

[캡스톤디자인 결과보고서]

연구과제

과제명 (작품명)	Multi-category classification emotion analysis with Korean translated into English using NMT API	참여학기	2020년 1학기
--------------	---	------	-----------

강좌정보

과목명	데이터분석 캡스톤 디자인	학수번호	SWCON32100
과제기간	2020년 3월 16일 ~ 2020년 6월 26일	학점	3

팀구성

팀명	GamSung		팀구성 총인원	3명
구분	성명	학번	소속학과	학년
대표학생	엄세웅	2014102714	국제학과	4
참여학생	장서진	2017102832	프랑스어학과	4
	정희재	2017103757	소프트웨어융합학과	3

지도교수 확인

지도교수	성명	이대호	직급	전임교수
	소속학과	소프트웨어융합학과	지도교수 확인	성명 : 이대호 (인)

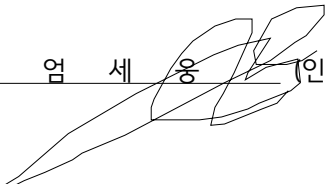
붙임

[첨부1] 과제 요약보고서  
[결과물] 최종결과물 (최종작품 사진/도면/발표자료 등)

본 팀은 과제를 성실히 수행하고 제반 의무를 이해하여 이에 따른 결과보고서를 제출합니다.

일자 : 2020년 6월 26일

신청자(또는 팀 대표) 엄세웅 (인)



[캡스톤디자인 과제 요약보고서]

과제명	Multi-category classification emotion analysis with Korean translated into English using NMT API														
1. 과제 개요	<p>가. 과제 선정 배경 및 필요성</p> <p>Single-polarity sentiment analysis는 문장의 감정을 ‘긍정/부정/중립’으로 나눠서 분석한다. 하지만 부정적인 감정에는 공포와 불안, 불안 등 여러 가지 감정이 있으며 긍정적인 감정에도 여러 가지 감정들이 있다. Ekman은 자신의 논문 “An Argument for Basic Emotions: Cognitoin and Emotion”을 통해 감정을 7가지 카테고리인 “ANGER, DISGUST, FEAR, HAPPINESS, SADNESS, SURPRISE”로 나누었다. 영어의 경우 트위터, 댓글, 온라인 뉴스 데이터와 Ekman의 7가지 감정 분류를 활용한 multi-category classification emotion anaylsis에 대한 연구가 활발하다.</p> <p>하지만 한국어 감정 분석은 대부분 Single-polarity(긍정/부정)에 머물고 있다. 여러 가지 이유 중에서도 가장 큰 문제점은 한국어의 교착성이다. 한국어의 경우 영어와는 다르게 형태소별로 나눌 수 있는 교착어로 이루어져 있어 상대적으로 감정 사전을 구축하는 것이 어렵다. 감정 사전 구축이 어렵다보니 한국어의 감정 분석은 Multi-polarity로 나아가지 못하고 Single-polarity에 머물고 있다.</p> <p>이로 인하여 여러 카테고리의 감정 분석 결과를 여론조사에 활용하며 기존의 통계적 분석을 대체하고 있는 미국과는 다르게 한국은 한정적인 분야에서만 감정 분석 결과를 사용하고 있다. 따라서 본 연구는 한국어 문장의 감정 분석에 특화된 학습 모델을 찾고자 한다.</p> <p>나. 과제 주요내용</p> <p>본 연구에서는 한국어 문장을 세 가지 NMT API를 사용하여 한국어 문장을 영어로 번역한 뒤, multi-category classification emotion analysis를 통해 Ekman의 7가지 감정 카테고리로 분류하는 것을 목표로 한다. 이때 훈련 데이터는 이미 라벨이 붙어 있는 영어 문장 데이터를 사용하며, 테스트 데이터는 네이버 영화 댓글 데이터와 트위터에서 “총선”을 키워드로 한 크롤링 데이터를 사용한다. NMT API는 Google, Kakao, Papago NMT API를 사용한다. 이후 Multiplicative LSTM, Multi-class SVM Kernels, Transformer 모델을 사용하여 multi-category classification emotion analysis 모델을 구축하고 NMT-모델 조합의 precision 결과를 비교 분석한다.</p> <p>다. 최종결과물의 목표</p> <ul style="list-style-type: none"><li>- Train data: Kaggle labeled emotion dataset 13,948</li><li>- Test data: Naver Movie Data + Twitter Data with “election” keyword</li></ul> <p>위 테스트 데이터와 훈련 데이터를 사용하여 다음 9가지 NMT-모델 조합의 Precision을 비교 분석한다. 이때 각 NMT-모델의 성능은 Precision &gt; 0.5를 목표로 하며, 이 중 가장 성능이 좋은 NPT-모델 조합을 제시하는 것을 목표로 한다.</p> <table><tr><th>NMT API</th><th>Model</th></tr><tr><td rowspan="3">Kakao NMT API</td><td>Multi-class SVM kernels</td></tr><tr><td>Multiplicative LSTM</td></tr><tr><td>Transformer</td></tr><tr><td rowspan="3">Papago NMT API</td><td>Multi-class SVM kernels</td></tr><tr><td>Multiplicative LSTM</td></tr><tr><td>Transformer</td></tr><tr><td rowspan="3">Google NMT API</td><td>Multi-class SVM kernels</td></tr><tr><td>Multiplicative LSTM</td></tr><tr><td>Transformer</td></tr></table> <p>표 1. NMT API - 모델 조합</p>	NMT API	Model	Kakao NMT API	Multi-class SVM kernels	Multiplicative LSTM	Transformer	Papago NMT API	Multi-class SVM kernels	Multiplicative LSTM	Transformer	Google NMT API	Multi-class SVM kernels	Multiplicative LSTM	Transformer
NMT API	Model														
Kakao NMT API	Multi-class SVM kernels														
	Multiplicative LSTM														
	Transformer														
Papago NMT API	Multi-class SVM kernels														
	Multiplicative LSTM														
	Transformer														
Google NMT API	Multi-class SVM kernels														
	Multiplicative LSTM														
	Transformer														

## 2. 과제 수행방법

팀 구성원이 3명이기 때문에 다음과 같이 역할을 분배하여 연구를 수행하였다. 결과 비교 분석과 같이 표에 없는 활동은 팀 구성원 모두가 함께 진행하였음을 밝힌다. 이후 각 구성원 별로 어떻게 과제를 수행했는지 나누어서 보고서를 기술하였다.

이름	Data	NMT API	Model
엄세웅	Twitter	Papago NMT API	Multiplicative LSTM
장서진	Movie	Google NMT API	Multi-class SVM kernels
정희재	Label Data	Kakao NMT API	Transformer

표 2. 팀 구성원 별 NMT API 및 모델 역할 분담

### 1) 엄세웅

#### ① 데이터 수집

모델의 테스트 데이터셋을 확보하기 위하여 트위터의 “총선”을 키워드로 검색한 데이터를 사용하였다. 해당 데이터는 영화 데이터에 비하여 상대적으로 길이가 길며 단위 문장 당 드러나는 감정의 수가 적은 특징을 가지고 있다. 예를 들어 영화 댓글은 “전반부는 좋았으나 후반부는 화가 날 정도로 별로였다”처럼 여러 가지 감정이 혼재해 있는 경우가 많지만 트위터 데이터는 상대적으로 감정이 통일되어 있었다.

트위터에서 공식으로 지원하는 Tweeter Crawling API “TWEETPY”가 있으나 최근 7일 데이터만 수집할 수 있는 한계가 있다. 7일 이전의 데이터 수집을 위해선 Premium-API를 구매해야 하는데 500request에 \$149/월 이다. 따라서 본 연구에는 트위터 공식 API대신 twitterscraper 패키지를 사용하였다.

#### ② NMT API

Papago NMT API는 Naver Developers에 올라와 있는 공식 API 사용 예제에 따라 구현하였다. 이후 결과를 JSON파일로 저장하여 입력 데이터로 사용하기 쉽게 저장하였다.

```
if(rescode==200):
    response_body = response.read()

    # Json format
    result = json.loads(response_body.decode('utf-8'))
    pprint(result)

    # Json result
    with open('Crawler\\translated_files\\translated_0401_0402.txt', 'w', encoding='utf8') as f:
        f.write(result['message']['result']['translatedText'])
else:
    print("Error Code:" + rescode)
```

그림 1. Papago NMT API 호출 코드 예제

#### ③ 데이터 전처리

데이터 전처리는 용도에 맞게 데이터를 사전에 변경하는 작업으로. 토큰화, 정규화, 불용어 처리 등 여러가지 자연어 처리 기법을 사용하여 데이터 전처리를 진행하였다.

##### (1) 데이터 정제 및 정규화

라벨링된 데이터는 트위터 데이터를 포함하고 있기 때문에 용도에 맞게 데이터를 정제 및 정규화 해야 한다. 트위터 아이디 및 링크는 분석 용도에 적합하지 않기 때문에 정제 작업을 통해 제거해야 하며, 영어의 표현 방식을 통일하기 위해 데이터 정규화 과정을 거쳐야한다.

- 데이터 정제: 하이퍼링크, 트위터링크, 트위터 아이디 제거
- 데이터 정규화: can't -> cannot, won't -> will not 등으로 표현 정규화

##### (2) 토큰화, 어근 추출 및 불용어 처리

본 연구에서는 nltk 패키지를 사용하여 토큰화, 불용어 처리, 어간 추출을 진행하였다. 먼저 어간 추출이란 어간(Stem)을 추출하는 작업으로 단어는 어간과 어미로 이루어져 있으며 어간에 단어의 의미

가 내포되어 있다. 따라서 어간을 추출하여 의미 있는 부분만을 추려내었다.

다음으로 불용어 처리는 문장에서 유의미한 단어를 선택하고 필요 없는 것들은 제거하는 작업을 의미한다. 예를 들어 문장에서 I, my, me는 자주 등장하는 단어이지만 문장에 자연어 처리 분석에 큰 도움이 되지 않는다. 이렇게 자주 등장하지만 분석에 도움이 되지 않는 단어들을 제거하는 작업을 불용어 처리라고 한다.

```
[[ 'im', 'feel', 'optimist', 'stash', 'reduc', 'abil', 'month', 'expect', 'realli', 'big', 'empti', 'post', 'next', 'time'], 'happiness']  
[[ 'feel', 'safe', 'berri'], 'happiness']  
[[ 'find', 'time', 'feel', 'respect', 'get', 'point', 'across', 'someth', 'may', 'make', 'feel', 'uncomfort', 'nice', 'seem', 'encourag', 'thing', 'keep', 'happen'], 'happiness']  
[[ 'good', 'cook', 'mind', 'u', 'feel', 'content', 'everytim', 'got', 'prepar', 'simpl', 'humbl', 'dish', 'eaten'], 'happiness']  
[[ 'feel', 'everyon', 'abil', 'artist', 'special', 'way', 'find', 'attract', 'art', 'unleash', 'fromth', 'virgin', 'artist'], 'happiness']  
[[ 'im', 'feel', 'rather', 'mellow', 'id', 'like', 'point', 'thing', 'dont', 'understand'], 'happiness']  
[[ 'search', 'search', 'search', 'rare', 'feel', 'satisfi', 'solut', 'found'], 'happiness']  
[[ 'mother', 'tremend', 'phone', 'talk', 'hour', 'good', 'mood'], 'happiness']  
[[ 'live', 'virginia', 'eight', 'whole', 'month', 'feel', 'super', 'weird'], 'happiness']  
[[ 'cant', 'wait', 'till', 'summer', 'feel', 'somewhat', 'carefre'], 'happiness']
```

## 그림 2. Cleaning, Normalization, Tokenization, Stemming, Stop word 결과

### (3) Vocabulary 구성

정수 인코딩을 사용하기 위해 토큰화된 단어를 가지고 Vocabulary를 구성한다. 여기서 Vocabulary란 단어, 복수와 같은 형태는 다르지만 의미는 같은 단어를 같은 단어로 묶어 주는 기법을 의미한다. 또한 단어의 빈도 순서대로 정렬하여 인코딩을 하기 위하여 nltk의 FrqDist 클래스를 사용했다.

```
think peopl littl would one work thing hate still even  
love bit back need see much say got look ive  
good come way someth today could start tri life dont  
right friend take pretti help alway also year use home  
new around cant someon week thought never last though talk  
amaz find made well hope u lot watch amp person  
bad didnt live felt fuck girl read sure weird everi  
quot strang happi miss actual ever night write left let
```

## 그림 3. Vocabulary 생성 결과

### (4) 정수 인코딩

컴퓨터는 기본적으로 텍스트 데이터 보다 숫자 데이터를 훨씬 더 빠르게 처리할 수 있다. 따라서 자연어 처리에서는 앞서 만든 Vocabulary를 정수로 변환였다. 인덱스를 부여하는 방법에는 여러 가지 방법이 있지만 앞서 Vocabulary를 빈도수로 정렬하였기 때문에 빈도수를 기준으로 인덱싱을 진행했다.

```
[[3, 1, 909, 3782, 3030, 1066, 213, 411, 8, 298, 630, 129, 119, 7], 0]  
[[1, 565, 3783], 0]  
[[63, 7, 1, 503, 4, 196, 720, 35, 143, 9, 1, 299, 244, 132, 1626, 18, 100, 105], 0]  
[[32, 987, 252, 67, 1, 451, 2254, 29, 845, 988, 2255, 1430, 2050], 0]  
[[1, 148, 1066, 610, 433, 34, 63, 1431, 944, 3784, 5321, 2586, 610], 0]  
[[3, 1, 197, 1114, 209, 2, 196, 18, 41, 214], 0]  
[[1116, 1116, 1116, 1117, 1, 754, 1627, 220], 0]  
[[275, 3785, 300, 61, 160, 32, 595], 0]  
[[74, 3786, 2051, 170, 213, 1, 422, 80], 0]  
[[54, 144, 611, 452, 1, 485, 1285], 0]
```

## 그림 4. 정수 인덱싱 생성 결과

## ④ Multiplicative LSTM 모델

### (1) 모델 선정 이유

Multiplicative LSTM은 자기 회귀 밀도 추정(autoressive density estimation)에 대해 더 표현력이 있다. 따라서 mLSTM은 평균적으로 LSTM보다 성능이 뛰어나며 deep variants for a range of character level language modelling 문제들에 적합하다. 하지만 mLSTM 모델 하나만으로는 놓치는 문맥들이 발생할 수 있고 단어와 단어 사이의 관계를 명확히 표현하는데 부족함이 있다. 그래서 출력 시퀀스의 정확도를 보정해주면서 각 시점의 단어와 연관성이 있는 단어를 집중(attention)시켜 표현해주는 Attention 기법/메카니즘을 사용하여 정확도를 높이고자 한다. 실제로 attention을 통하여 문장을 represent하는 단어와 그 정도를 확인할 수 있어 LSTM 모델과 Attention 메커니즘의 조합은 text classification에서 많이 접목되고 있다. 7가지의 감정으로 emotion analysis를 진행하고자 하는 목적과 매우 부합하기에 이 모델을 구현해보고자 한다.

## (2) 모델 특징

Multiplicative LSTM은 mRNN의 factorized hidden-to-hidden transition과 LSTM의 gating framework를 합친 알고리즘이다. mRNN과 LSTM은 mRNN의 intermediate state  $m_t$ 를 추가하여 연결할 수 있으며 아래 수식의 시스템을 따른다.

$$m_t = (W_{mx}x_t) \odot (W_{mh}h_{t-1})$$

$$h_t = W_{ht}x_t + W_{hm}m_t$$

$$i_t = \sigma(W_{ix}x_t + W_{im}m_t)$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_t)$$

$$f_t = \sigma(W_{ft}x_t + W_{fm}m_t)$$

이때  $m_t$  와  $h_t$  의 차원은 동일하며  $m_t$  는 LSTM의 모든 유닛에 적용된다. 이로 인해 동일한 hidden layer의 LSTM 대비 1.25배 만큼의 recurrent weight를 생성한다.

## (3) 모델 구현

모델은 keras를 사용하여 구현하였다. Multiplicative LSTM은 keras.layers.LSTM 파일에 위 수식을 반영하여 별도의 python 파일을 만들어서 사용하였다. 이하 세부적인 특징은 다음과 같다.

### - Loss Function: Sparse categorical crossentropy

Sparse categorical crossentropy의 경우 다중 분류 손실함수로, categorical crossentropy와는 다르게 one-hot 인코딩을 할 필요가 없다. 현재 정수 타입 인코딩을 진행하였기 때문에 sparse categorical crossentropy를 손실함수로 사용하였다.

### - Optimizer: Adam

Adam optimizer는 stepsize가 gradient의 rescaling에 영향을 받지 않는 장점을 가지고 있다. 즉, gradient가 커져도 bound가 되어 있어서 어떠한 objective function을 사용한다 하더라도 안정적으로 최적화를 할 수 있어서 optimizer로 Adam method를 사용하였다.

Model: "model\_1"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 35)	0	
embedding_1 (Embedding)	(None, 35, 100)	1182800	input_1[0][0]
dropout_1 (Dropout)	(None, 35, 100)	0	embedding_1[0][0]
bidirectional_1 (Bidirectional)	(None, 35, 200)	160800	dropout_1[0][0]
dropout_2 (Dropout)	(None, 35, 200)	0	bidirectional_1[0][0]
time_distributed_1 (TimeDistrib	(None, 35, 1)	201	dropout_2[0][0]
reshape_1 (Reshape)	(None, 35)	0	time_distributed_1[0][0]
attention_vec (Activation)	(None, 35)	0	reshape_1[0][0]
dot_1 (Dot)	(None, 200)	0	dropout_2[0][0] attention_vec[0][0]
dense_2 (Dense)	(None, 100)	20100	dot_1[0][0]
dense_3 (Dense)	(None, 7)	707	dense_2[0][0]
Total params: 1,364,608			
Trainable params: 1,364,608			
Non-trainable params: 0			

그림 5. Multiplicative LSTM Layer Output

정의한 모델에 test/train 데이터를 활용하여 모델을 훈련시켰다. Overfitting을 방지하기 위하여 keras의 EarlyStopping 모듈을 사용하였다. 그림 6은 Test/Validiaton set의 loss/accuracy 결과다.

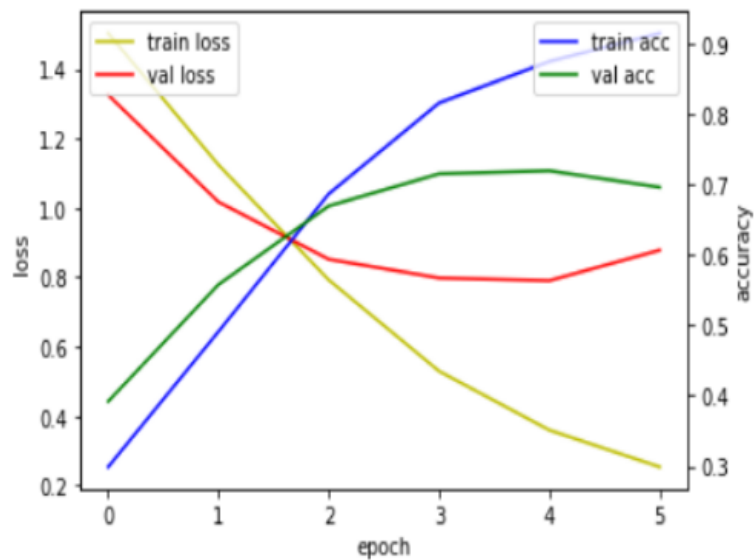


그림 6. Loss/Accuracy of Test/Validation set

앞서 구현한 모델에 attention\_vec layer를 추가하여 attention mechanism을 추가하였다. 이후 Test data 200개를 사용하여 결과를 출력하였다. 그림 8은 “I like romantic comedy, but this movie was not good. When I see the main character feeling low self-esteem and inferiority feeling in the movie, I honestly do not press the pause once or twice.” 의 문장을 mLSTM + Attention 모델에 적용한 결과이다.

```
# Re-create the model to get attention vectors as well as label prediction
model_with_attentions = keras.Model(inputs=model.input,
                                     outputs=[model.output,
                                              model.get_layer('attention_vec').output])
```

그림 7. Attention mechanism 적용

**Text:** like romant comedi movi good see main charact feel low self-esteem inferior honest press paus twice

Original Sentence: I like romantic comedy, but this movie was not good. When I see the main character feeling low self-esteem and inferiority

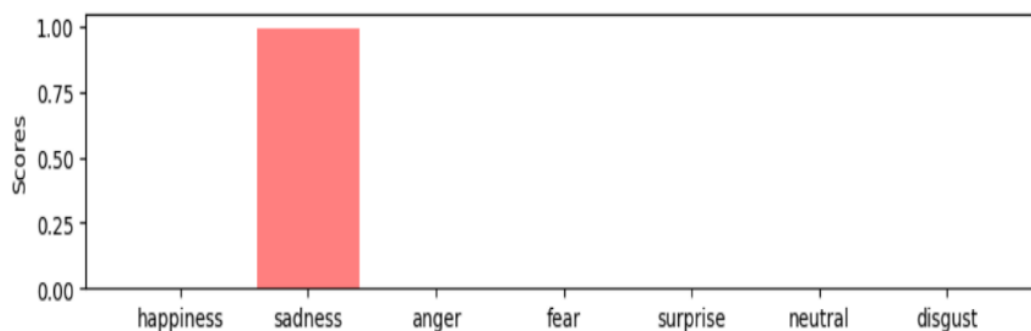


그림 8. mLSTM + Attention 모델 결과 예시

## 2) 장서진

### ① 데이터 수집

7가지의 감정을 잘 예측하였는지 확인하기 위한 test set으로서 영화 리뷰 데이터를 활용하였다. 그 이유는 영화 리뷰는 영화를 본 후 감상평을 적는 것이기 때문에 사람들의 7가지 감정이 잘 녹아들어 있을 거라 판단했기 때문이다.

영화 데이터로서 깃허브에 공개된 NSMC의 네이버 영화 리뷰 데이터를 긍정/부정으로 분류한 자료를 활용하였다. 해당 데이터는 총 20만 개의 네이버 영화 리뷰가 담겨있었으며 부정(0)과 긍정(1)로 레이블링이 되어있었다. 그리고 Large Movie Review Dataset(Maas et al, 2011)를 참조하여 데이터를 구축하였다는 점에서 신뢰도가 있었고 140자 이내 길이의 문장들이 있다는 점이 train set과 유사하였다.

해당 데이터는 긍정과 부정이라는 2개의 감정으로 레이블링이 되어있지만 본 연구는 7가지의 감정을 예측하고자 하였기 때문에 기존 긍정과 부정 레이블링은 제거하여 사용하였다.

### ② NMT API

공식 번역 개발 가이드를 참고하여 Googletrans NMT API를 구현하였다.

```
from googletrans import Translator

for i in range(len(data_train)):
    string = data_train[i]
    translator = Translator(proxies=None, timeout=None)
    result = translator.translate(string, dest="en")
    print(result.text)
```

그림 9. Google NMT API 호출 코드 예제

### ③ 데이터 전처리

mSVM 모델에 적용하기 전에 두 단계를 거쳐 전처리를 진행하였다. 가장 먼저, 텍스트를 토큰 리스트로 변환한 후 각 텍스트에서 토큰 출현 빈도를 센 다음 각 텍스트를 BOW 인코딩 벡터로 변환하는 CountVectorizer를 사용하였다. 이후 CountVectorizer를 사용하여 단어 수를 계산한 뒤 IDF(역 문서 빈도) 값 계산한 후 Tf-idf 점수 계산하는 Tfidftransformer를 사용하여 노멀라이제이션을 진행하였다.

### ④ 모델

#### (1) 모델 선정 이유

N. M. Hakak, M. Mohd, M. Kirmani and M. Mohd, "Emotion analysis: A survey"에서는 감정 분석과 관련하여 감정의 표현, 여러가지 감정 분석 모델, 감정 분석 모델에 적용된 데이터, 여러가지 computational 접근법을 소개하고 있다. 이때 각 감정 분석 모델의 성능을 비교한 부분이 있는데 Balabantaray, Mudasar Mohammad, and Nibha Sharma- Multi-Class Twitter Emotion Classification: A New Approach 아티클에서 사용한 Multi Class SVM의 정확도가 73.24%로 가장 높은 것을 확인할 수 있었다. 따라서 해당 모델을 사용하여 감정 분석을 진행하였다.

#### (2) 모델 특징

KSVM은 SVM과는 다르게 선형적으로 분류되지 않는 클래스를 구분하도록 비선형적 특성을 추가한 모델이다. SVM에 Kernel Trick을 사용하여 데이터를 확장하지 않고 확장된 특성에 대한 데이터 포인트들의 거리를 계산한다. 해당 모델을 사용한 Balabantaray, Mudasar Mohammad, and Nibha Sharma- Multi-Class Twitter Emotion Classification: A New Approach에서는 Crawling한 305,310개의 Tweet 데이터를 사용하였으며 Unigram/Bigrams/Personal-pronouns/Adjectives/Word-net Affect emotion lexicon/Word-net Affect emotion lexicon with left/right context/Word-net Affect emotion POS/POS/POS-Bigrams/Dependency-Parsing Feature/Emoticons를 feature로 사용하였다.

### (3) 모델 구현

multi-class svm 모델을 구현할 수 있는 대표적인 사이킷런의 패키지 3가지(SGDClassifier, LinearSVC, SVC)중 최적의 성능을 내는 알고리즘을 찾고자 하였다. 세 패키지의 알고리즘의 차이는 LinearSVC와 SVC는 각각 멀티클래시피케이션을 'one vs rest'와 'one vs one' 방식으로 한다는 점에서 차이가 있었고 SGDClassifier는 liblinear와 libsvm을 사용하는 두 알고리즘과 다르게 stochastic gradient descent 방식으로 옵티마이징하였다.

각 모델들을 파라미터 튜닝을 해가며 데이터를 7:3 비율로 나누어 train/test set으로 검증 결과에 대해 비교하였다. 그렇게 train/test set에 대한 검증 결과를 가장 높게 나오도록 하는 파라미터를 가지고 정확한 성능 검정을 위해 5-fold 교차 검증으로 세 모델의 성능을 비교하였다. 세 알고리즘의 5-fold의 결과 SGDClassifier가 타 모델에 비해 성능이 0.01이 더 좋았다. 뿐만 아니라 모델의 training 속도도 가장 빨랐다. 그렇게 SGDClassifier 패키지의 모델을 사용하여 감정 분류를 하기로 결정하였다. 이후 SVM의 성능을 높이는 대표적인 방법 중 하나인 Bagging으로 성능을 높여보고자 하였지만 유의미한 결과를 내지는 못하였다.

패키지	조정 파라미터	Best Accuracy	5-fold Accuracy(mean)
SGDClassifier	penalty, alpha, random_state, max_iter	0.747	0.736
LinearSVC	C	0.741	0.735
SVC	kernel, gamma, C	0.748	0.730
SDGClassifier + Bagging	n_estimators, max_samples	0.745	-

표 3. 모델 별 조정 파라미터 및 정확도

이렇게 구현한 multi-class svm으로 레이블링이 되어있는 train set을 학습시킨 후 CalibratedClassifierCV를 사용하여 레이블링이 되어있지 않은 200개의 문장이 담긴 test set에 대하여, 각 문장별로 모델이 낸 예측 확률과 클래스 결과를 얻을 수 있도록 하였다. 이후 각 NMT api별 예측 확률과 클래스 분포를 살펴보았다.

[I'm watching the general elections, and there's no candidate without a criminal record.]는 43.20% 확률로 neutral 리뷰이지 않을까 추측해봅니다.

[The government's framing them to reduce the number of inspections ahead of the general elections, because of the significant decrease in the number of confirmed cases.Those bullies aren't tired, they're evil, and they're angry at the fact that they might be mistaken, and we respect Yoon Bong-gil, not you. "You don't want an old man who fought a war between Korea and Japan," said the granddaughter of Hashirabong-gil.]는 33.60% 확률로 disgust 리뷰이지 않을까 추측해봅니다.

[Leaders of the ruling and opposition parties will vote in advance today at the community center, and on the 15th, we'll take her to the polls and help her vote.For your information, my mother, who will vote on the 15th, says she and I have the same vote [Voting is important. Those who need to be screened out in the general elections.]는 49.22% 확률로 happiness 리뷰이지 않을까 추측해봅니다.

그림 10. 문장별 예측 확률과 클래스 출력 결과 예시

### Google\_nmt\_api test set

- predicted percentage distribution

Accuracy	Count
30%	87
20%	50
40%	49
50%	10
10%	4

- predicted class distribution

Label	Count
neutral	84
disgust	84
sadness	8
happiness	7
anger	6
fear	6
surprise	5

### Kakao\_nmt\_api test set

- predicted percentage distribution

Accuracy	Count
30%	72
20%	62
40%	45
50%	13
10%	7
60%	1

- predicted class distribution

Label	Count
disgust	85
neutral	79
happiness	11
fear	8
sadness	8
surprise	6
anger	3

### Papago\_nmt\_api test set

- predicted percentage distribution

Accuracy	Count
30%	78
40%	49
20%	48
50%	12
10%	2

- predicted class distribution

Label	Count
disgust	94
neutral	68
happiness	8
sadness	8
anger	4
fear	4
surprise	3

그림 11. NMT api별 예측 확률과 클래스 분포



### 3) 정희재

#### ① 데이터 수집

모델의 training data set으로 사용할 7가지의 감정으로 라벨링 된 labeled emotion data set을 구축하였다. 총 3가지의 data set을 각 감정 별 최대 2000개의 라벨된 문장으로 개수를 맞추어 통합된 하나의 training data set을 완성하였다. 사용한 원본 data set들은 모두 kaggle에서 가져왔으며, 일반적인 문장의 데이터와 트위터의 텍스트 데이터가 공존하였다. 이에 대한 전처리는 감정 분석 모델 별로 방식이 상이하여 데이터 수집 단계에서 진행하지 않고 각 모델 별 모델 구축 과정 시 진행하였다.

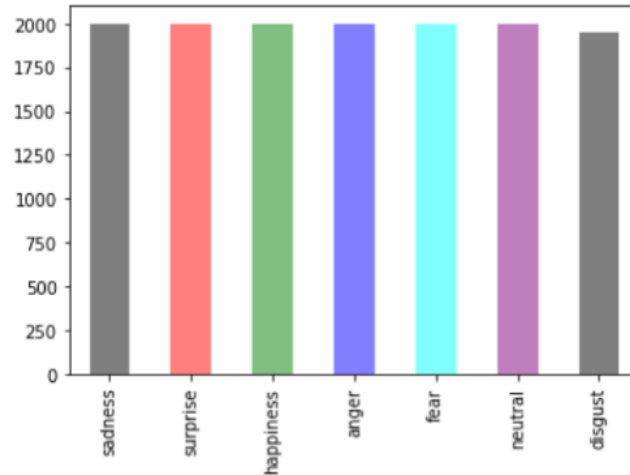


그림 12. Labeled emotion data set

#### ② NMT API

Kakao Developers에서 제공하는 REST API 중 번역 API를 사용하여 크롤링을 통해 수집한 한국어 test data set을 번역한 Kakao test data set을 제작하였다.

```
URL = 'https://kapi.kakao.com/v1/translation/translate'
APP_KEY = {APP KEY}

r = requests.get(URL, headers=headers, params = paras )
json_data = json.loads(r.text)
trans_text = json_data.get('translated_text')
```

그림 13. Kakao NMT API 호출 코드 예제

#### ③ 데이터 전처리

사용한 Transformer 모델의 경우, 문장을 토큰화하여 벡터화 처리 후 지정한 문장의 최대 길이에 맞춰 패딩을 시켜준 뒤 사용하여야 한다. 사용한 training data set에 트위터 데이터가 포함되어 있어 username, hashtag, url 등의 값들을 정규표현식을 이용하여 제거해주는 작업이 우선적으로 진행되었다. 이후, nltk의 stopwords와 기본적인 punctuation 등을 포함한 불용어 처리를 진행하였고, 어간 추출(stemming) 작업을 통해 training data set으로 구축하는 단어 사전(vocabulary)에서 어간 별 중복도를 높여 의미 전달이 극대화될 수 있도록 하였다. 마지막으로 정수 인코딩과 패딩 처리 후 모델의 input으로 사용하였다. 그림 14에서 데이터 전처리 순서에 맞춰 전처리 과정이 이루어지는 문장 예시들을 확인할 수 있다:

원문	""@StuartPaulo i don't think John Lennon even trended today, did he? ""	@toosweetmel lmfao yeah when you ask bout them ppl look at you with like they ate something bad..'
데이터 정제 및 정규화	"" i do not think john lennon even trended today, did he? ""	lmfao yeah when you ask bout them ppl look at you with like they ate something bad..'
토큰화, 어근 추출, 불용어 처리	['think', 'john', 'lennon', 'even', 'trend', 'today']	['lmfao', 'yeah', 'ask', 'bout', 'ppl', 'look', 'like', 'ate', 'someth', 'bad']
Vocabulary 정수 인코딩	[12, 1846, 11909, 21, 2671, 35]	[5366, 339, 141, 1215, 713, 30, 2, 865, 36, 11911]

그림 14. 데이터 전처리 예시

#### ④ Transformer 모델

##### (1) 모델 선정 이유

기존의 recurrent model(RNN 등)은 순차적인 특성이 유지되는 뛰어난 장점이 있지만, 어떤 정보와 다른 정보 사이의 거리가 멀 때 해당 정보를 이용하지 못하는 'Long-Term Dependency Problem'이 존재한다. 이를 보완하기 위해 attention 기법 중에서도 multi-head self-attention을 사용하는 Transformer 모델을 도입함으로써 긴 문장 내에서도 각 정보들이 유의미하게 활용될 수 있도록 하였다.

##### (2) 모델 특징

기존 rnn, lstm, sequence-to-sequence 등 다양한 NMT(Neural Machine Translation) 모델이 존재함에도 불구하고 Transformer 모델이 많이 알려지게 된 이유는 기존의 모델들보다 훨씬 단축된 시간 동안 더 좋은 성능을 내기 때문이다. Transformer 모델은 recurrence를 사용하는 대신 attention 기법만을 사용하여 input과 output의 dependency를 포착해 낸다. 그 중에서 multi head attention을 통하여 여러 소단위로 쪼개 병렬처리를 해줌으로써 한 문장에 대해 다양한 구간에서 attention 계산이 가능하여 이를 통해 문장 내 단어들 간의 가중치 계산이 보다 효과적이다. 또한, self-attention, 즉 레이어간 input과 output 전달의 연결을 통해 이전 레이어의 output을 현재 레이어의 input으로 전달해줌으로써, position 정보를 추가해주게 된다. 이 position 정보를 통해 불필요한 위치를 다시 확인할 필요가 없어져 효과적인 output 도출이 가능한 것이다.

##### (3) 모델 구현

Transformer의 가장 큰 특징은 multi head self-attention과 positional embedding을 사용하는 것이다. 각 layer를 구현하여 transformer block을 구현하였다. input 문장의 최대 길이는 training data set의 최대 길이였던 35를 기준으로 정하였고, dimension 및 attention head의 개수 조정을 여러 차례 시도한 결과 embedding size와 hidden layer size를 32로, attention head는 2개로 설정하여 모델을 구현하였다. Loss function으로는 Sparse Categorical Cross Entropy를, optimizer로는 Adam을 사용하였다. 아래 그림 15의 구조를 가진 모델을 구현하였고, 최종 validation loss와 validation accuracy는 0.8691과 0.7155로 약 71%의 테스트 정확도를 가졌다.

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 35)]	0
token_and_position_embedding (None, 35, 32)		382336
transformer_block (Transform (None, 35, 32)		6464
global_average_pooling1d (G1 (None, 32)		0
dropout_2 (Dropout)	(None, 32)	0
dense_6 (Dense)	(None, 20)	660
dropout_3 (Dropout)	(None, 20)	0
dense_7 (Dense)	(None, 7)	147
Total params: 389,607		
Trainable params: 389,607		
Non-trainable params: 0		

그림 15. Transformer Model Layer Summary

### 3. 수행결과

#### 1. NMT API - 모델 분석 결과

각 NMT API별로 MSVM, mLSTM Attention, Transformer를 조합하여 분석한 총 200개의 결과 중 10개의 데이터를 표로 나타냈었다. 표 1은 Google NMT API와 각 모델의 조합이며 차례대로 Kakao, Papago NMT API와 모델의 결과를 표 4, 5, 6을 통해 나타내었다.

문장	라벨	Google NMT API		
		MSVM	MLSTM	TF
저예산으로 만들었다는데 상당히 재밌다 오락영화로서는 합격점이다	surprise	surprise	neutral	surprise
삶의 끝에 죽음이 기다리는 것처럼 사랑에도 끝이 존재하기에 사람이기에 두렵기에	fear	fear	fear	fear
김기덕은개뿔실망이다김기덕감독이보면웃을듯	disgust	disgust	disgust	disgust
너무 기억에 남는 영화쓰레기라고 적은 인간은 뭐냐 영화 보긴 했나	anger	neutral	disgust	neutral
와진짜 이게 뭐냐 내용 막장에 음악도 정말 하 보지 말걸	disgust	anger	disgust	disgust
치유는 고백 이전에 범죄자를 처벌하는 것부터 시작해야 한다	neutral	neutral	anger	anger
진짜 해도해도 너무한다 시청자상대로 장난질 하는 것 같은데이런쓰레기 만든 작가 감독 고소 하는 방법없냐 야그만해라	disgust	neutral	disgust	disgust
멋져요 인생은 진짜 짜여진 각본인거같아서 참 슬프고 보면서 우울했어요	sadness	happiness	disgust	neutral
이 드라마 진짜 재밌게 봤는데 그래도 난 재밌었다	happiness	neutral	neutral	surprise
미친 영화를 보고 내입에서 찰진 욕만 나왔다 이런걸 독립 영화 드립 치다니	anger	neutral	disgust	surprise

표 4. Google NMT API - Model 결과 비교

문장	라벨	Kakao NMT API		
		MSVM	MLSTM	TF
저예산으로 만들었다는데 상당히 재밌다 오락영화로서는 합격점이다	surprise	sadness	neutral	surprise
삶의 끝에 죽음이 기다리는 것처럼 사랑에도 끝이 존재하기에 사람이기에 두렵기에	fear	fear	fear	fear
김기덕은개뿔실망이다김기덕감독이보면웃을듯	disgust	neutral	disgust	disgust
너무 기억에 남는 영화쓰레기라고 적은 인간은 뭐냐 영화 보긴 했나	anger	disgust	disgust	surprise
와진짜 이게 뭐냐 내용 막장에 음악도 정말 하 보지 말걸	disgust	disgust	disgust	surprise
치유는 고백 이전에 범죄자를 처벌하는 것부터 시작해야 한다	neutral	neutral	surprise	surprise
진짜 해도해도 너무한다 시청자상대로 장난질 하는 것 같은데이런쓰레기 만든 작가 감독 고소 하는 방법없냐 야그만해라	disgust	neutral	disgust	surprise
멋져요 인생은 진짜 짜여진 각본인거같아서 참 슬프고 보면서 우울했어요	sadness	sadness	disgust	surprise
이 드라마 진짜 재밌게 봤는데 그래도 난 재밌었다	happiness	disgust	neutral	surprise
미친 영화를 보고 내입에서 찰진 욕만 나왔다 이런걸 독립 영화 드립 치다니	anger	disgust	disgust	neutral

표 5. Kakao NMT API - Model 결과 비교

문장	라벨	Papago NMT API		
		MSVM	MLSTM	TF
저예산으로 만들었다는데 상당히 재밌다 오락영화로서는 합격적이다	surprise	sadness	neutral	surprise
삶의 끝에 죽음이 기다리는 것처럼 사랑에도 끝이 존재하기에 사람이기에 두렵기에	fear	disgust	fear	fear
김기덕은개뿔실망이다김기덕감독이보면웃을듯	disgust	neutral	surprise	disgust
너무 기억에 남는 영화쓰레기라고 적은 인간은 뭐냐 영화 보긴 했나	anger	disgust	neutral	neutral
와진짜 이게 뭐냐 내용 막장에 음악도 정말 하 보지 말걸	disgust	disgust	disgust	neutral
치유는 고백 이전에 범죄자를 처벌하는 것부터 시작해야 한다	neutral	neutral	surprise	surprise
진짜 해도해도 너무한다 시청자상대로 장난질 하는 것 같은데이런쓰레기 만든 작가 감독 고소 하는 방법없냐 야그만해라	disgust	disgust	disgust	disgust
멋져요 인생은 진짜 짜여진 각본인거같아서 참 슬프고 보면서 우울했어요	sadness	sadness	disgust	surprise
이 드라마 진짜 재밌게 봤는데 그래도 난 재밌었다	happiness	disgust	surprise	neutral
미친 영화를 보고 내입에서 찰진 욕만 나왔다 이런걸 독립 영화 드립 치다니	anger	disgust	surprise	neutral

표 6. Papago NMT API - Model 결과 비교

이후 표 7과 표 8을 통해 분석 결과를 정량화 하였다. Movie, Twitter 행은 input 라벨과 output 라벨이 일치하는 빈도를 나타낸다. 그 결과 Google NMT API를 사용했을 때는 MSVM이 총 200개의 문장 중에서 92개의 감정 데이터를 맞추며 가장 높은 정확도를 보였다. Kakago NMT API와 Papago NMT API를 사용했을 때는 MLSTM이 각각 71개와 73개를 맞추며 가장 높은 정확도를 보였다. 또한 Movie 데이터와 Twitter 데이터에 두드러지는 차이점이 존재하였기 때문에 별도의 분석을 진행하였다. 영화 데이터는 문장의 길이가 비교적 짧고 여러 감정이 혼재해 있는 반면, Twitter 데이터는 문장의 길이가 비교적 길고 감정이 분명한 문장이 상대적으로 많았다. 따라서 각 데이터별 precision과 모든 데이터의 precision을 분석하였다.

라벨	구글			카카오			파파고		
	MSVM	MLSTM	TF	MSVM	MLSTM	TF	MSVM	MLSTM	TF
Movie	35	39	32	31	39	29	34	36	27
Twitter	57	34	24	57	32	16	51	37	16
Total	92	73	56	88	71	45	85	73	43
M_Precision	0.35	0.39	0.32	0.31	0.39	0.29	0.34	0.36	0.27
T_Precision	0.57	0.34	0.24	0.57	0.32	0.16	0.51	0.37	0.16
Total Precision	0.46	0.37	0.28	0.44	0.36	0.23	0.43	0.37	0.22

표 7. NMT API - Model 비교 분석 결과

라벨	mSVM			mLSTM			Transformer		
	구글	카카오	파파고	구글	카카오	파파고	구글	카카오	파파고
Movie	35	31	34	39	39	36	32	29	27
Twitter	57	57	51	34	32	37	24	16	16
Total	92	88	85	73	71	73	56	45	43
M_Precision	0.35	0.31	0.34	0.39	0.39	0.36	0.32	0.29	0.27
T_Precision	0.57	0.57	0.51	0.34	0.32	0.37	0.24	0.16	0.16
Total Precision	0.46	0.44	0.43	0.37	0.36	0.37	0.28	0.23	0.22

표 8. Model - NMT API 비교 분석 결과

	Model	NMT API	Precision
영화 데이터	MLSTM	Google/Kakao	0.39
트위터 데이터	MSVM	Google/Kakao	0.57
총평	MSVM	Google	0.46

표 9. 가장 높은 Precision의 Model - NMT API 조합

따라서 영화 리뷰 데이터와 같이 문장이 짧고 감정이 다양한 Input Data를 사용할 경우에는 구글 또는 카카오 NMT API와 mLSTM + Attention 모델 조합을 사용하는 것이 성능이 가장 우수하다는 것을 알 수 있다. 반면 트위터 데이터와 같이 문장이 대체적으로 길고 감정이 한정적인 Input Data를 사용할 경우에는 구글 또는 카카오 NMT API와 mSVM 모델 조합을 사용하는 것이 가장 우수하다. 마지막으로 전체 데이터를 사용했을 때는 구글 데이터와 mSVM 모델 조합을 사용했을 경우 0.46으로 성능이 가장 우수하였다.

#### 4. 기대효과 및 활용방안

##### 가. 기대효과

한국어는 영어에 비해 상대적으로 적은 인구가 사용하는 언어이다. 이에 한국어 자연어 처리를 위한 데이터의 양이 영어에 비해 현저히 적으며 그 데이터의 질과 다양성에서도 큰 차이가 난다. 또한, 교착어라는 한국어의 특성으로 인해 sentiment analysis, emotion analysis, intent analysis 등과 같이 감정에 대한 분석이 요구되는 자연어 처리 활용 예제가 많지 않다. 이에 영어의 경우 Ekman의 basic emotion model의 7가지 감정 카테고리(happy, sad, anger, disgust, surprise, fear)를 기반으로 감정 분석 classification을 진행하여 소비자 감정 예측, 여론조사 파악 등에 널리 활용되고 있는 반면, 한국어는 단순 긍정/부정이 아닌 multi-class로 라벨링된 감정 데이터마저 존재하지 않아 multi-category 감정 분석 모델이 존재하지 않는다. 이번 연구를 통해 따로 감정으로 라벨링된 한국어 데이터가 없어도 한국어 문장을 7가지의 감정으로 multi-category classification이 가능한 모델을 구축하였다. 해당 모델은 시간 소모적인 라벨링 단계를 거치지 않고도 한국어 문장 데이터를 다양한 감정으로 감정 분석이 가능하다는 장점이 있어 텍스트 데이터를 통한 감정 파악 및 예측에 다방면 활용될 수 있을 것이다.

##### 나. 활용방안

미국의 경우 2016년 대선 당시 트럼프 대통령의 대선 행보에 대하여 사람들의 반응을 조사하는데 기존 여론조사의 단점인 적은 표본 수를 보완하고자 Ekman의 6가지 감정 카테고리를 활용한 유권자 분석을 진행하였다. 다른 모든 여론조사 방식들이 트럼프 당선의 예측을 실패한 가운데 데이터 벤처기업인 제닉AI가 개발한 '모그AI'가 실제로 SNS 데이터를 기반으로 한 sentiment analysis를 통해 유일하게 트럼프 당선을 예측하였다. 이처럼, SNS 데이터가 방대해짐에 따라 선거 당선 예측뿐만 아니라 정부의 각종 정책에 대한 시민들의 반응을 조사하는 데도 유의미하게 사용될 수 있을 것이다. 또한, 현재 COVID-19와 같은 국가적 재난 상황에서 시민들의 반응을 분석하는 데에도 유용하게 사용될 수 있을 것으로 예상된다.

#### 5. 결론 및 제언

모델의 전반적인 성능이 좋지 못하며, 가장 좋은 precision을 가진 Google NMT API와 Multi-Class SVM Kernels의 조합은 0.46의 precision으로 낮은 성능을 보였다. 영어 데이터를 사용하였을 때 평균 precision 80% 이상의 높은 성능을 보이는 모델들이 한국어 데이터를 활용하였을 때 precision이 낮게 나온 이유를 다음과 같이 분석하였다.

##### 1. 훈련 데이터

모델 훈련에 사용한 훈련 데이터 셋의 수가 총 약 14,000개로 많지 않으며 라벨링 또한 명확한 기준으로 부여되지 않았다. 예를 들어 주어진 7가지 감정 라벨 중에서 happy와 surprise를 구분하기 어려웠으며, anger와 disgust 또한 구분하기 어려웠다. 훈련 데이터 분석 결과 통일된 기준이 적용되지 않고 상당 부분 개인의 주관에 반영된 것을 확인할 수 있었다. 적은 훈련 데이터와 일관되지 않은 라벨링 기준이 모델의 전반적인 성능 저하에 영향을 미쳤을 것으로 예상된다.

## 2. 테스트 데이터

NMT API별 성능차이로 인하여 같은 문장을 입력 데이터로 사용하더라도 출력 결과가 각기 다르다. 이 과정에서 주어진 한글 문장이 영어로 제대로 번역이 되지 않아 문맥의 의미, 즉 감정 데이터가 변질되는 경우가 많았다. 이와 더불어 본 연구에 사용된 데이터는 댓글 및 SNS 데이터로 감정 분석을 위해 최적화된 데이터가 아니었으며 감정 데이터의 혼재와 데이터의 질적 문제로 인하여 정확한 감정 예측이 어려웠다. 예를 들어 영화 문장의 경우 여러 감정이 한 문장에 혼재한 경우가 많았으며, 트위터 데이터의 경우 완성된 문장이 아닌 경우가 많았다.

모델의 성능이 다소 낮았지만 위 두 가지 문제는 여러 가지 방법을 통해 극복할 수 있다. 훈련 데이터 문제의 경우 일관되게 라벨링된 데이터를 더 많이 수집하여 모델을 훈련시켜 극복할 수 있으며 두 번째 문제 또한 감정 분석에 최적화된 데이터를 사용한다면 지금보다 더 높은 성능을 기대할 수 있을 것이다.

※ 본 양식은 요약보고서이며, 최종결과물을 필히 추가 제출하여야 함.

팀 학생대표 성명 : \_\_\_\_\_ 엄 세 응 (인)

