'

자연어 처리를 활용한 한국어 유튜브 댓글 분류

1. 개요 및 목적

1.1. 프로젝트 개요

본 프로젝트는 한국어 자연어처리(NLP)모델을 만들어 유튜브 사이트에 달린 댓글의 종류를 분석하는 모델을 만드는 것을 목적으로 한다. 프로젝트는 데이터 수집, 형태소 분석 등의 데이터 전 처리, 데이터의 feature를 뽑아내기 위한 모델 설계, 분류모델 설계의 순서대로 진행될 예정이며 설계된 분류모델의 정확도가 실용적으로 활용할 수 있을 정도라고 판단될 경우 브라우저 확장 프로그램, 채널 관리 보조 도구 등의 형태로의 활용을 고려할 것이고, 이외의 경우 정확도 개선을 목표로 두고 프로젝트를 진행할 것이다

1.2. 개발 동기

최근 여러 메신저, SNS나 동영상, 숏폼 등을 게시하는 다양한 플랫폼들이 강세를 띄고 있다. 그림 1.1을 살펴보면 13세이상 60세 미만의 연령대에서 90% 이상 이 모바일 메신저와 유튜브를 사용하고 있다고 답했다. 이러한 플랫폼에서 생산되는 댓글, 반응들의 데이터는 인간이 직접 분류하며 이용하기에는 한계가 있다.

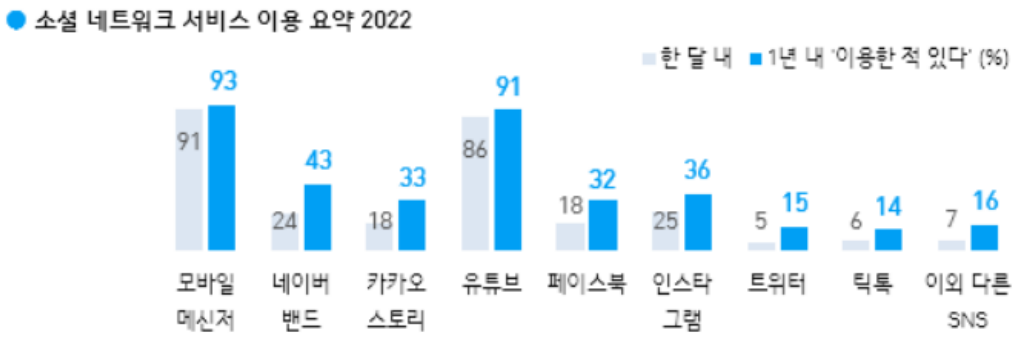

그림 1.1 소셜 네트워크 서비스 이용 요약 2022

본 연구를 통해 많은 활용 가치가 있는 이러한 데이터들을 직접 수집하고 목적을 정해 분류하고 적용해보려 한다.

1.3. 최종 구현 목표

수집된 유튜브 영상의 댓글 데이터를 형태소 분석 등의 전처리를 수행하고 word embedding 모델의 값을 가중치를 적용하여 댓글 문장을 continuous vector로 표현하는 것이 1차 목표이다. 이 후 클러스터링을 통해 분류를 진행하고 PCA, t-sne분석을 통해 분류 결과를 시각화하고, 정확도를 개선과 분류 결과를 다른 곳에 응용할 수 있을 정도로 개선하는 것이 최종 목표이다. 추가적으로 word embedding 모델을 학습시킬 수 있을 정도의 데이터가 수집된다면 word2vec, fasttext, glove 세 모델을 학습시켜보고 테스트 데이터가 분류 된 결과를 비교해 보려 한다.

2. 배경

2.1. 자연어 데이터 처리의 중요성 대두

설문 조사를 통한 통계자료, 인간이 분류한 데이터를 사용했던 과거와는 달리 오늘날 생산되고 있는 데이터들은 사전 라벨링이 되어 있지 않은 경우도 많고 데이터의 크기가 매우 커 유의미한 의미를 찾아내기 위해서 인간이 직접 데이터를 분류하는 것은 수지타산이 맞지 않는다. 프로젝트의 초점은 유튜브 한국어 댓글 분류지만. 데이터를 수집, 전처리하고 분류하는 과정에서, 데이터에서 가치가 있는 특징들을 찾아내는 좋은 경험일 것이라 생각한다.

2.2. LLM(Large Language Model)의 등장

자연어를 이해하고 생성할 수 있는 GPT-3와 같은 LLM들이 경쟁적

으로 생겨나고 있다. 자연어를 다루는 여러가지 기술이 강세라는 것은 그만큼 자연어를 분석하고 의미 있는 정보를 뽑아내어 사용하는 것이 중요하다고 할 수 있다. 파라미터의 개수나 데이터의 양이 압도적인 LLM의 성능을 따라잡을 수 없지만 범용적으로 설계되어 있는 LLM과는 달리 더 단순하고 특정 목적만을 위한 모델을 설계하여 더 특정 목적에 한에서 더 적은 자원으로 유의미한 성능을 얻을 수 있다면 상황에 따라 의미있게 사용될 수 있을 것이라 생각한다.

3. 설계 및 구현

3.1 프로젝트 일정

구분	추진내용	프로젝트 기간					
		3월	4월	5월	6월	7월	8월
계획	요구사항 분석						
	계획서 작성						
스터디	데이터 크롤링 스터디						
	데이터 셋 전처리 스터디						
	문장 분류 모델 스터디						
데이터	데이터 수집 및 전처리						
모델 설계	Word embedding 모델 설계						
	문장 분류 모델 설계						
구현	모델 구현						

그림 1.2 프로젝트 일정

구분	추진내용	프로젝트 기간					
		7월	8월	9월	10월	11월	12월
데이터 수집	약 10만개까지 수집 예정						
모델 설계	Word embedding 모델 설계						
	Sentence embedding 모델 설계						
모델 학습	Word embedding 모델 학습						
모델 평가	k-means clustering						
	PCA, t-sne 분석 시각화						
	스팸 댓글 분류 여부						

그림 1.2 프로젝트 일정

3.2 프로젝트 설계

3.2.1 데이터 수집 및 전처리

데이터 수집으로는 Youtube api와 Selenium프레임 워크를 사용하여 스크랩했다. 각각 video_id와 사이트 URL을 입력하면 해당 영상의 댓글 데이터를 수집할 수 있다. Youtube api를 사용하면 해당 영상의 모든 댓글을 한 번에 수집할 수 있고, Selenium을 사용하면 로딩 조건에 따라 수집되는 댓글의 수가 약 1000개에서 5000개 가량의 데이터를 수집할 수 있다. 하지만 Youtube api의 경우 사용할 수 있는 할당량이 존재하고, 다양한 영상의 데이터를 수집하기 위해 두 방법을 병행하여 데이터 수집을 진행했다.

3.2.2 분류 모델 구현

문장을 분류하기 위해서 문장을 Continuous Vector로 표현하는 모델을 구하고 이를 통해 구한 값을 비지도학습의 방식으로 군집화 하

는 방식을 사용했다. Sentence embedding 모델은 문장의 embedding 값을 문장을 구성하는 단어의 embedding vector 값의 평균값으로 계산하는 모델을 기본으로 구성하고 가중치를 임의로 조정하여 개선을 시도했다. Word embedding 모델은 미리 학습된 word2vec 모델과 약 10만개의 trainData를 통해 학습시킨 word2vec 모델과 fasttext 모델을 사용했다.

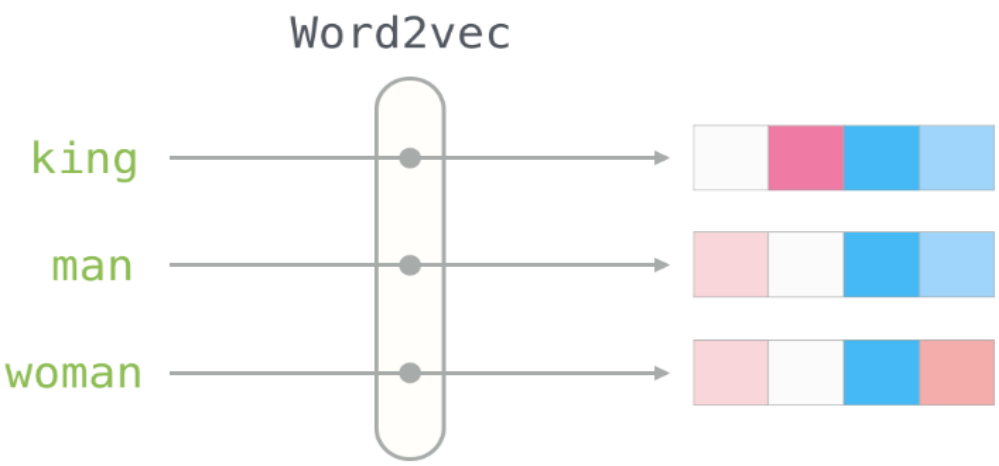

그림 1.3 Word2vec 모델

Word2vec 모델은 word를 embedding하는 방법 중 하나로 단어의 특성을 벡터화 하여 부여하고 유사한 특성을 지닌 단어들이 같은 벡터 방향을 띄게 하는 방식으로 구현된다. 단점으로는 학습 데이터에 존재하지 않아 단어 사전에 없는 단어의 경우에는 임베딩 벡터값을 정의할 수 없고 학습에서 사용한 window 파라미터 내의 주변 단어를 유사 단어라고 가정하는 방식이기 때문에 멀리 떨어진 단어 사이의 관계를 고려하지 않는다. 따라서 상대적으로 문장의 전체적인 의미가 임베딩 값에 반영되지 않을 수 있다.

위의 문제들을 고려하여 해당 연구에서는 word2vec 모델과 함께 fasttext 모델을 사용하여 결과를 비교해 보았다. Fasttext model은 단어의 embedding 값을 구할 때 단어를 분할하여 각각의 값을 구

하고 이를 더해 계산한다. 해당 방식때문에 사전에 존재하지 않는 단어라도 embedding 값을 구할 수 있는 가능성이 존재하기 때문에 word2vec 모델보다 개선된 결과를 얻을 수 있으리라 가정했다.

Sentence embedding의 경우 본 프로젝트에서는 1~2문장으로 이루어진 짧은 한국어 문장에서 단어의 순서를 바꾸어도 문장의 의도가 크게 바뀌지 않는다고 가정하고, 문장에서 단어가 등장하는 순서는 고려하지 않는다. 그래서 앞에서 언급한 세 word embedding 모델을 사용하여 각 문장을 구성하는 단어의 embedding 값의 평균값을 해당 문장의 embedding 값으로 계산했다.

해당 방법은 문장을 구성하는 각 단어가 의미적으로 같은 가중치를 가진다고 가정한다. 하지만 실제로는 문장 안에서 다른 문장보다 더 큰 비중을 가지는 단어가 존재하고 이러한 것을 고려하지 않으면 길이가 긴 문장일수록 중요한 의미를 가지고 있는 단어의 임베딩 값이 희석되는 문제가 발생할 것이다. 문제 해결을 위해 가능하다면 각 단어마다 가중치를 두어서 계산하는 방법을 시도해 보려 했다. 우선은 평균값이나 임의로 정한 기준을 통하여 가중치를 조정해 보고 개선이 필요하다면 모델을 학습할 때의 데이터에서 각 단어의 등장확률을 통해 해당 단어에 가중치를 적용하는 SLF(smooth inverse frequency) 방식을 적용해 볼 것이다.

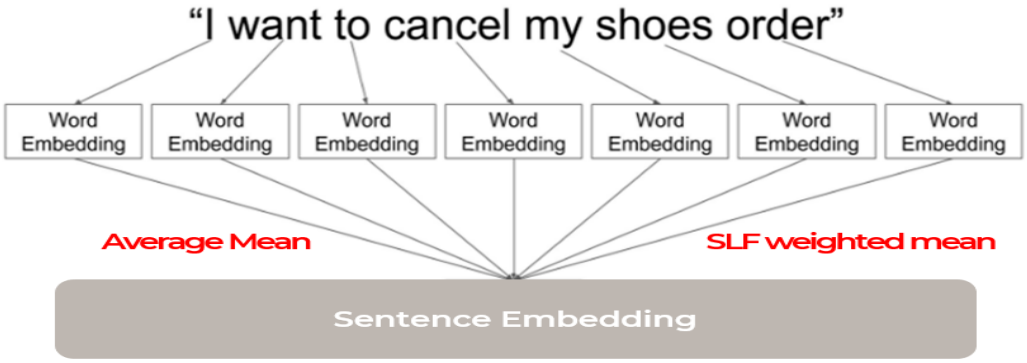

그림 1.4 Sentence Embeeding 모델

3.2.3 전체 동작

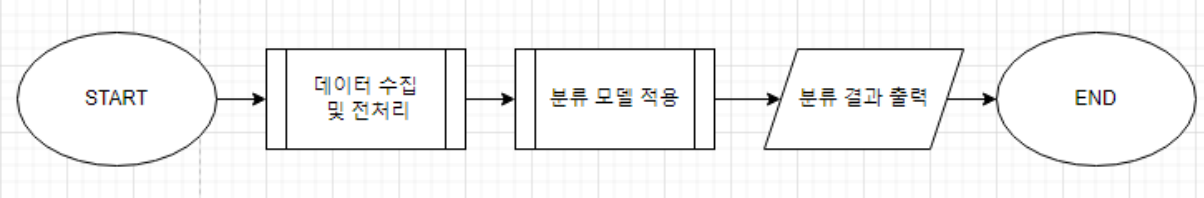

그림 1.5 전체 동작 흐름도

그림 1.5는 프로젝트의 전체 동작을 간단하게 나타낸 흐름도이다. 먼저 데이터를 수집하고, 단어 단위로 전처리한 후 3개의 word embedding model을 통해 얻은 vector값을 평균값, 또는 가중치 적용하여 계산하여 문장을 나타내는 continuous vector값를 구한다. 이후 k-means clustering, PCA, T-sne 방식의 차원축소를 통해 문장이 군집을 이루고 있는지 확인하고, 긍정-부정-스팸 세가지로 분류해 둔 Testcase를 적용하여 군집을 이루는지 확인해 보았다.

4. 실행 결과

4.1. 데이터 수집 및 전처리 결과

	아이디	
0	ㅏ)ㅏ	방장이 얼마나 열심히 일해놨으면 아직까지 빈자리를 채우는 침튜브.. 항상 감사하다
1	박상헌	방장이 없어도 돌아가는 침튜브, 이게 바로 제로 아닐까?
2	쓸데없지만재밌어	침착맨마저 제로가 된 세상.
3	덕키	쉴때 든든한 고봉밥 만들어 놓아서 이렇게 침손실없이 영상보네 침순이는 행복합니다
4	플레르드뽀	침착맨 쉰다니까 침착맨원본박물관,침착맨플러스 정독하고 있음 ... 오히려 좋아 ..밀린 거 많았는데 따라가라고 시
5	부장아재해돌쿤	아니 얼마나 남겨 놨길래 유투부쟁이들은 개방장 방송 쉬는걸 느낄수가 없네 침수자들 넘모 고마워요 요로분♡
6	xnbdjsak	뇌절과 호들갑에 누구보다 심술을 잘 내면서 관련 컨텐츠는 꼭 찍어 내보내는 역설적인 유튜버
7	오현우	나의 저녁을 책임져준 침착맨 제로 감사하다
8	Nateee	이 날 침하하에서 침착맨 사랑해로 방송시작하자해서방송 시작 할 때 다 같이 침착맨 사랑해 채팅 쳤는데컨텐츠하
9	우주고양이	이렇게 쉬더라도 끝없이 편집본이 나오는걸보니 이제 격달제로 라이브를 할지도
10	Play- maker	침착맨없이도 굴러가는 침튜브를 봤으니 이제 안심하고 원펀데이를 맞이할수있겠다
11	지니	방송을 쉬면서도 계속 올라오는 침튜부 감사합니다
12	콩	펩시제로 라임맛 뒤로 빼는 거 보고 이 사람은 진짜 우리가 어느 포인트에서 열받는지 알고 있다는 생각이 들었음
13	검은냥이 코코& 잡덕집사	ㅋㅋㅋ 맛나게 드시지만 점점 뒤에서 배불러하는 맥콜병건 너무 잼나요
14	쮸스	방장 없이 돌아가는 침튜브... 방장 얼마나 일을 하고 간고야
15	김도지	지속가능한 침튜브를 위해 지금같이 열흘하고 품앗이 다니고 휴식 취하는건 어떨까
16	seunghun	제로가 되어버린 방장의 제로 음료 리뷰이게 진짜 제로 아닐까?
17	ㅅㅇ	환타제로 진짜 맛있지. 먹고 놀라서 한박스 쟁여놨었음
18	널디언니사랑해	"제로"침튜브에 올라온 "춘추제로시대".....이건 귀하다...
19	주마등	침착맨은 이제 하나의 기업이다. 그가 없어도 톱니바퀴는 돌아간다...
20	이윤경	폭력적인 귀여움에 가날픈 손꾸락까지 완벽하다

그림 1.6 수집 데이터 예시

그림 1.6는 수집된 데이터를 출력한 결과이다. 유튜브 영상 하나를 기준으로 약 1000개에서 6000여개의 데이터를 수집하였으며 영상별로 xlsx형태의 파일로 저장했다.

데이터 전처리의 경우 우선 파이썬에서 제공하는 한국어 형태소 분석 패키지 Konlpy를 사용하여 조사를 제거하고 명사형태의 단어로 정리했다. 이 후 불용어를 제거하였는데 영어에서 잘 적용되었는데 한국어의 어간, 어미의 특징 때문인지 잘 작동하지 않아서 정규식을 사용해 어간 불용어, 어미 불용어를 구분해서 처리했다. 추가적으로 학습에 사용한 데이터에서 단어의 등장 빈도를 기준으로 정리하여 자주 등장하지만 분류에 도움이 되지 않는 단어들을 불용어사전에 추가했다. 하단 그림 1.7은 전처리 이후 데이터가 저장된 모습이다.

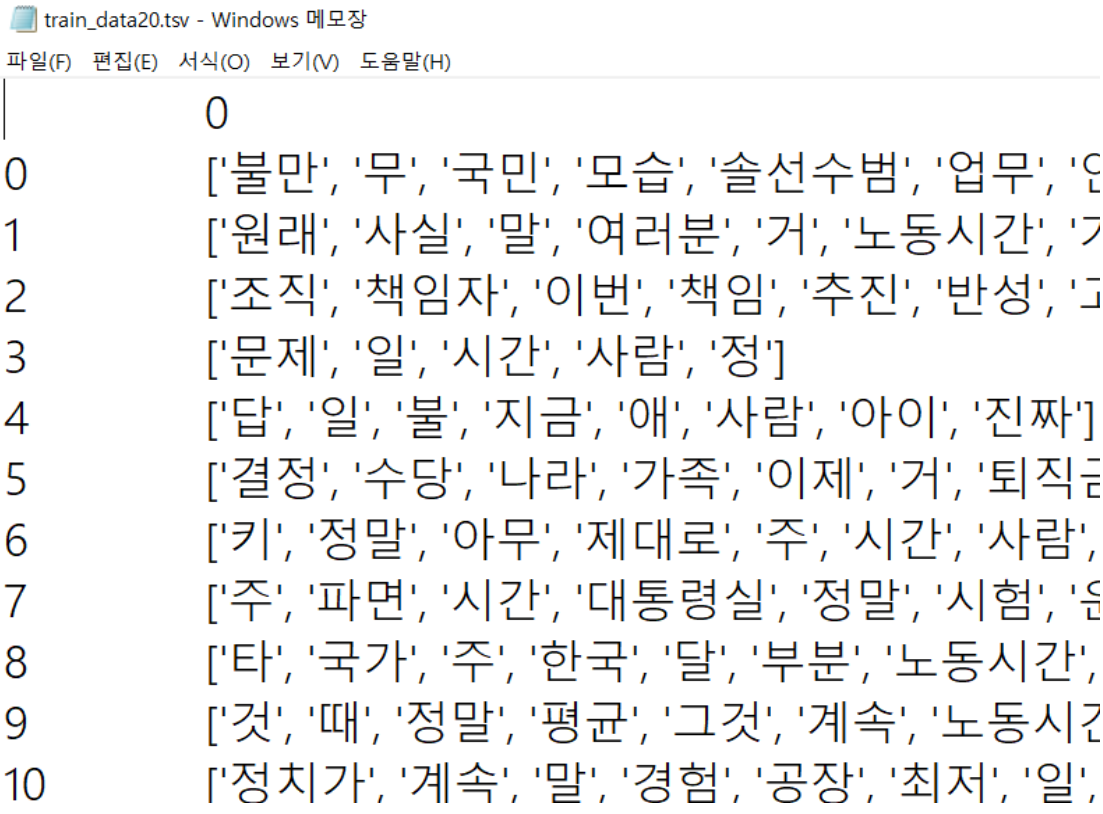

그림 1.7 전처리 후 데이터

4.2. 분류 결과

4.2.1 문장 간 유사도 측정

데이터를 수집하고 pre-trained word2vec 모델을 사용해서 분류를 진행했다. 그림 1.8은 문장의 유사도를 측정한 방식으로 sentence embedding이후 각 vector값의 코사인 유사도를 측정하여 유사한 문장 사이의 유사도를 측정했다.

```
print('문서 1과 문서2의 유사도 :',cos_sim(sentence2vec[0], sentence2vec[1]))
print('문서 1과 문서3의 유사도 :',cos_sim(sentence2vec[0], sentence2vec[2]))
print('문서 1와 문서4의 유사도 :',cos_sim(sentence2vec[0], sentence2vec[3]))

similarity = []

input_sentence_vec = sentence2vec[0]
temp = 0
for sentence in sentence2vec:
    similarity.append(cos_sim(input_sentence_vec, sentence))
```

```
문서 1과 문서2의 유사도 : 0.29738772
문서 1과 문서3의 유사도 : 0.4482292
문서 2와 문서3의 유사도 : 0.5610777
```

그림 1.8 문장 유사도 측정 방식

입력 문장을 기준으로 가장 높은 유사도를 가진 댓글 5개를 출력하는 방식으로 문장의 embedding이 잘 적용되었는지 확인했다.

그림 1.9 분류결과(1)를 살펴보면 입력 텍스트에 투표라는 단어가 포함되어 있고, 유사도가 가장 높은 5개의 문장도 투표를 포함한 문장이었다.

```
input sentence : 서민들이 부자와 기득권을 위해 투표잘했습니다.
top_5 similarity sentences
국민 여러분들 투표 제대로 합시다 좀
이래서 투표를 잘해야한다
절대로 친일당 다신 뽑지않으리 이렇게 투표가 중요합니다
불만가지는놈들 누구 투표햇는지 알고싶네 ㅎ
투표의 중요성!!!
```

그림 1.9 분류 결과(1)

또한 그림 1.10의 분류결과(2)에서는 대통령, 솔선수범 등의 단어가 핵심인 문장을 입력 값으로 넣었을 때 출력된 문장들 모두 비슷한 의미를 지닌 문장들이었다.

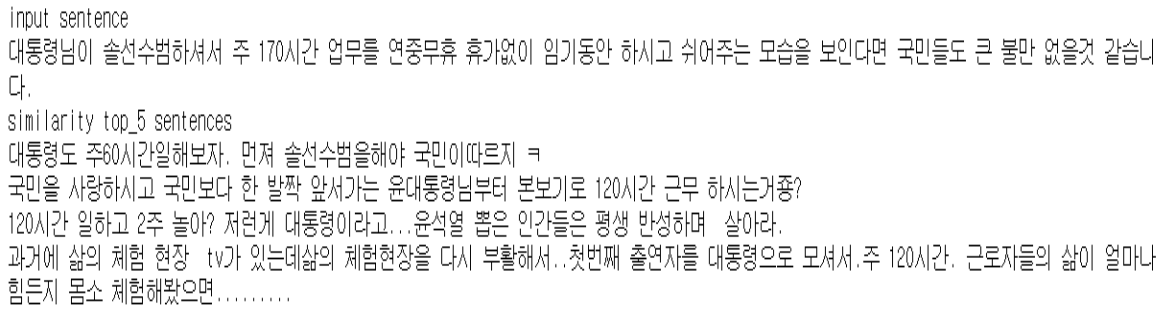

그림 1.10 분류 결과(2)

코사인 유사도를 통해 확인한 문장의 임베딩 상태는 유사한 단어가 포함된 문장은 잘 출력되는 모습을 보였지만 긍정, 부정, 분노, 슬픔 등의 감성의 유사도를 분류할 정도로 높은 유사도를 보이지는 않았다.

4.2.3 k-means clustering

해당 테스트부터는 긍정, 부정, 스팸으로 직접 분류해서 정리해 놓은 테스트 데이터를 사용해서 테스트했다. 테스트 데이터는 110개의 댓글로 이루어져 있으며 그 중 50개는 긍정의 댓글이고, 각각 30개는 부정, 스팸 댓글로 이루어져 있다. 실제 영상의 댓글을 수집하면 90%이상의 댓글이 긍정 댓글인 것을 확인할 수 있었는데 해당 상황을 반영해주기 위해 긍정 댓글의 개수를 더 높게 설정했다.

우선 아래 그림 1.11은 pre-trained word2vec 모델을 기반으로 한 클러스터링 결과이다.

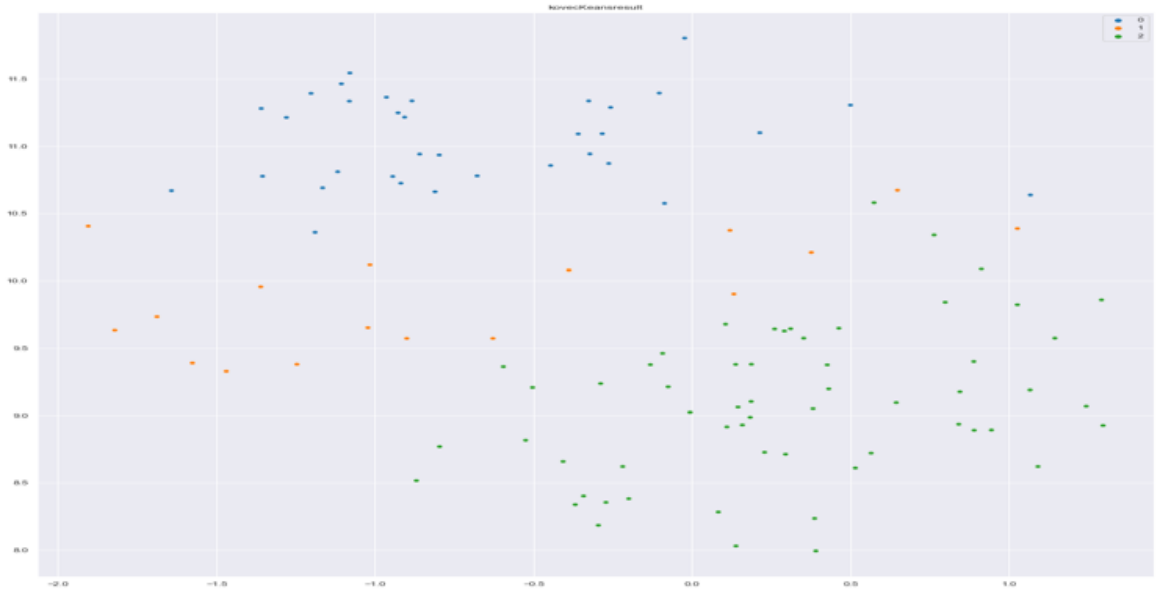

그림 1.11 K-means clustering (pre-trained word2vec)

색으로 구분된 결과는 긍정, 부정, 스팸으로 나누어진 데이터를 뜻하는 것이 아닌 클러스터링 결과로 군집된 집단을 나타낸다. 그림을 살펴보면 각 점들 간의 뚜렷한 군집이 보이지 않아 결과를 확인하기 쉽지 않았다.

해당 방식으로는 결과를 확인하기 힘들 것이라 판단되어 Tensorboard의 Embedding Projector 기능을 사용하여 임베딩 문장 벡터들을 tsv파일로 저장하여 PCA, T-SNE 차원축소를 통해 군집을 확인해 보았다.

4.2.3 PCA, T-SNE 시각화

차원 축소를 통한 시각화 방법 중 PCA와 T-SNE 방식을 사용하여 군집을 확인해 보았다. PCA 방식은 200차원의 벡터값 중 분산이 가장 큰 차원의 벡터를 골라 그 차원을 기준으로 투영시키는 방법이다. 투영시키는 과정에서 군집 데이터가 뭉개지는 단점이 있다. 아래의 그림 1.12는 테스트 데이터를 pre-trained word2vec 모델로 임베딩

결과를 PCA방식을 통해 3개의 축을 기준으로 압축시킨 결과이다.

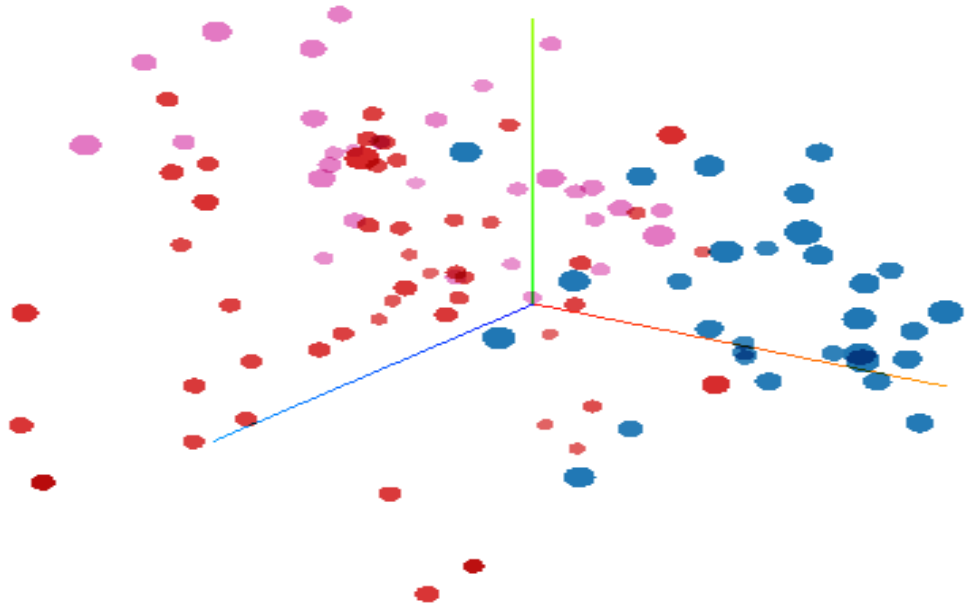

그림 1.12 PCA 시각화 (pre-trained word2vec, 3component)

각각 붉은색점은 긍정, 자주색 점은 부정, 그리고 푸른색 점은 스팸 데이터를 나타낸다. 푸른색의 스팸과 나머지 데이터는 명확하게 군집이 구분이 되었지만 긍정, 부정 데이터들은 서로 겹치는 부분이 존재하여 명확하게 분류하는 경계선을 찾기 힘들었다. 축을 2개로 감소시키면 더욱 명확하게 결과를 확인할 수 있었다.

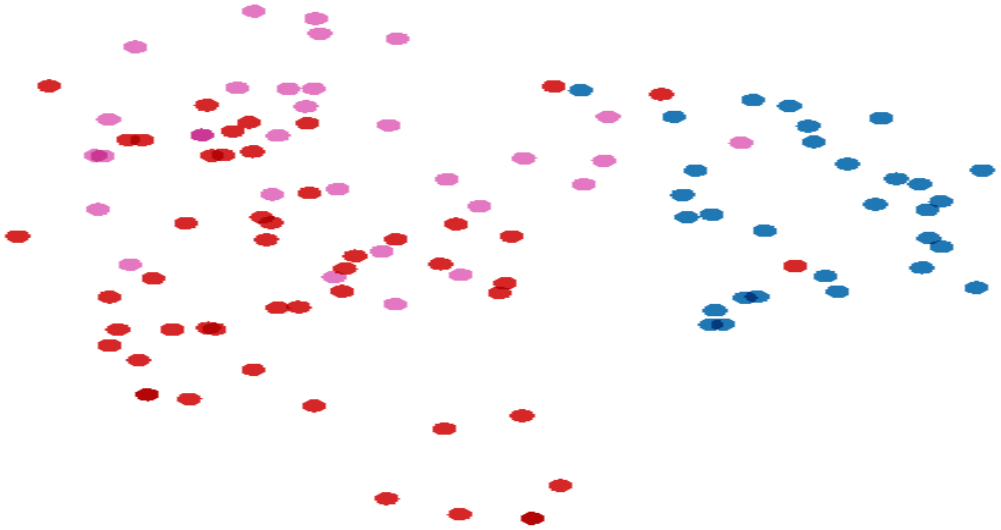

그림 1.13 PCA 시각화 (pre-trained word2vec, 2component)

아래의 그림 1.14, 15, 16은 각각 pre-trained wor2vec, word2vec, faasttext 모델로 분류한 테스트 데이터를 T-SNE 방식으로 차원을 축소하여 시각화 한 것이다. T-SNE 방식은 차원을 축소하면서도 고차원의 벡터에서 가지고 있던 유사성을 저차원에서도 보존해주는 특징이 있다. 우선 그림 1.14는 PCA 분석에서 살펴볼 수 있었던 것처럼 푸른색의 스팸 댓글은 다른 댓글과 분리되어 있는 모습을 확인할 수 있었고 긍정과 부정 댓글은 위와 아래로 나름의 분류된 경향을 보이긴 하나 극단적인 지점을 제외하고는 서로 뒤섞인 모습을 확인할 수 있었다. 분류의 사용하는 모델이 단어의 의미를 중점에 두고 등장 순서를 고려하지 않기 때문에 모델의 한계 상 감성분석에서 명확히 분류하지 못하는 경우가 존재한다고 생각된다.

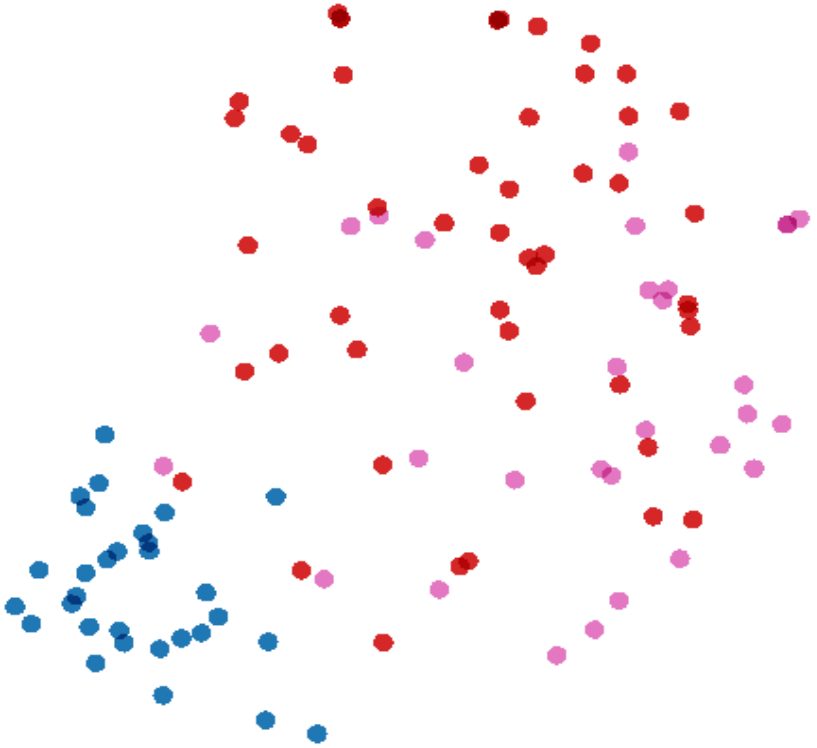

그림 1.14 T-SNE 시각화(pre-trained word2vec)

아래 그림 1.15는 약 10만개의 데이터로 직접 학습시킨 word2vec 모델을 사용하여 분류한 테스트 데이터를 T-SNE 방식으로 시각화 한 것이다. 테스트에 관련된 데이터로 학습을 시켰기 때문에 더 나은 분류 결과를 보일 수 있을 것이라 기대했지만 적절히 분류되지 않았다. 초기에 계획했던 학습 데이터의 양보다 더 많은 데이터가 필요한 것으로 판단되고 댓글 데이터가 해당 영상에 관련된 단어가 주로 나오다 보니 오히려 편향을 주어서 해당 단어의 뜻을 나타내는데 문제가 되었을 것이라 생각된다.

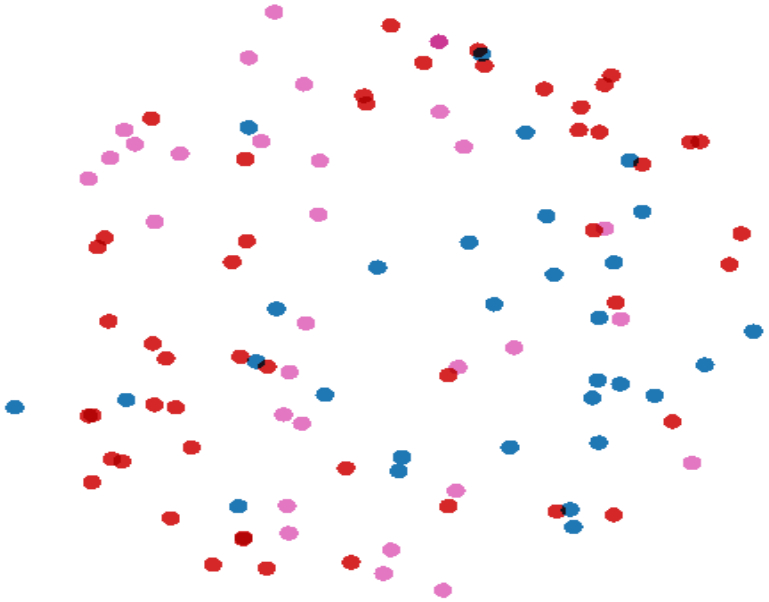

그림 1.15 T-SNE 시각화(word2vec)

마지막으로 그림 1.16은 약 5만여개의 댓글데이터로 학습시킨 fasttext 모델로 테스트 데이터를 분류한 결과를 T-SNE 분석을 통해 나타낸 것이다. Fasttext 모델은 하나의 단어를 지정된 수 만큼 분할하여 학습하기 때문에 상대적으로 적은 데이터 양에도 word2vec 모델보다 학습이 잘 이루어 졌다고 생각하고 pre-trained 모델만큼은 아니지만 스팸 댓글과 다른 댓글은 구분된 모습을 보였다.

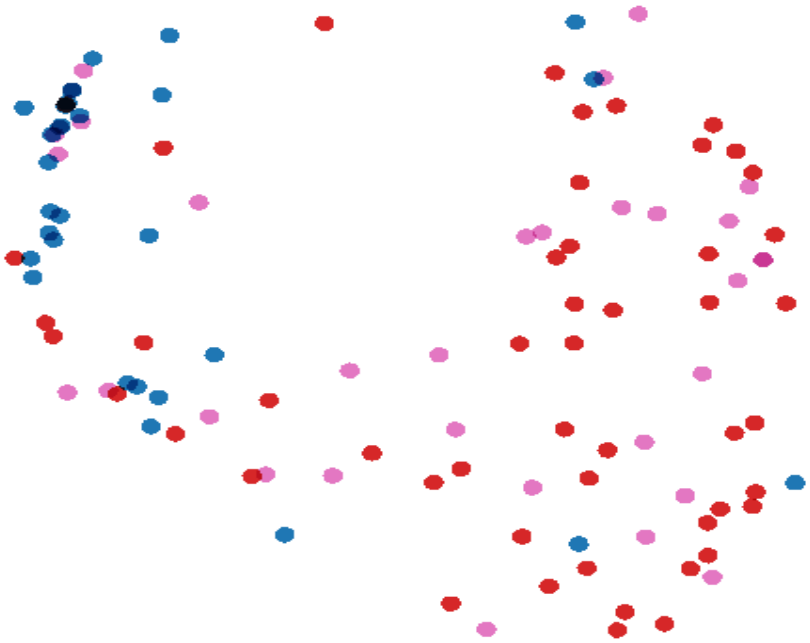

그림 1.16 T-SNE 시각화(fasttext)

5. 결론 및 향후 계획

5.1. 결론

초기에 계획했던 것처럼 명확한 분류의 결과를 얻지는 못했다. 하지만 같은 양의 데이터로 학습시킨 word2vec모델과 fasttext 모델을 비교했을 때 상대적으로 fasttext 모델의 성능이 더 높다는 것을 확인할 수 있었다. 감성 분석은 성공적으로 이루어지지 않았지만 pre-trained word2vec 모델을 사용한 경우 댓글의 스팸여부를 구분하는 것 과 유사 문장을 찾아내는 것에는 좋은 결과를 보였다. 테스트 데이터와 관련된 데이터를 학습에 활용하는 것은 생각했던 것처럼 효과적이지는 않았으며 학습의 목표가 단어의 의미를 continuous vector 값으로 임베딩하는 것 이기 때문에 오히려 단어들이 사용되는 다양한 상황이 고려되는 학습 데이터를 사용하는 것이 자연어처리에 더 도움이 될 것이라는 것을 알았다.

5.2. 향후 계획

데이터를 추가로 수집하고 Sentence embedding에서 가중치를 활용하는 방법을 구현한다면 자세한 감성분류는 힘들어도 내용과 관련이 없는 스팸을 찾아내는 기능과 유사한 문장을 찾아내는 기능은 구현할 수 있을 것 같다. 한국어 word2vec 학습을 위한 데이터를 추가로 수집하고 Sentence embedding에서 가중치를 적용하는 방법을 적용하면 정확도를 개선할 수 있을 것이라 기대하고 있다. 추 후 Glove 방식이나 Fasttext 방식으로 word embedding 모델을 학습시키고 가중치를 적용하여 분류하는 시도를 해 보려 한다.

6. 참고자료

Selenium과 BeatifulSoup를 사용한 유튜브 댓글 크롤링
https://velog.io/@kjh1337/Selenium%EA%B3%BC-BeatifulSoup%EB%A5%BC-%EC%82%AC%EC%9A%A9%ED%95%9C-%EC%9C%A0%ED%8A%9C%EB%B8%8C-%EB%8C%93%EA%B8%80-%ED%81%AC%EB%A1%A4%EB%A7%81

NLP 튜토리얼: 라벨링 없이 트위터 유저들을 자동으로 나누어보기
https://beomi.github.io/2020/01/05/Clustering_Twitter_Users/

자연어처리 기술을 활용한 유튜브 악성 댓글 자동 블라인드 시스템
https://www.dbpia.co.kr/pdf/pdfView.do?nodeId=NODE11043899&googleIPSandBox=false&mark=0&ipRange=false&accessgl=Y&language=ko_KR&hasTopBanner=true

단순하지만 강력한 Smooth Inverse Frequency 문장 임베딩 기법
https://bab2min.tistory.com/631

pre-trained ko_word2vec 모델
https://monetd.github.io/python/nlp/Word-Embedding-Word2Vec-%EC%8B%A4%EC%8A%B5/

[논문 리뷰] Transformer _ 워드 임베딩(Word Embedding), 어텐션, 셀프 어텐션(Self-Attention) 이해하기
단순하지만 강력한 Smooth Inverse Frequency 문장 임베딩 기법
https://mingchin.tistory.com/307
[NLP] 단어부터 문장까지 GloVe Embedding / Clustering — 지니티토리
자연어 처리 기술을 이용한 감성분석 기법에 관한 연구(윤태성 연세대학교 공학대학원 컴퓨터공학 전공 국내석사) : https://scienceon.kisti.re.kr/srch/selectPORSrchArticle.do?cn=DIKO0015302535