡터화하였고, Logistic Regression, Decision Tree, Random Forest 등 총 3가지 방법으로 벡터화한 문장을 감성에 따라 분류하는 학습을 진행하였다. 테스트 수행 시 평가 지표로 정확도(Accuracy)를 활용하였고, 주요 하이퍼 파라미터 및 단어 임베딩의 합산하는 식을 변경하며 실험하였다. 본 논문은 이를 통해 어떤 임베딩 기법이 감성을 분류하는 데에 효과적인지 비교하는 것이 목표이다.

본 논문의 구성은 다음과 같다. 2장에서는 자연어 처리와 임베딩, 감성분류와 관련된 기존 연구를 소개하고 3장에서는 감성 분류를 위한 워드 임베딩 성능 비교 실험을, 4장에서는 성능 평가를, 5장에서

는 결론을 기술한다.

2. 관련 연구

2.1 자연어 처리

자연어란 일상생활에서 사용하는 언어를 말한다. 자연어는 1945년, ENIAC의 탄생과 함께 기호, 문자 계열의 언어라는 뜻과 함께 등장하였다. 자연어 처리는 자연어를 컴퓨터가 처리하도록 하는 것을 말한다. 등장 당시의 자연어 처리는 기계번역의 의미를 가지고 있었으며 1970년부터 자연어의 의미가 확장되기 시작하며 자연어 처리의 의미 역시 현재의 의미로 점차 확장되었다. 자연어 처리의 현황과 전망[6]에 자연어 처리는 두뇌 기관에서 일어나는 인간이 서로 의사소통하기 위한 기호의 이해와 산출 과정의 작동을 컴퓨터가 모사하도록 하는 기술이라 하며, 방대한 코퍼스 자료로부터 귀납된 규칙들을 적용하고 새로운 규칙 체계를 학습하는 과정이라 기술한다. Recent Trends in Deep Learning Based Natural Language Processing[7]에서는 자연어 처리를 인간 언어 분석과 표현을 자동화하기 위한 계산 기법이라고 표현하기도 한다.

2.2 워드 임베딩

자연어를 처리하기 위해서는 사람이 사용하는 언어를 컴퓨터가 이해하고 처리할 수 있도록 형태를 바꾸어야 한다. 컴퓨터가 이해할 수 있도록 단어를 0과 1로 이루어진 형태로 바꾸는 것을 벡터화 혹은 워드 임베딩이라고 한다. 자연어를 벡터화하는 방법에는 One-hot Encoding, Bag of Words, N-gram, Countvectorizer, Tfidfvectorizer, Word2vec 등이 있다. Bag of words, N-gram, Countvectorizer, Tfidf vectorizer는 문장에서 혹은 코퍼스에서 단어가 등장하는 빈도에 기반한 방법이다. Efficient Estimation of Word Representation in Vector Space[2]에서는 Word2vec을 소개하며 이를 사용하여 단어를 벡터화하면 [King - Man + Woman = Queen]과 같은 연산을 수행할 수 있다. Word2vec을 수행하는 방법에는 CBOW, Skip-gram 두 가지 방법이 있는데 CBOW는 주변 단어로 중심 단어를 예측하는 방법이고 Skip-gram은 중심 단어로 주변 단어를 예측하는 방법이다.

Pretrained Embedding(사전훈련된 임베딩)은 사전에 학습된 워드 임베딩을 가져와서 사용하는 것으로, 본 논문에서는 Google news 코퍼스로 학습된 3백만 개의 300 차원 영어 단어 벡터 데이터를 활용하였다. Pretrained Embedding을 사용하면 단어의 임베딩 시간이 따로 들지 않고, 대량의 데이터로 학습된 임베딩 데이터이기 때문에 벡터가 단어의 일반적인 특성을 잘 나타낸다는 장점이 있다. 그러나 분석하고자 하는 데이터의 특징을 담고 있지 않다는 단점을 가지고 있다.

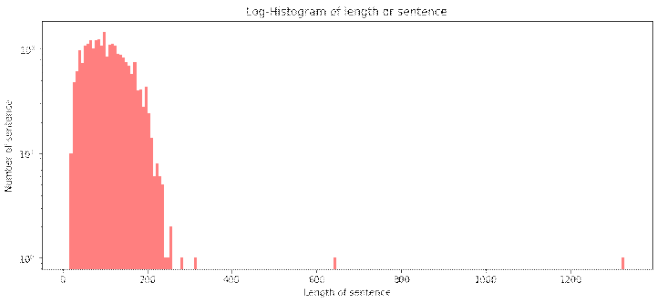
2.3 감성 분류

감성 분류란 문장, 문단, 단어 등에 대하여 기쁨, 슬픔, 수치, 고독, 분노 등의 감성 종류로 분류하는 것이다. Sentiment Analysis using Word2vec-CNN-BiLSTM Classification[4]에서는 Word2vec, CNN, BiLSTM 모델을 활용하여 텍스트 데이터를 긍정, 부정, 중립의 감성으로 분류하였고, Sentiment Analysis and Classification using Convolutional Neural Networks[5] 또한 Word2vec 및 CNN 모델을 활용하여 텍스트 데이터를 긍정, 부정, 중립의 감성으로 분류하였다. BERT를 활용한 속성기반 감성분석[8]에서는 BERT 모델로 텍스트 데이터를 긍정, 부정, 중립으로 분류하였다.

3. 모델 구현 및 다각도 실험

3.1 실험 개요

기존 연구에서는 대부분 긍정, 부정, 중립의 3가지 감성 혹은 긍정, 부정의 2가지 감성으로 분류 시도하였다. 따라서 7가지의 다양한 감성으로 분류를 시도함과 더불어 각 임베딩 및 분류 모델 사이의 정확도 차이를 통해 어떤 모델이 감성 분류에 적합한지 알아보려 한다.



<그림 1> 탐색적 데이터 분석 - 문장의 길이 분석

<그림 2> 탐색적 데이터 분석 - 단어 빈도수 분석

감성 분류를 위한 워드 임베딩 모델의 구현은 Google Colabolatory 환경에서 수행되었다. 본 연구는 데이터 수집 및 정제, 임베딩, 분류 후 평가의 순서로 진행된다. 첫 번째로, 연구의 목적에 맞는 데이터를 수집한다. 이후, 연구를 수행하기 알맞은 형태로 데이터를 정제한다. 정제한 데이터로 임베딩, 분류 모델을 학습하는데 CountVectorizer, TfidfVectorizer, CBOW, Skip-gram, Pretrained Word2vec의 임베딩 순서로 수행하였다. 임베딩하여 데이터를 벡터화한 후, Logistic Regression, Random Forest, Decision Tree 모델로 분류 학습을 진행하였다. 마지막으로 이 모델의 성능을 테스트용 데이터로 평가하였다.

3.2 데이터

Ghazi, Inkpen, Szpakowicz(2015)의 Emotion-stimulus 데이터를 활용한다.[9] 이 데이터는 'happy', 'sad', 'anger', 'fear', 'surprise', 'disgust', 'shame'의 총 7가지의 감성과, 그 감성을 드러내는 총 2414개의 문장으로 구성되었으며, 820개의 감성의 원인이 되는 부분을 포함한 데이터와 1594개의 원인을 포함하지 않는 데이터로 이루어져 있다. 테스트용 데이터는 단어 임베딩을 통한 고전 문학 감성 분석[5]에서 등장한 소설 속 문장과, 그 문장의 감성 데이터를 활용한다.

<그림 1>은 문장의 길이를 분석하여 그래프로 나타낸 것이고, <그림 2>는 워드클라우드 기법을 통해 단어의 등장 빈도를 시각화한 것이다. 데이터 속 문장의 길이는 평균 106.8798, 최대 1327의 길이를 가지고 있다. 가장 많이 등장한 단어는 'said', 'face' 이었고 그 다음을 'felt', 'happy', 'embarrassed'와 같은 감성을 나타내는 단어가 뒤를 이었다.

데이터는 다음과 같은 과정으로 정제하였다. 첫째로 영어가 아닌 느낌표, 마침표 등의 특수 문자를 제거했다. 다음으로 대문자를 소문자로 바꾸어 주었다. the, a, is와 같은 stopwords를 제거하고 마지막으로 단어들 사이에 공백을 첨가하여 하나의 문장으로 바꾸었다.

3.3 실험 진행

총 5가지의 임베딩 기법과 3 종류의 모델로 분류 학습을 진행하였다. CountVectorizer, TfidfVectorizer, CBOW 방법을 사용한 Word2vec, Skip-gram 방법을 사용한 Word2vec, 그리고 사전에 학습된 Word2vec 임베딩을 사용하여 데이터의 문장들의 벡터화하였다. 이렇게 임베딩된 문장들을 각각 Logistic Regression, Decision Tree, Random Forest 모델로 감성 분류 학습을 진행하였다.

CountVectorizer, TfidfVectorizer의 경우 sklearn 라이브러리를 사용하였다. Word2vec은 gensim 라이브러리를 사용하였는데 파라미터가 다양하고, 그에 따라 결과가 많이 달라질 수 있기 때문에 워드 벡터의 차원을 100에서 500, 윈도우의 크기를 1에서 10 등 파라미터의 값을 바꾸어가며 학습을 진행하였다. Pretrained Word2vec은 구글뉴스 데이터로 사전 훈련된 데이터를 사용하였다.

이렇게 임베딩한 데이터를 CountVectorizer, TfidfVectorizer는 그대로, Word2vec은 문장을 구성하는 단어 벡터의 평균을 학습 데이터로 하는 분류 학습을 진행한다. 감성 분류를 위해 Logistic Regression, Decision Tree, Random Forest 모델 등과 같은 보편적으로 많이 사용하는 분류 모델을 사용하였다.

<표 1> 임베딩 기법에 따른 분류 모델의 정확도

워드 임베딩 기법	분류 모델	Accuracy1		Accuracy2	
		Validation1	Test1	Validation2	Test2
CountVectorizer	Logistic Regression	0.96	0.37	0.97	**0.68**
	Decision Tree	**0.97**	0.21	**0.98**	**0.68**
	Random Forest	**0.97**	**0.47**	0.97	**0.68**
TfidfVectorizer	Logistic Regression	**0.91**	**0.32**	**0.91**	0.57
	Decision Tree	0.73	0.26	0.90	**0.68**
	Random Forest	0.87	**0.32**	0.89	**0.68**
Word2vec (CBOW)	Logistic Regression	0.68	0.29	**0.86**	**0.71**
	Decision Tree	0.39	0.36	0.68	0.64
	Random Forest	**0.78**	**0.43**	083	0.64
Word2vec (Skip-gram)	Logistic Regression	**0.76**	0.29	**0.85**	**0.64**
	Decision Tree	0.36	0.36	0.72	0.50
	Random Forest	0.69	**0.50**	0.83	**0.64**
Pretrained Word2vec	Logistic Regression	0.66	0.74	**0.87**	0.68
	Decision Tree	0.40	0.42	0.67	**0.73**
	Random Forest	**0.82**	**0.63**	0.80	0.63

<표 2> Pretrained Word2vec의 Random Forest 결과 중 일부분

sentence	real	expect
uh what is this, it is stinky	disgust	anger
My next step was to look for the beast which had been the cause of so much wretchedness; for I had, at length, firmly resolved to put it to death.	fear	sad

4. 성능 평가

5개의 임베딩 기법과, 3개의 분류학습을 진행한 결과, <표 1>의 Accuracy1(7가지 감성으로 분류)과 Accuracy2(2가지 감성으로 분류)에서 Validation(훈련 정확도), Test(테스트 정확도)의 정확도를 보인다.

첫째, Decision Tree로 분류했을 때의 성능은 대체로 좋지 않다. Countvectorizer와 Tfidfvectorizer의 훈련 정확도를 제외하면 평균 34.5의 정확도를 가질 만큼 임베딩을 통한 감성 분류에는 Decision Tree 모델은 효과적이지 않음을 알 수 있다.

<표 2>은 Pretrained Word2vec를 통해 Random Forest로 분류한 모델의 훈련 데이터 중 2개의 문장을 따온 것이다. 비록 훈련 정확도에 비하여 테스트 정확도가 많이 낮지만 표 1과 같이 구체적인 감성은 틀렸으나 긍정적인 감성과 부정적인 감성으로 분류하면 두 문장 모두 실제 감성, 예측 감성 모두 부정적인 감성으로 분류되는 케이스가 많아, 'disgust', 'anger', 'fear', 'sad'는 부정적인 감성으로 나머지는 긍정적인 감성으로 분류하도록 하여 정확도를 계산해 보았다. 그 결과는 <표 1>의 Test2 와 같다.

7가지의 감성 분류에서 2가지의 감성 분류로 축소할 경우 <표 1>의 Test1에서 Test2로, 정확도가 향상되었음을 알 수 있다. 이를 통해 임베딩을 통한 감성 분류는 긍정, 부정과 같이 간단한 분류에는 쓸만하지만, 7가지 감성과 같은 구체적이고 다양한 분류에는 적합하지 않음을 알 수 있다.

이 외에도 특이한 결과를 보이는 것이 바로 CountVectorizer와 TfidfVectorizer의 정확도이다. TfidfVectorizer는 CountVectorizer의 단점을 보완한 기법으로 알려져 있다. 하지만 본 논문에서는 한 가지의 경우를 제외한 모든 경우의 정확도가 TfidfVectorizer보다 CountVectorizer가 높은 것을 확인할 수 있다. 이는 데이터셋도 감성과 관련된 데이터셋이고, 자주 등장하는 단어는 감성과 연관이 있는 단어라는 특수한 상황이기 때문이다. 때문에 TfidfVectorizer에서는 감성과 연관된 단어의 영향력이 작도록 설정하였을 것이고 Countvectorizer보다 성능이 안 좋게 나온 것으로 추정된다.

5. 결론

본 논문에서는 다양한 임베딩 기법들을 통하여 문장의 감성 분류를 진행하였다. 총 5가지의 임베딩 기법을 통해 단어 및 문장의 벡터화를 진행하였고 문장 벡터를 분류 모델에 학습시켜 감성 분류를 시도하였다. 훈련 정확도와 테스트 정확도의 차이에서, Word2vec에 비하여 tfidf, count기반 임베딩이 훈련 정확도에 비하여 테스트 정확도가 많이 떨어졌다. 데이터 셋의 크기에 비하여 감성 라벨의 종류가 많아 정확도가 낮게 나오고, 양질의 학습 데이터셋이 충분히 반영되지 않아 정확도가 일관성을 보이지 않는 등 결과가 미흡하다. 따라서 향후 연구에서 양질의 데이터셋을 학습하고 테스트하여 더 의미있는 결과를 낼 수 있도록 수행할 예정이다.

참고문헌

[1] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, "Neural Probabilistic Language Model", NIPS, 2000

[2] Thomas Mikolov, Kai Chen, Greg Corrado, Jeffey Dean, "Efficient Estimation of Word Representations in Vector Space", ICLR, 2013

[3] Jeffrey Penning, Richard Socher, Christopher D.Manning, "GloVe: Global Vectors for Word Representation", EMNLP, 2014

[4] Wang Yue, Lei Li, "Sentiment Analysis using Word2vec-CNN-BILSTM Classification", IEEE, 2020

[5] 김한나, "합성곱 신경망을 사용한 감성 분석 및 분류", 한국교육학술정보원, 2020

[6] 고창수, "자연어 처리의 현황과 전망", 우리말학회, 2012

[7] Young, T., Hazarika, D., Poria, S., & Cambria, E. "Recent Trends in Deep Learning Based Natural Language Processing", IEEE Computational Intelligence Magazine, 2018

[8] 박현정, 신경식, "BERT를 활용한 속성기반 감성분석: 속성카테고리 감성분류 모델 개발", 한국지능정보시스템학회, 2020

[9] Diman Ghazi, Diana Inkpen, Stan Szpakowicz, "Detecting Emotion Stimuli in Emotion-Bearing Sentences", Lecture Notes in Computer Science, 2015