'

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 1.1 을 살펴보면 13세이상 60세 미만의 연령대에서 90% 이상 이 모바일 메신저와 유튜브를 사용하고 있다고 답했다. 이러한 플랫폼에서 생산되는 댓글, 반응 들의 데이터는 인간이 직접 분류하며 이용하기에는 한계가 있다.

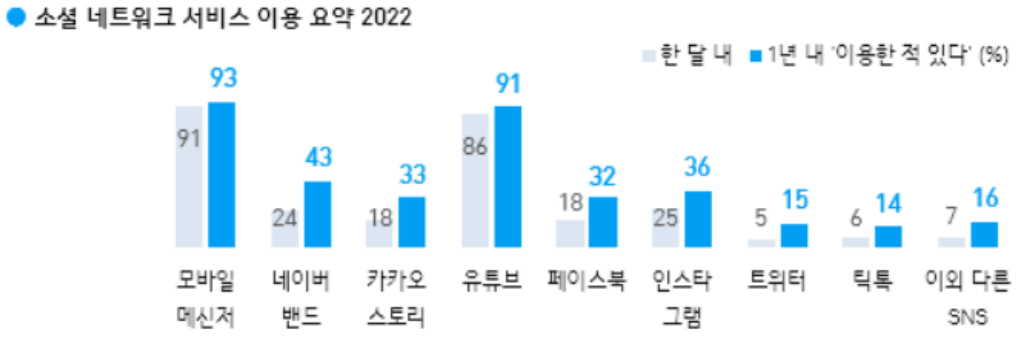

그림 1.1 소셜 네트워크 서비스 이용 요약 2022

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 가치를 찾아내는 좋은 경험일 것이라 생각한다.

2.2. LLM(Large Language Model)의 등장

자연어를 이해하고 생성할 수 있는 GPT-3와 같은 LLM들이 경쟁적으로 생겨나고 있다. 자연어를 다루는 여러가지 기술이 강세라는 것은 그만큼 자연어를 분석하고 의미 있는 정보를 뽑아내어 사용하는 것이 중요하다고 할 수 있다. 파라미터의 개수나 데이터의 양으로는

LLM의 성능을 따라잡을 수 없지만 범용적으로 설계되어 있는 LLM과는 달리 단순 문장의 분류에만 초점을 맞추어 모델을 설계한다면 유의미한 성능을 얻을 수 있을 것이라 생각한다.

3. 설계 및 구현

3.1 프로젝트 일정

구분	추진내용	프로젝트 기간					
		7월	8월	9월	10월	11월	12월
데이터 수집	약 20만개까지 수집 예정						
모델 설계	Word embedding 모델 설계						
	SLF 가중치 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

Selenium프레임 워크를 사용하여 사이트 URL을 입력하면 해당 영상의 댓글 데이터를 수집한다. 로딩 조건의 한계 상 약 1000개의 한국어 문장 데이터셋을 수집할 수 있으며 학습의 효율을 높이기 위해

여러가지 영상의 데이터를 수집할 것이다.

3.2.2 분류 모델 구현

문장을 분류하기 위한 모델은 Sentence embedding 모델 방식을 사용할 것이며 embedding 값은 문장을 구성하는 word의 embedding vector 값에 각 단어가 등장할 확률을 고려한 가중치를 곱하는 SLF 방식을 사용하여 해당 문장의 embedding vector값을 계산 할 것이다. Word embedding 모델은 word2vec 모델을 사용해 보았으나 문장의 특성을 잘 반영하지 못해서 python gensim 라이브러리를 이용해 Fasttext 방식의 모델을 학습해서 사용하고 Glove 방식의 모델도 구현한 후 결과를 비교해 보려 한다.

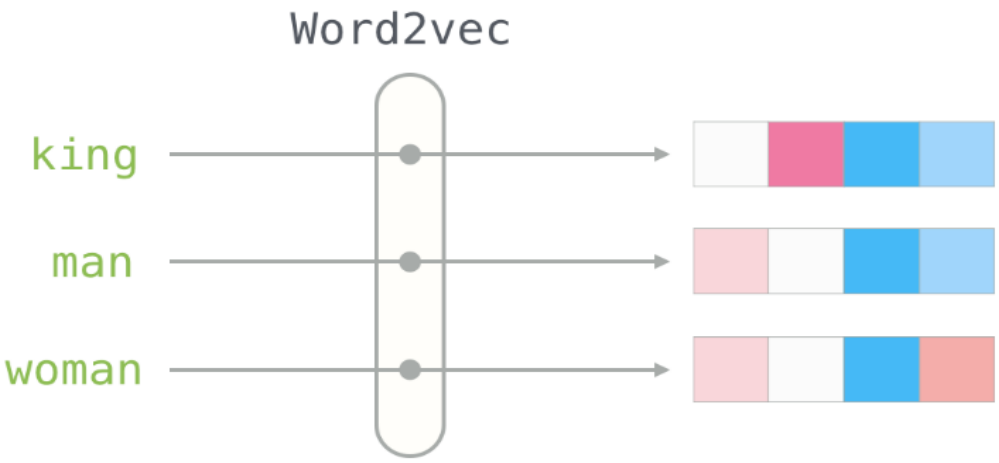

그림 1.3 Word2vec 모델

Word2vec 모델은 word를 embedding하는 방법 중 하나로 단어의 특성을 벡터화 하여 부여하고 유사한 특성을 지닌 단어들이 같은 벡터 방향을 띄게 하는 방식으로 구현된다. 단점으로는 단어 사전에 없는, train 단계에서 학습하지 않은 단어의 경우에는 벡터값을 정의할 수 없고 문장의 전체적인 의미는 반영하지 못하고 window 파라미터 범위의 주변 단어의 영향을 받아 embedding 된다.

해당 문제를 개선하기 위해 사전에 없는 단어도 embedding 값을 계산 할 수 있는 Fasttext모델을 사용하고, 문장의 의미를 반영하는 데 더 효과적이라는 Glove 모델을 구현하려 한다.

Sentence embedding의 경우 word2vec를 사용했을 때는 문장을 구성하는 단어의 평균값으로 계산했다. 해당 방법은 문장을 구성하는 각 단어가 의미적으로 같은 가중치를 가진다고 가정한다. 하지만 실제로는 문장 안에서 전체의 의미를 대표하는 단어가 존재하고 이러한 것을 고려하지 않으면 단어가 많은 문장일수록 중요한 의미를 가지고 있는 단어의 임베딩 값이 희석되는 문제가 있다. 해당 문제를 해결하기 위해 단어의 등장확률과 관련된 가중치를 곱해 값을 구하는 SLF(smooth inverse frequency) 방식을 사용하려 한다.

3.2.3 분류결과 시각화

그림 1.4는 pre-trained word2vec 모델로 embedding한 문장들을 k-means clustering을 통해 분류해 본 결과를 나타낸 것이다. Fasttext, Glove 방식의 분류가 이루어지면 그림1.4와 같이 분류해보고 결과를 비교해 볼 것이다.

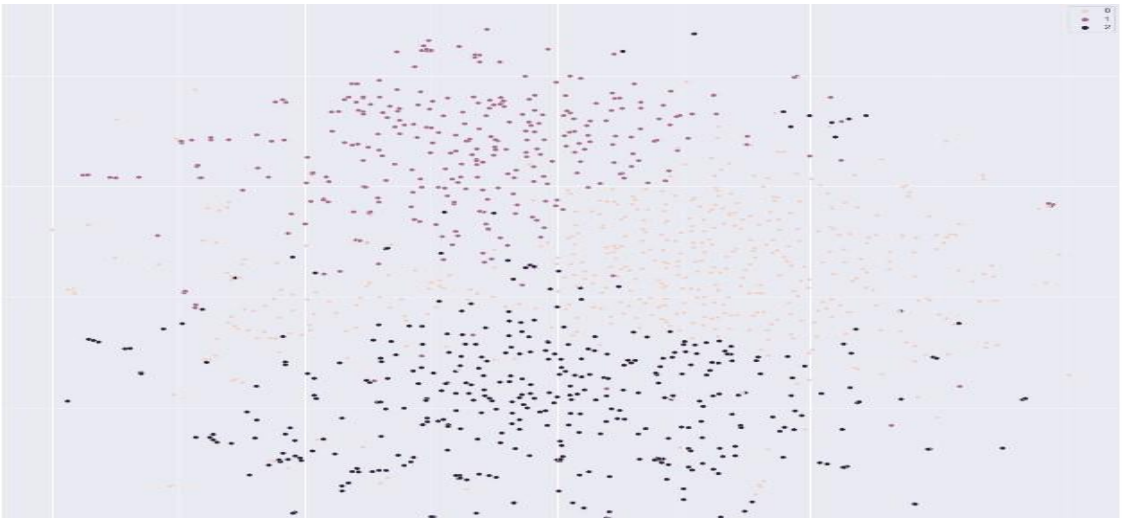

그림 1.4 k=3 k-means clustering 예시

3.2.4 전체 동작

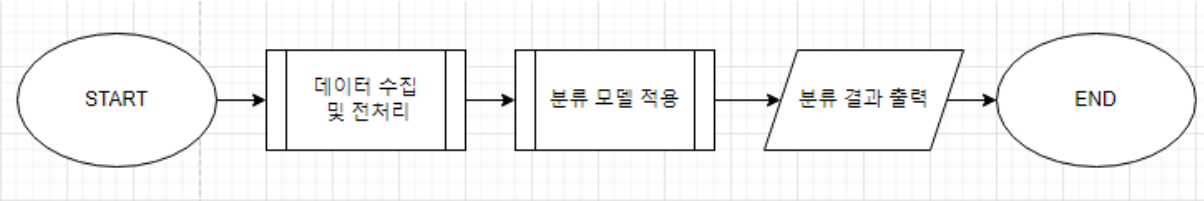

그림 1.5 전체 동작 흐름도

그림 1.5는 프로젝트의 전체 동작을 간단하게 나타낸 흐름도이다. 먼저 데이터를 수집하고, 단어 단위로 전처리한 후 word embedding model을 통해 얻은 vector값을 가중치를 곱해 계산하여 문장을 나타내는 continuous vector를 구한다. 이 후 k-means clustering, 스팸 데이터 라벨링 후 지도학습 등의 방식으로 분류를 진행하고 결과를 평가한다.

4. 실행 결과

4.1. 데이터 수집 및 전처리

	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 갸날픈 손꾸락까지 완벽하다

그림 1.5 수집 데이터 예시

그림 1.5는 수집된 데이터를 출력한 결과이다. 유튜브 영상 하나를 기준으로 약 1000개의 데이터를 수집하였으며 URL마다 따로 엑셀 파일의 형태로 저장했다. 추 후 word2vec 학습 과정에서 사용하기 위해 데이터가 충분히 쌓이면 train_data_set을 구성하여 한 곳으로 병합 할 계획이다. 약 15000개의 댓글 데이터가 쌓여 word embedding model을 학습시켜보았다. 유튜브 댓글의 특성 상 길이가 짧은 데이터가 다수를 차지하고 있어서 성공적인 모델을 위해서는 더 많은 양이 필요했다. 10만개정도의 데이터가 쌓이면 다시 시도해 보려 한다.

데이터 전처리의 경우 파이썬에서 제공하는 한국어 형태소 분석 패키지 Konlpy를 사용했다. 영어에서 잘 적용되었던 불용어 처리의 경우 한국어의 어간, 어미의 특징 때문인지 잘 적용되지 않아 정규식을 사용해 어간 불용어, 어미 불용어를 구분해서 처리해 보려 한다. 기본적으로 Word2vec 모델을 사용할 것 이기 때문에 조사, 문장부호, 영어, 숫자 등을 모두 제거하고 명사 형태의 단어만 남도록 전처리 했다.

4.2. 모델 분류 결과

데이터를 수집하고 pre-trained word2vec 모델을 사용해서 분류를 진행했다. 그림 1.6은 문장의 유사도를 측정한 방식으로 sentence embedding이후 vector값의 코사인 유사도를 측정하여 유사한 문장 간의 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 1.6 문장 유사도 측정 방식

입력 문장을 기준으로 가장 높은 유사도를 가진 댓글 5개를 출력하는 방식으로 분류 결과를 확인했다.

그림 1.7 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 1.7 분류 결과(1)

또한 그림 1.8의 분류결과(2)에서는 대통령, 솔선수범 등의 단어가 핵심인 문장을 입력 값으로 넣었을 때 출력된 문장들 모두 비슷한 의미를 지닌 문장들이었다.

input sentence
대통령님이 솔선수범하셔서 주 170시간 업무를 연중무휴 휴가없이 임기동안 하시고 쉬어주는 모습을 보인다면 국민들도 큰 불만 없을것 같습니다.
similarity top_5 sentences
대통령도 주60시간일해보자. 먼져 솔선수범을해야 국민이따르지 ㅋ
국민을 사랑하시고 국민보다 한 발짝 앞서가는 윤대통령님부터 본보기로 120시간 근무 하시는거죵?
120시간 일하고 2주 놀아? 저런게 대통령이라고...윤석열 뽑은 인간들은 평생 반성하며 살아라.
과거에 삶의 체험 현장 tv가 있는데삶의 체험현장을 다시 부활해서..첫번째 출연자를 대통령으로 모셔서.주 120시간. 근로자들의 삶이 얼마나 힘든지 몸소 체험해봤으면.....

그림 1.8 분류 결과(2)

5. 개선 사항 및 향후 계획

5.1. 데이터 수집 및 전처리

분류 결과를 확인하던 도중 수집한 댓글들의 의견이 다양하게 분포되어 있지 않고 편향되어 있다는 사실을 알게 되었다. 비지도 방식으로 분류를 진행하므로 결과를 분명하게 구분하기 위해서는 다양한 의견의 데이터 셋을 수집하여야 한다. 그리고 데이터 전처리 과정에서 불용어 처리를 활용하는 방법에 정규식 표현을 활용하여 의미를 가지고 있는 어간, 어미는 삭제하지 않고 불용어 처리를 하는 방식을 구현해야 한다.

5.2. 분류 모델 개선

pre-trained word2vec 모델을 분류의 핵심 모델로 사용했다. 아주 터무니없는 결과를 얻은 것은 아니지만 단어사전에 없는 단어가 댓글에 포함되어 있는 경우가 종종 있어서 문제가 되었다. 데이터 수집과 전처리가 충분히 완료되면 사전에 없는 단어도 값을 구할 수있는 Fasttext 방식의 모델을 학습시켜서 사용해보고 Glove 방식의 모델 또한 구현하여 두 방식의 결과를 비교해 보려 한다. 그리고 Sentence embedding에서 단순 word embedding 값의 평균치를 구

하는 방식이 아닌 등장확률과 관련된 가중치를 곱해 문장의 의미를 더욱 반영할 수 있는 방식으로 구현하겠다.

5.3. 분류 결과 적용 아이디어

설계한 모델이 긍정, 부정, 유사도 등의 기준으로 문장을 나누는 것이 성공한다면 문장을 입력했을 때 비슷한 의견의 문장과 반대 의견의 문장을 출력하도록 구현하고, 광고성 스팸 댓글에 라벨링을 하고 분류의 정확도를 측정해 볼 것이다.

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