[소프트웨어학과]

SentencePiece*를 이용한 유튜브 댓글의 감정분류 문제 성능 개선 연구

[결과보고서]

Version 1.0

2020. 6. 23.

학번: 201723825

이름: 임윤진

지도교수: 손경아 교수님

목 차

<표 목차>

<그래프 목차>

<그림 목차>

요약

The Purpose of this study is to improve the performance of the sentiment classification by effectively embedding YouTube comments with slang, abbreviations and grammatical errors, even in situations where fast pre-processing is required or the size of the word dictionary is limited. As a solution to this problem, the idea of applying the SentencePiece is presented instead of Keras Tokenizer.

As a result, it was confirmed that the performance of tokenization and embedding with SentencePiece can achieve up to 75.77% performance in the LSTM binary classification model. In addition, when the model was repeatedly trained, the performance of the SentencePiece was still better than that of the Keras Tokenizer, and the results were stable. In the end, SentencePiece can effectively solve the Out Of Vocabulary problem by reducing the number of unknown missing tokens in sentences.

1 개요

1.1 연구 배경

유튜브(YouTube)는 세계 최대 규모의 동영상 공유 플랫폼으로서 창작자(크리에이터)가 콘텐츠를 공유하면 댓글 등의 형식으로 논의되는 것이 특징이다. 즉, 유튜브에서 창작자와 사용자는 텍스트를 매개로 상호 행위적 관계에 놓인다.[1] 특히 창작자의 입장에서 사용자의 반응을 파악하는 것은 수익원인 콘텐츠 생성에 영향을 미치기 때문에 댓글을 통한 사용자와의 적극적인 소통 양상을 보인다. 이러한 배경을 바탕으로 유튜브 댓글의 감정 분석은 창작자들에게 유의미한 작업으로 간주된다.

1.2 연구 목표

따라서 본 연구에서는 빠른 전처리 작업이 필요하고 단어 사전의 크기가 제한된 상황에서도 댓글 데이터를 효과적으로 벡터화해서 감정 분류기의 성능을 높이는 방향으로 정했다. 특히 수많은 단어 변칙이 존재하는 유튜브 댓글 데이터에서 단어 사전에는 없는 단어를 처리하는 Out Of Vocabulary(OOV) 문제를 잘 해결해보고자 했다.

2 관련 선행 연구 조사

2.1 선행 연구

댓글을 이용한 감정 분석 중에서도 텍스트를 긍정 또는 부정으로 분류하는 문제는 다양한 해결방법으로 성능이 개선되어왔다. 한편, 은어나 줄임말, 문법적 오류가 다수 존재하는 댓글의 특성을 고려한 연구는 그 특성에서 비롯된 문제점을 보완하기 위해 복잡한 텍스트 분류 작업을 요구했다. 트위터(Twitter)를 이용하여 감정 분류를 진행한 선행 연구[2]에 따르면, 댓글을 전처리하는 과정에서 연구자가 지정한 규칙에 따라 숫자 및 특수문자를 제거하거나 철자를 교정하고, 반복되는 문자를 찾아 없애는 작업 등을 거쳐야 했다. 또한, 여러 대용량 단어 사전을 사용하여 수록된 단어마다 감정 점수를 부여하는 방식을 제시했다.

2.1 선행 연구와의 차이점

표 1 과 같이, 필자는 앞서 설명한 텍스트 분류 작업을 개선할 점으로 보고 작업속도를 최소화할 수 있는 전처리 방식, 대용량 단어사전을 도입하지 않아도 최대한 모든 단어를 숫자로 표현할 수 있는 방식, 문장의 특성을 고려한 딥러닝 학습으로 감정 분류를 하는 방식을 제안하고자 한다.

<표 1>

구분	선행 연구*	개선 방향
전처리 작업	중복 단어 제거, 불용어 제거, 철자 교정, 숫자 및 특수문자 제거, 지정 문자 교체 등.	작업 속도를 고려하여 이모티콘과 불용어 제거를 제외한 전처리 작업을 최소화.
단어 사전 생성	대용량 단어 사전을 사용. Slang 단어 추출 및 번역 과정을 거침.	단어 사전의 크기를 줄여도 Out Of Vocabulary 문제를 효과적으로 처리할 수 있음.
감정 분류	모든 단어의 Sementic Orientation(SO)를 측정하고 문장의 감정 점수를 계산.	Sequential Data 의 특성을 고려한 Recurrent neural network 를 이용.

3 연구 결과

3.1 문제 정의

본 연구에서는 유튜브 댓글의 감정 분석을 위해 단어 사전의 생성 및 문장 토큰화(Tokenization)와 벡터화(Vectorization) 과정에 초점을 맞추어 댓글 전처리 작업을 최소화하고 대용량 단어 사전에 의존하지 않아도 데이터 내의 단어를 최대한 수치화할 수 있는 방안을 제안하고자 한다. 이에 파이썬(Python)에서 텍스트 토큰화와 벡터화를 위해 사용되는 라이브러리인 Keras Tokenizer 와 번역 문제에서 사용되는 SentencePiece[3] 라이브러리를 이용하여 Recurrent Neural Network 기반 감정 분류 모델의 성능을 비교한다. 필자는 SentencePiece 라이브러리를 사용하는 것이 Keras Tokenizer 라이브러리를 사용하는 것보다 더 좋은 결과를 도출할 것이라고 가정한다.

3.2 사용 데이터 (변경 과정 포함)

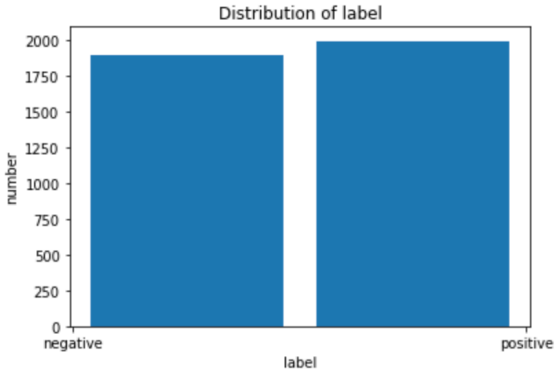
프로젝트 초반에는 'yelp' 리뷰 데이터 50 만개를 이용하여 분석을 진행하였으나, 그 결과 Tokenizer 라이브러리 기반 임베딩(성능 0.9468)이 SentencePiece 라이브러리 기반 워드 임베딩(성능 0.9455)보다 약간 좋은 학습 성능을 낸다는 것을 확인했다. 이는 yelp 데이터 문장 내에 사용된 단어들이 완성체에 가까워서 OOV 를 처리하는 데 큰 차이가 없었기 때문인 것으로 판단했다. 따라서 은어나 줄임말, 문법적 오류가 다수 존재하는 유튜브 댓글로 데이터를 변경하여 다시 분석해보기로 하였다.
사용 데이터는 2019 년에 음식 / 영화 / 뷰티 / 제품 리뷰 / 교육 카테고리 별 영상의 댓글을 웹 크롤링 방식으로 추출하고, 긍정, 부정을 각각 0, 1 로 라벨링한 것이다. 각 영상의 출처 링크는 표 3 과 같다.

<표 2>

음식	cereal	https://www.youtube.com/watch?v=DHFGf-yq5Ng&t=12s
	ramsay	https://www.youtube.com/watch?v=ZJy1ajvMU1k
	burger	https://www.youtube.com/watch?v=wgOhw4OABeE
영화	aladin	https://www.youtube.com/watch?v=6VCwsZn3cZ4
	us	https://www.youtube.com/watch?v=MRcfoMAiklQ
	spy	https://www.youtube.com/watch?v=8paUJtSrea4
	joker	https://www.youtube.com/watch?v=1gzdlfCp2Iw
뷰티	waterprf	https://youtu.be/9FltyjuGvGk
	routine	https://youtu.be/1L9EA7J2pXg
	16	https://youtu.be/g3rmTxOiuBc
	eyeliner	https://youtu.be/Q7ZMIUOqhHE
	winnie	https://www.youtube.com/watch?v=gmxhy4aXOSQ
리뷰	Iphone11	https://www.youtube.com/watch?v=DyX-QZZBgpw
	rewind	https://www.youtube.com/watch?v=YbJOTdZBX1g&t=20s
	dogbot	https://www.youtube.com/watch?v=8t8fyiiQVZ0
	book	https://www.youtube.com/watch?v=sbPjVsfnS_c&t=19s
교육	github	https://www.youtube.com/watch?v=Loav1kbA640
	wrongdijk	https://www.youtube.com/watch?v=qx9sJ3O3JM0
	Codepro	https://www.youtube.com/watch?v=GBuHSRDGZBY
	marriage	https://www.youtube.com/watch?v=M3NQE0BaAQw&feature=youtu.be

그 결과 총 3884 개의 댓글 데이터를 확보했고, 그래프 1 과 같이 각 라벨에 포함된 데이터의 개수를 파악했을 때 부정 데이터는 1892 개, 긍정 데이터는 1992 개로 나타났다.

<그래프 1>



그 다음 댓글에서 이모티콘과 불용어만 제거하는 간단한 전처리 방식을 거쳤다. 댓글의 평균 길이와 사용 단어 수를 계산한 결과, 총 6016 개의 단어가 존재했고, 단어 빈도수를 나열했을 때 희귀 단어를 제외하고 전체의 약 70%를 차지하는 단어 4300 개를 단어 사전의 크기로 지정했다.

3.3 제안하는 기법

표 2 와 같이 Keras Tokenizer 는 전처리된 텍스트 데이터를 입력 데이터로 사용하고, 단어를 등장 빈도수 기준으로 나열하여 인덱스를 부여한다. 또한, 문장 토큰화 시 띄어쓰기 단위로 단어를 분리하기 때문에 문장을 복원할 때 원본과 다를 수 있다. 아래 예시를 보면, 원문에서는 'world'와 '!' 사이가 붙어 있지만 복원 후 문장은 띄어쓰기가 된 채로 출력되는 것을 알 수 있다.

- 원문: Hello world!!
- 토큰화된 문장: [Hello] [world] [!] [!]
- 복원 후 문장: Hello world ! !

반면, SentencePiece 는 전처리를 하지 않은 데이터를 입력해도 byte-pair-encoding 이나 unigram language model 등의 분리 알고리즘에 따라 서로 다른 토큰화 결과를 출력할 수 있다. 그리고 띄어쓰기를 '_'로 변환하기 때문에 아래 예시처럼 토큰화 후에도 손실없이 문장을 복원할 수 있는 특징을 가지고 있다. 깃허브에 설명한 내용[4]에 따르면, 기본 파라미터가 unigram 으로 설정되어 있고, BPE 와의 토큰화 결과를 비교했을 때 경험적 차이를 느끼지 못했다고 한다.

- 원문: hello world
- 토큰화된 문장: [_] [h] [e] [ll] [o] [_w] [o] [r] [l] [d]
  [_] [he] [ll] [o] [_world]
- 복원 후 문장: hello world

<표 3>

구분	Keras Tokenizer	SentencePiece
일반적인 입력 데이터	전처리 작업한 데이터	전처리 하지 않은 raw 데이터
문장 토큰화	띄어쓰기 단위로 단어 분리. '.' 사이 띄어쓰기에 대한 정보를 담지 못함.	Byte-pair-encoding, unigram language model 등의 분리 알고리즘을 지원.
문장 복원	문장을 완벽하게 복원하는 데 한계가 있음.	띄어쓰기를 '_'로 변환하여 손실 없이 원본 복원 가능. Decoder 지원.

아래 예시는 Keras Tokenizer, SentencePiece 두 가지 라이브러리를 이용해 유튜브 댓글 데이터 중 한 문장을 토큰화한 결과이다. Tokenizer 와 달리 SentencePiece 의 토큰화 방식을 보면, 한 단어 내에서도 '_goo', 'ood'처럼 여러 구간으로 쪼개지는 것을 확인할 수 있다. 이러한 특징을 이용하여 'gooooood'이라는 새로운 단어가 등장해도 '_goo', 'oo', 'ood'로 분리하여 숫자로 표현할 수 있기 때문에 단어 사전에 존재하지 않는 단어를 처리하는 Out Of Vocabulary 문제를 잘 해결할 수 있다고 생각했다.

Keras Tokenizer

- 원본 [actually reaaally gooood movie ! 9/10]
- 토큰화 [actually, <UKN>, < UKN>, movie, 9, 10]
- 벡터화 [43, 1, 1, 2, 630, 119]

SentencePiece unigram

- 원본 [actually reaaally gooood movie ! 9/10]
- 토큰화 [▁actually, ▁rea, a, ally, ▁goo, ood, ▁movie, ▁!, ▁9, /10]
- 벡터화 [74, 340, 79, 487, 2303, 548, 10, 6, 657, 627]

4 성능 분석

4.1 성능 분석 환경

서버 사양

Intell Corel i7-6850K CPU @ 3.60GHz – 12core, 1202 MHz

MEM : 32G

GeForce GTX 1080Ti 11G * 2

성능 분석 지표는 10-cross validation 을 적용하여 도출된 10 개의 validation set accuracy 의 평균과 표준편차로 정한다. 또한, validation set 마다 예측한 댓글 데이터의 라벨과 텍스트를 따로 저장할 수 있도록 했다.

4.2 학습 모델

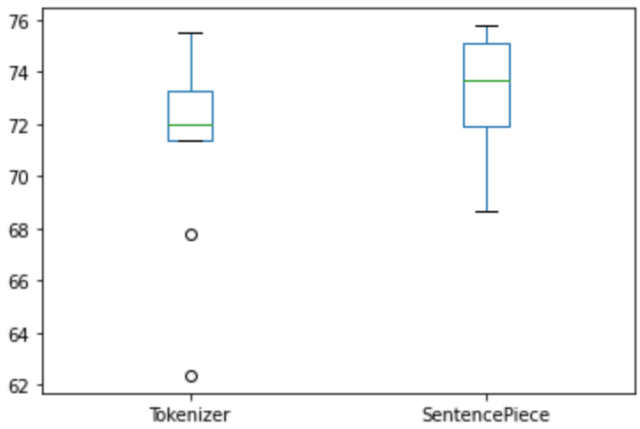
<그림 1>

```
Model: "sequential_29"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_29 (Embedding)     (None, None, 100)         430000
_________________________________________________________________
lstm_87 (LSTM)               (None, None, 100)         80400
_________________________________________________________________
lstm_88 (LSTM)               (None, None, 100)         80400
_________________________________________________________________
dropout_29 (Dropout)         (None, None, 100)         0
_________________________________________________________________
lstm_89 (LSTM)               (None, 100)               80400
_________________________________________________________________
dense_29 (Dense)             (None, 2)                 202
=================================================================
Total params: 671,402
Trainable params: 671,402
Non-trainable params: 0
```

파이썬 tensorflow 를 이용하여 생성한 학습 모델은 그림 1 과 같이 총 3 층의 LSTM 을 쌓고 current dropout 과 dropout 을 각각 0.4, 0.5 로 설정했다. 예측은 0 또는 1 의 이진 분류 방식이기 때문에 binary crossentropy loss function 을 사용했다. Optimizer 는 Adam 이다. 서로 다른 두 가지 방식으로 생성한 텍스트 벡터 셋을 입력해보고 약 67 만개의 파라미터를 학습하여 이진 분류기의 성능을 비교했다.

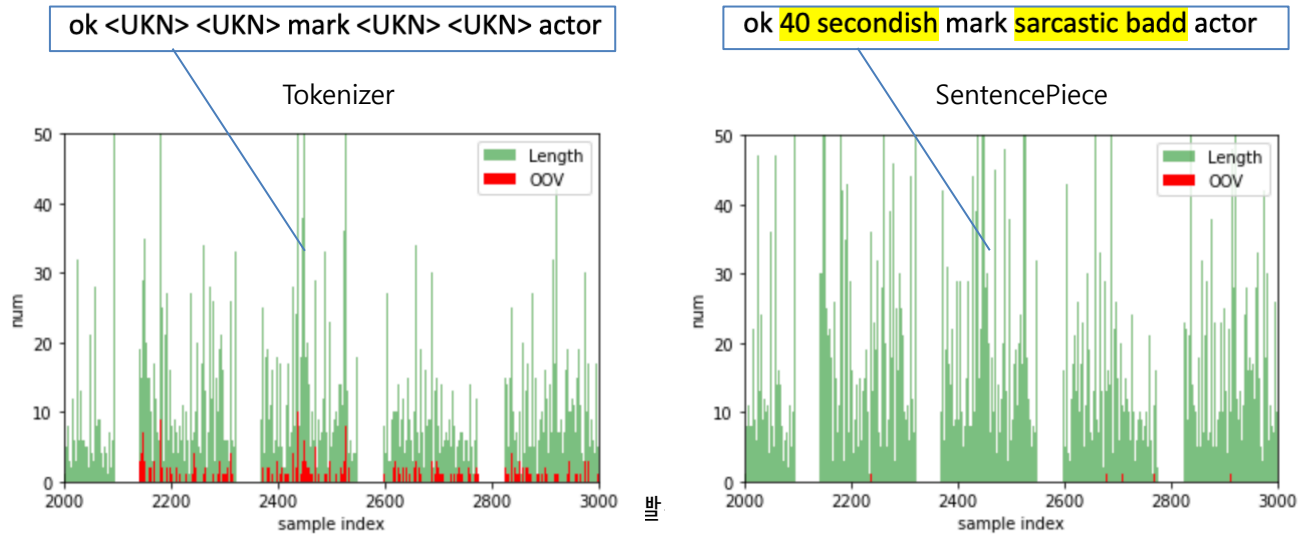
4.3 성능 분석 결과

<그래프 2>



10 개의 validation set accuracy 의 평균과 표준편차를 구한 결과, Keras Tokenizer 의 경우 평균 71.39%, 표준편차는 +-3.65% 이었고, SentencePiece 의 경우 평균 73.30%, 표준편차는 +-2.21% 이었다. 즉, 똑같이 단어 사전의 개수를 4300 개로 제한한 상황에서 SentencePiece 가 Tokenizer 보다 성능이 좀 더 좋고, 표준편차가 더 작은 것을 확인할 수 있었다. 또한, 그래프 2 의 box plot 을 살펴보았을 때 Tokenizer 의 성능에는 68%, 62% 등의 이상치가 존재하는 것을 알 수 있었다.

<그래프 3>



Tokenizer 를 사용할 때의 Out Of Vocabulary 개수가 더 많이 분포하는 것을 알 수 있었다. 따라서 SentencePiece 가 유튜브 댓글 데이터의 OOV 처리에 효과적인 것을 확인할 수 있었다.

<표 4>

S: sentencepiece, T: Tokenizer

index	Method	text	predict
176	S	disliked video ? ! guys crazy ? ! good !	1
	T	disliked video guys crazy good	0
314	S	's right gouda wins ! best cheese , ca n't change mind	1
	T	's right gouda <UKN> best cheese ca n't change mind	0
320	S	hey ! ! guys burger topping showdown . best cheese . see pairs well cheese .	1
	T	hey guys burger <UKN> <UKN> best cheese see <UKN> well cheese	0
321	S	cook burgers old cast iron skillet	1
	T	cook burgers old cast iron <UKN>	0
333	S	everybody talks smith like something prove ? ? ? loved movie bottomline	1
	T	everybody talks smith like something prove loved movie <UKN>	0
347	S	owowowow , smiled :)	1
	T	<UKN> <UKN>	0
358	S	100 % agree ! great outline work original director ' take advantage .	1
	T	100 agree great <UKN> work original director ' take <UKN>	0
378	S	yeah agree christ character jaffar & wierd magic stick suit real life movie change . prefect , watch 3 times ...	1

	T	yeah agree christ character jaffar wierd magic stick <UKN> real life movie change <UKN> watch 3 times	0
383	S	love shirts 3 videos ofcource hehe	1
	T	love shirts 3 videos <UKN> <UKN>	0

특히 4th validation set 은 모델을 반복적으로 학습시켰을 때 SentencePiece 쪽에서 지속적으로 높은 성능을 보여주었기 때문에 4th validation set 을 중심으로 텍스트 데이터의 특징을 조사해 보았다. 이때 Sentencepiece 는 예측이 맞았는데 Tokenizer 에서 틀린 경우가 총 106 개였다. 106 개의 텍스트를 비교해본 결과를 표 4 에 정리하고 이 중 SentencePiece 에서 결측값이 발생하지 않은 단어에 대해 하이라이트 표시를 해주었다. 표에 대한 분석 결과, 다음과 같은 특징을 가지고 있었다.

a. 물음표, 느낌표, 온점 등과 같은 특수문자를 data 로 사용한다. '☺'와 같은 감정 표현을 포함할 수 있다.

b. 단어 사전의 크기 제한으로 포함되지 못했던 단어를 표현할 수 있다. 따라서 <UKN> 토큰 대신 'wins', 'owowowow', 'smiled', 'advantage', 'prefect', 'suit', 'hehe' 등의 단어를 포함할 수 있다.

c. 두 텍스트 간의 차이가 없음에도 예측 결과가 다른 경우가 많이 존재하였다.

d. 0 보다 1 을 못 맞추는 경우가 많았다.

다음 표 5 은 영화 카테고리의 영상 댓글 2780 개를 뽑아서 다시 모델 학습한 결과를 분석한 것이다. 영화 리뷰 분야에서도 SentencePiece 가 평균 70.14, 표준편차 +-2.30%의 성능을 보였고, Tokenizer 는 평균 67.77%, 표준편차 +-2.86%의 성능을 보였다. 특히 8th validation set 은 SentencePiece 쪽에서 Tokenizer 보다 7.19% 더 높은 성능을 보여주었기 때문에 8th validation set 을 중심으로 텍스트 데이터의 특징을 조사해 보았다.

<표 5>

S: sentencepiece, T: Tokenizer

index	method	text	predict
25	S	' shocked ' bring " climax. " easily biggest letdown , personally	1
	T	' shocked ' bring " <UKN> " easily biggest letdown personally	0
35	S	saw movie last night pretty much took words right mouth . thing different would say though jasmines new song felt place stylistaclly . sounded way modern .	1
	T	saw movie last night pretty much took words right mouth thing different would say though <UKN> new song felt place <UKN> sounded way modern	0
46	S	actually reaaally gooood movie ! 9/10 ? ? ? ?	1
	T	actually <UKN> <UKN> movie 9 10	0
63	S	loved movie aside jafar ' casting portayal . weak acting addition	0

		almost resemblance look-wise . mediocre representation important character almost singe handedly downgraded movie . , really enjoyed	
	T	loved movie aside jafar ' casting <UKN> weak acting addition almost <UKN> look <UKN> mediocre representation important character almost <UKN> <UKN> <UKN> movie really enjoyed movie nostalgia	1
99	S	ok 40 secondish mark sarcastic badd actor	0
	T	ok <UKN> <UKN> mark <UKN> <UKN> actor	1
161	S	(hears jeremy give opening synopsis film) ... get smart (film) ? comedy spy agency field agents identitiies ousted send desk jockey get people ens	1
	T	<UKN> jeremy give opening <UKN> film get smart film comedy spy agency field <UKN> <UKN> <UKN> send <UKN> <UKN> get people <UKN> <UKN> moments comedy bits times genuinely funny literally <UKN> get smart sounds	0
169	S	4:37 c-c-c-c-combo breaker ! ! ! !	1
	T	4 <UKN> c c c c <UKN> <UKN>	0
170	S	wanted stand throw roses jeremy finished opera end video	1
	T	wanted stand throw <UKN> jeremy finished <UKN> end video	0
228	S	a- bahahahaha . movie hot garbage . cuz black cast doesnt mean need bend backwards chris . movie deserved hilariocity review	1
	T	a <UKN> movie hot garbage cuz black cast doesnt mean need <UKN> <UKN> chris movie deserved <UKN> review	0
258	S	great film , really enjoyed , lot unexplained feel needs explaining . come first place ? made like clones start , exist place things , marry people , kids etc. ? also , many other	
	T	great film really enjoyed lot <UKN> feel needs explaining come first place made like clones start exist place things <UKN> people kids etc also many cothers showed hell lot beach many one person <UKN> <UKN> <UKN> attack one ever find literally living <UKN> however many years one found heard lupita	

8^{8h} validation set 에서 Sentencepiece 는 예측이 맞았는데 Tokenizer 에서 틀린 경우가 총 40 개였다. 40 개의 텍스트를 비교해본 결과, 다음과 같은 특징을 가지고 있었다.

a. 비정형적인 단어를 표현할 수 있다. 예를 들어 'reaaally', 'gooood', 'badd', 'secondish', 'bahahahaha'등의 단어를 포함할 수 있다.

b. 영화와 관련된 단어를 더 세부적으로 포함하기도 한다. 'climax', 'opera', 'synopsis', 'sarcastic', 'hilariocity' 등.

c. index 258 예시처럼 Tokenizer 에서 쓰인 단어가 더 많아도 SentencePiece 가 더 정확하게 맞춘다는 것을 알 수 있다.

5 결론

본 연구의 목적은 빠른 전처리 작업이 필요하거나 단어 사전의 크기가 제한된 상황에서도 은어나 줄임말, 문법적 오류가 다수 존재하는 유튜브 댓글을 효과적으로 임베딩하여 감정 분류기의 성능을 높이는 것이었다. 또한, 단어 사전에 없는 단어를 처리하는 Out Of Vocabulary(OOV) 문제를 잘 해결하는 방법을 찾아보고자 했다. 이를 해결하는 방안으로 문장 벡터화 과정에서 일반적으로 사용되는 파이썬 라이브러리인 Keras Tokenizer 대신에 Neural Machine Translation 에서 주로 사용되는 SentencePiece 를 적용하는 아이디어를 제시하였다. 그 결과 SentencePiece 로 토큰화 및 벡터화를 수행할 경우, LSTM 이진 분류 모델에서 최대 75.77%의 성능을 낼 수 있는 것을 확인했다. 또한, 모델의 학습을 반복적으로 수행했을 때, 여전히 SentencePiece 의 성능이 Keras Tokenizer 보다 좀 더 좋았고, 안정적인 결과를 내는 것을 확인할 수 있었다. 결국, SentencePiece 는 전처리 작업을 최소화할 수 있고 단어 사전의 크기가 제한적이어도 문장 내 결측값인 <UKN>토큰의 개수를 줄이기 때문에 OOV 문제를 효과적으로 해결한다는 사실을 알게 되었다.

이러한 연구 결과는 선행 연구에서 트위터 댓글을 사용한 이진 분류기의 성능인 약 91.54% 보다 낮은 성능을 보였지만, 유튜브 댓글 데이터를 새로 수집하여 분석해보았다는 점에서 차별점이 있다. 또한, 본 연구는 텍스트 분석의 필수적인 과정인 전처리와, 토큰화, 벡터화 방식에 대해서 다루었기 때문에 향후 다른 텍스트 분석에도 이 연구 결과를 활용할 수 있다는 점에서 가치가 있다. 본 연구를 발전시키기 위해서 Elmo 와 같은 학습 모델을 사용하여 성능을 높이거나, 영문이 아닌 국문 텍스트를 이용한 연구가 가능할 것으로 본다.

참고자료

[1] 주현식, "네트워크화되는 '나'(me) – 유튜브와 대화적 상상력, 문화와 융합", 제 41 권 5 호, 2019, pp. 689

[2] Fazal, M.K., Aurangzeb, K., Shakeel, A. & Muhammad, Z., "Lexicon-Based Sentiment Analysis in the Social Web", J. Basic. Appl. Sci. Res., pp. 238-248

[3] Kudo, T., & Richardson, J, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing", *arXiv preprint arXiv:1808.06226*.

[4]] Kudo, T., & Richardson, J, "*Sentencepiece python module*" (2020). Accessed: Apr.16, 2020. [Online]. Available: https://github.com/google/sentencepiece/blob/master/python/sentencepiece_python_module_example.ipynb