악플 분류 및 순화 문장 재생성 프로젝트

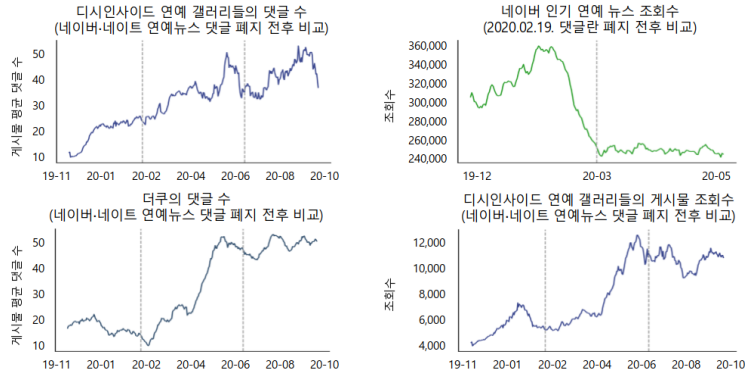
NLP-06 HAPPY

김준휘, 류재환, 박수현, 박승현, 설유민

1. Introduction

1-1. 문제 정의

악성 댓글(이하 악플)은 포털 사이트가 활성화된 이후로 꾸준히 사회적인 문제로 대두되고 있다. 이에 따라 2020년경부터 대부분의 포털 사이트에서 연예/스포츠 뉴스에 대한 댓글란을 폐지하여 악플을 방지하고자 했다. 그러나, 2020년경 네이버 연예뉴스 조회수와 타 커뮤니티의 활성화 정도를 비교했을 때(아래 그림) 댓글 서비스 폐지와 맞물려 많은 관련 커뮤니티의 조회수 및 댓글수가 상승했다. 포털은 이용자 및 매출 감소를 감수하고 사회적 책임을 다하기 위해 댓글 서비스를 폐지했으나 댓글 서비스에 대한 수요는 사라지지 않았음을 볼 수 있다.



댓글/ 악플 타 커뮤니티로 분산, 음성화　　포털 연예뉴스 조회수 20% 감소

그림 1 네이버 연예뉴스 댓글 폐지에 따른 커뮤니티 활성화 정도 및 네이버 이용자 감소

따라서 본 프로젝트에서는 댓글 서비스를 유지하되 악성 댓글의 의미를 가능한 유지하되, 순화한 표현으로 대체하는 방법을 연구하여 서비스에 적용하였다. 빠르게 변화하며 다양한 의미를 내포할 수 있는 언어의 특성상 딥러닝 모델을 적용하는 것이 적절할 것으로 판단했다. 이를 구현하기 위해 악플과 정상적인 댓글을 분류하기 위한 classification model과, 악플로 분류된 댓글에 대해 의미를 유지하며 순화된 표현을 재생성하는 generation model을 구현하여 서비스에 적용하고자 하였다.

1-2. 역할 분담

김준휘 : 혐오 표현 데이터셋 분석. 분류 모델 설계와 구현. 분류 모델 API 제작.
류재환 : 생성 모델 설계와 구현. 생성 모델 API 제작.
박수현 : 분류 모델 설계와 구현. 데이터 수집.
박승현 : 생성 모델 설계와 구현. 데이터 수집. 웹서비스 설계와 구현.
설유민 : 생성 모델 설계와 구현. 데이터 수집.

1-3. 기존 연구

현재 대형 포털(네이버, 카카오) 에서는 댓글 서비스를 폐지한 연예/스포츠를 제외한 뉴스, 웹툰 등 댓글 서비스에서 댓글 필터링 서비스를 운영하고 있다. 네이버의 댓글 필터링 서비스인 네이버 클린봇은 혐오표현을 감지하여 댓글 전체를 차단하는 기능을 수행하며, 카카오의 세이프봇은 혐오표현으로 분류된 부분을 타 기호로 치환하는 기능을 수행한다. 그러나 이 방식에도 한계점이 존재한다. 네이버 클린봇은 혐오표현으로 분류된 경우 댓글 전체를 차단하며 이에 대한 근거를 제시하지 않기 때문에 의견을 잘못 필터링했을 경우 서비스 신뢰도에 치명적인 영향을 미친다. 카카오 세이프봇은 혐오표현으로 분류된 부분만 치환하는 기능을 수행하기 때문에 단순 혐오표현 치환으로 해결할 수 없는 문장 자체가 혐오 및 편향성을 드러내는 경우에 대해서는 대응이 불가하다.

그림 2 현재 사용되는 댓글 필터링 서비스

1-4. 해결 방안 및 기대 효과

본 서비스에서는 댓글 등록 전 사용자에게 필터링 결과를 알려준다. 사용자는 댓글 등록 전 댓글에 혐오표현의 소지가 있는지 안내받는 동시에 의미를 유지하며 재생성된 순화 표현을 제시 받는다. 사용자는 이를 보고 경각심을 가질 수 있으며 교체한 표현으로 댓글을 달 수 있다.

이를 통해 기분 좋은 온라인 커뮤니티를 만드는 동시에 기존 한국어 댓글 데이터를 수집한 연구에서 사용되지 못한 혐오 표현들을 순화하여 다양한 데이터의 구축에 기여한다.

그림 3 서비스 화면 예시

2. Classification Model

2-1. 데이터셋 선정

직접 혐오 발언의 기준을 세우고 데이터를 구축했다면 더 좋았겠지만, 프로젝트 기간 상 기존의 데이터셋을 사용할 수밖에 없었다. 공개되어 있는 한국어 혐오 표현 데이터셋 5종을 수집하였다. 사용한 데이터셋에 대한 간략한 정보는 아래 표 1에 정리되어 있다.

표 1 사용한 한국어 혐오표현 데이터셋 정보

데이터셋	Apeach(링크)	Beep!(링크)	Unsmile(링크)	K-MHaS(링크)	KOLD(링크)
원천	사람들이 직접 데이터셋을 작성.	2018년1월 ~2020년2월까지의 연예 뉴스 댓글	2019년 1월 ~ 2020년 6월까지의 뉴스 댓글, 커뮤니티 댓글 수집.	2018년 1월 ~ 2020년 6월 사이에 수집된 뉴스 댓글들로 구성.	2020년 3월 ~ 2022년 3월 사이의 네이버 뉴스 댓글과 유튜브 영상 댓글들 크롤링.
혐오	1,922	4,410	12,636	47,060	20,310
비혐오	1,848	3,486	3,739	42,909	20,119

필터링 서비스에서 가장 중요한 것은 **혐오 표현이 존재하지 않는 문장을 혐오 표현으로 오판하지 않는 것이다**. 즉, recall 수치가 중요하다고 판단했다. 이를 측정하기 위해 5개의 데이터셋으로 각각 학습한 뒤, 각 데이터셋의 평가셋으로 모델을 평가하여, 가장 recall 수치가 높게 나타나는 데이터셋을 선정하기로 결정하였다.

표 2 Recall 수치 평가표

		Training Set				
		BEEP!	K-MHaS	KOLD	UnSmile	Apeach
Dev Set	APEACH	73.62	92.26	81.74	78.02	88.00
	BEEP!	93.22	100	93.51	92.38	91.54
	K-MHaS	58.20	89.84	60.53	64.08	60.36
	KOLD	77.08	90.90	82.23	74.55	79.30
	UnSmile	86.67	97.73	88.20	91.52	86.81
	평균	77.75	**94.14**	81.24	80.11	81.20

표 3 Precision 수치 평가표

		Training Set				
		BEEP!	K-MHaS	KOLD	UnSmile	APEACH
Dev Set	APEACH	92.81	65.14	89.95	85.90	86.38
	BEEP!	75.24	22.18	64.95	62.37	59.16
	K-MHaS	97.35	**87.55**	95.47	94.52	86.57
	KOLD	79.37	33.71	81.84	79.17	67.56
	UnSmile	90.47	64.77	88.86	90.97	81.33
	평균	87.04	54.67	84.21	82.58	76.20

Recall 수치를 측정해본 결과, K-MHaS 데이터셋으로 학습한 결과가 가장 높게 나타났다. Precision 수치도 측정해본 결과, K-MHaS 데이터셋이 다른 데이터셋들에 비해 수치가 많이 낮다. 이는 데이터셋들의 라벨 기준이 모두 다르기 때문에 나타난 결과로 보인다. 예시를 들어 설명하자면,

"동성애... 이해할 수 없다 제발 정신차려라..."

위 문장을 K-MHaS로 학습된 모델은 '비혐오'로 판별하고, KOLD로 학습된 모델은 '혐오'로 판별한다. 위 문장의 경우 경우에 따라 동성애를 좋아하지 않는다는 의견의 표시로도, 혹은 동성애에 대한 편견이 들어간 문장으로도 보는 사람에 따라 다르게 받아들일 수 있다. 우리는, 이런 **개인의 의견으로 생각될 수 있는 문장들은 최대한 필터링 하지 않기를 바란다**. K-MHaS 데이터셋은 이렇게 혐오로 판별하는 기준이 높은 것을 확인할 수 있고, 우리의 의도에 부합하는 데이터셋이라고 판단할 수 있었다.

2-2. Recall 수치 평가

K-MHaS 로 학습된 모델의 Recall 수치가 다른 데이터셋들에 비해 낮게 나타나지만 그렇다 하더라도 오판을 내릴 수는 있다. 우리는 모델의 오판을 최대한 줄이기 위해 추가적으로 recall 수치 검증을 하기로 했다.

각 데이터셋들로부터 누구에게나 이건 혐오 발언이 아니다 라는 공감을 이끌어낼 수 있을 만한 문장들을 40 개씩 총 200 개를 추출하여 recall 평가셋으로 구성하였다. 이 200 개의 문장을 모두 '비혐오'로 판별한다면, 우리가 생각하는 최소한의 기준을 만족했다고 평가할 것이다.

[기준]

- 욕설, 혐오의 의미를 갖는 단어를 사용하지 않았다.
- 개인의 주관, 편견 등이 포함되지 않았다.
- 말투가 정중하고 남을 기분 나쁘게 하지 않는다.

K-MHaS 로 학습된 모델은 200 개의 문장 중 아래의 한 개의 문장을 제외하고 모두 비혐오로 예측하였다.

"사람들이 나 얼굴 못생겼으니까 얼굴 다 성형하래."

2-3. 모델 선정

2-3-1. CNN 기반 모델

- 네이버 클린봇의 경우 학습 용이성이나 추론 속도를 고려하여 CNN 모델을 사용했다. 이를 참고해 Attention 기반이 아닌 CNN 모델도 실험해 보기로 결정했다.

- ResNet 을 참고해 residual 구조로 간단하게 구성했다. 모델 도식은 아래 그림에서 확인할 수 있다.

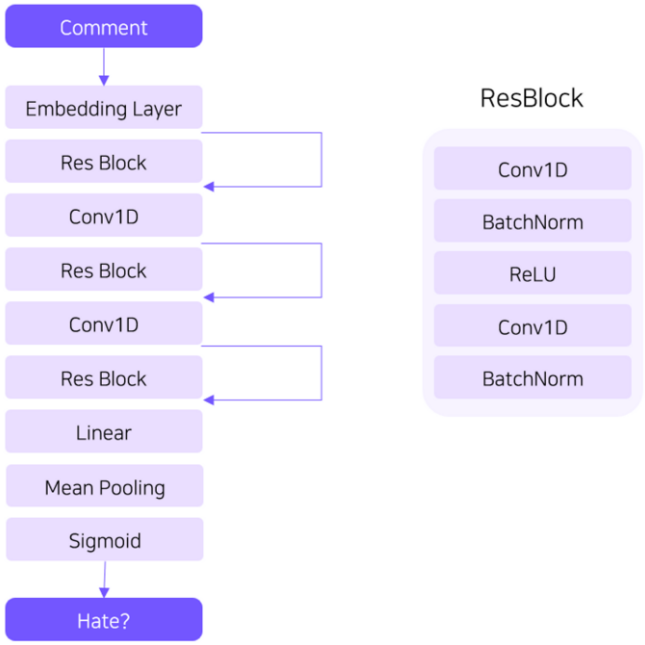


그림 4 CNN 모델 도식

2-3-2. Transformer 기반 모델

huggingface의 AutoModelForSequenceClassification을 이용해 간단하게 구현하였다. 베이스 모델로는 정제되지 않은 텍스트들로 학습된 **KoELECTRA**와 **KcELECTRA, KcBERT**를 중심으로 실험했다. 이는 우리가 **수집한 데이터셋들이 대부분 인터넷의 댓글들로 맞춤법, 띄어쓰기 등이 제대로 지켜지지 않은 정제되지 않은 텍스트**들 중심이기 때문이다. 실제로도 KLUE 등 정제된 데이터셋으로 학습된 모델들 보다 성능이 더 좋았다.

2-4. 실험 결과

	KcELECTRA-small	KcELECTRA-base	koelectra-small	koelectra-base	KcBERT-base	KcBERT-large	ResNet
f1 score	87.43	**90.88**	79.78	87.42	90.10	90.80	83.21

- K-MHaS 데이터셋으로 모델을 평가했다.
- KcELECTRA-base 의 f1 점수가 가장 높았다.

    - KcELECTRA 의 경우, 2022 년 버전으로 기존의 KcELECTRA, KcBERT 보다 훨씬 더 많은 비정제 데이터셋으로 학습되었기 때문에 본 task 에서 더 좋은 성능을 보인 것으로 보인다.
- AIstage 의 GPU 서버를 활용할 수 있는 환경이었기 때문에, V100 을 사용해 추론하기로 했다. Locust 를 활용해 RPS(Response Per Second)를 측정한 결과, KcELECTRA-base 모델 4 개를 띄웠을 때 170 까지 측정되었기 때문에 굳이 작은 모델을 사용하지 않아도 되겠다고 판단했다.

2-5. Token Classification

토큰 분류 모델 제작에는 KOLD 데이터셋의 혐오 단어 라벨링을 이용했다. KOLD 데이터셋의 형식은 아래와 같다.

"comment": "남녀평등 주장할 거면 여성징병제에도 동의하라고ㅋㅋㅋ 그리고 내 말에 그냥 시비만 걸지 말고 혜택은 다 쳐받으면서 왜 차별받는다고 말하는지 말해보라고ㅋㅋ"

"OFF_span": " 쳐받으면서 왜 차별받는다고 말하는지 말해보라고ㅋㅋㅋ"

"off_start_idx": [57]

"off_end_idx": [84]

위와 같은 라벨링을 이용해 문장을 토큰화 했을 때 각 토큰마다 해당 토큰이 혐오 표현을 담고 있는지 아닌지를 판별하는 식으로 모델을 학습했다. ODQA 의 reader 모델과 같이 시작 인덱스와 끝 인덱스를 예측하는 방법도 있지만 한 문장에 "OFF_span"이 여러개인 경우도 있고 해서 토큰 분류를 하기로 결정했다. KOLD 논문의 내용에 따라 '혐오 대상'을 같이 분류함으로써 토큰 분류의 정확도를 8% 향상시킬 수 있었다. '혐오 대상'은 KOLD 의 Level B. 의 라벨과 Level C. 를 대분류로 나누어 ["individual", "untargeted", "other", "race", "politics", "gender", "religion", "sexual_orientation", "others"] 총 9 개의 라벨로 분류하였다.

혐오 글자 라벨링은 혐오 표현으로 분류된 문장들에만 되어 있기 때문에 비혐오 표현을 같이 학습하면 라벨 불균형으로 문제로 모델이 대부분의 토큰을 비혐오로 분류하게 된다. 따라서 혐오 표현들만 가지고 학습을 진행해야 했다.

토큰 분류 모델을 학습할 때는 Recall 수치를 크게 고려하지 않았다. 그 이유는 분류 모델이 '혐오'로 분류한 문장들에 대해서만 결과를 보여줄 것이기 때문에 recall 수치가 크게 중요하지 않다 판단했기 때문이다. 또한 혐오 토큰이 라벨링된 데이터셋은 KOLD 데이터셋밖에 없어 다른 선택지가 존재하지 않았다.

3. Data for Generation

혐오 표현 데이터셋은 많지만, 혐오 표현에 대응하는 순화 문장으로 이루어진 한국어 데이터셋은 존재하지 않았다. 데이터셋을 구축하는 데에는 많은 작업이 필요할 것이라고 판단하여, parallel dataset이 없어도 순화를 할 수 있는 방법을 적용하고자 시도했다. 시도한 방법들에 대한 한계점을 파악하고, 효율적으로 데이터를 수집하고자 했다.

3-1. Non-parallel(1) : Task Reformulation

우리 task를 혐오 문장-순화 문장 parallel dataset이 없어도 학습할 수 있도록, 두 단계의 paraphrasing task로 재정의했다.

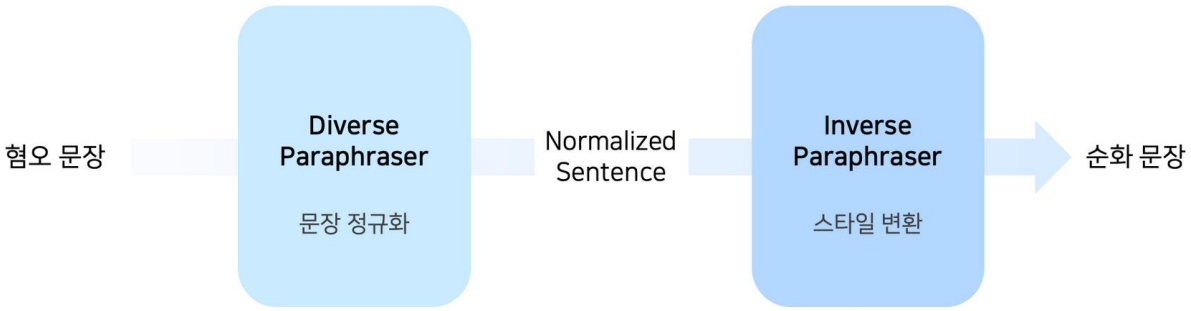


그림 5 STRAP 전체 프로세스. Diverse paraphraser는 어떤 style을 가진 문장을 정규화하는 paraphraser. Inverse paraphraser는 정규화된 문장에 원하는 style을 입힌 문장을 생성하는 paraphraser.

해당 방법론의 전체적인 학습 과정은 아래와 같다:

1. 일반적인 paraphrasing dataset 으로 diverse paraphraser 를 학습시킨다. Diverse paraphraser 는 어떠한 style 의 문장이 제시되더라도 알맞게 정규화할 수 있도록, 일반화된 성능을 가지도록 학습된다.
2. 원본 문장과 diverse paraphraser 를 통해 생성된 정규화된 문장을 짝지어, parallel dataset 을 구성한다. 해당 데이터셋에서 입력을 정규화된 문장으로, 출력을 원본 문장으로 한다. Inverse paraphraser 는어떠한 style 도 적용되지 않은 일반적인 문장에 style 을 씌울 수 있도록 학습된다.

해당 방법론의 전체적인 추론 과정은 아래와 같다:

1. Diverse paraphraser 를 통해 혐오 문장을 정규화한다. 문장에서 혐오 요소를 제거하고 의미만을 유지한 문장으로 paraphrasing 한다.
2. Inverse paraphraser 를 통해 정규화된 문장을 순화 문장으로 paraphrasing 한다.

모델은 아래 표와 같은 설정을 사용했다.

표 4 STRAP model parameter setting

PLM	skt/ko-gpt-trinity-1.2B-v0.5
Epoch	1
Batch size	2
Max length	448
Learning rate	5e-5
Optimizer	AdamW
Scheduling	Linear

해당 방법론의 핵심적인 문제는 아래와 같다:

- Diverse paraphraser 의존도가 너무 높다. 해당 기법은 우선 diverse paraphraser 의 성능이 안정적이어야 한다는 제약 조건이 존재한다. Diverse paraphraser 에서 혐오 문장을 제대로 정규화하지 못하는 경우, 비혐오 문장으로 학습된 inverse paraphraser 에 정규화된 문장을 입력으로 제시한다고 해도 좋은 style transfer 결과를 기대하기 어려웠다.
- Diverse paraphraser 의 성능이 충분히 강력하지 못했다. Diverse paraphraser 를 학습시키기 위해 사용할 수 있는 한국어 paraphrasing dataset 은 국립국어원 유사 문장 말뭉치를 사용했다. 해당 데이터셋은 정제된 기사, 뉴스 및 강연 대사, 대본으로 이루어져 있다. 정제되지 않은 욕설을 인지해야 하는 우리 task 의 데이터셋과는 괴리가 컸다. 아래 표에서 결과 예시를 확인할 수 있다.
- 의미 유지가 필수적인 diverse paraphraser 가 원본과 일부 단어만 일치할뿐 관련이 없는 맥락의 문장을 생성했다.
- 시간 및 메모리 차원에서 매우 비효율적이다. 순화 수행 시 diverse paraphraser 와 inverse paraphraser, 총 두 모델을 순차적으로 사용해야 한다. 또한, inverse paraphraser 학습 시 diverse paraphraser 를 거쳐 나온 혐오 표현의 정규화된 문장을 데이터셋으로 사용해야 한다. 약 20,000 문장에 대한 정규화된 문장을 생성하기 위해서는 1 GPU(Tesla V100, 32GB) 기준 최소 11 시간이 소요될 것으로 예상되었다.

표 5 STRAP을 이용한 정규화 결과 예시. 문장의 의미를 유지하지 못하고 욕설을 그대로 생성하기도 함

원본 문장	정규화된 문장
나무위키는 개나 소나 다 수정해서 쓸 수 있는 사이트라 신빙성이 전혀 없는 사이트로 유명합니다	나무 위키(소나무, 사슴)'는 아프리카의 흑성성으로 유명한데요. 이곳에서는 사막을 볼 때 새까
사람을 칼로 찌른 짱깨를 동포라니 조선족이 문재인을 칼로 찔렀으면 중국동포라고 했을까	짱 깨문 성명 없는 살인사건을 저지른 범죄자들이 중국을 탓으로 삼았다'는 식으로 중국이 범죄를 저질렀다고 한다. 중국은 '살인범이 중국에 의해 저질러진 죄에 대한 사죄의 뜻'이라며 범행을 주도한 사람이 중국인이라고 했다. 그러면서 이 범행은 조선의족,

3-2. Non-parallel(2) : 유사 task 데이터셋 활용

위의 방법에서, 문장을 정규화하는 등 우회적인 방법을 사용하기에는 명확한 한계가 보였다. 혐오 문장으로부터 순화 문장으로 바꾸는, 직접적인 style transfer 를 수행할 필요가 있다고 판단했다.

데이터셋 내 일부 욕설이 포함된 문장의 경우, 쌍을 이루는 formal style 문장에서는 제거되었다. 혐오 표현을 순화하기 위해 구축된 데이터셋이 아니더라도, 해당 데이터셋을 활용해 모델이 우리 task 에서 원하는 욕설 제거를 간접적으로 학습할 수 있다고 판단했다.

학습 및 평가 데이터셋으로는 Smilegate AI 의 한국어 문체 스타일 변환 SmileStyle 데이터셋을 사용하였다.

표 6 SmileStype 예시. 혐오 문장으로는 chat, choding, halmae, joongding 스타일을 가정. 순화 문장으로는 formal style을 가정

chat/choding/halmae/joongding	formal
그래도 병신 안되서 다행이네 응급차는 불렀냐?	그래도 안 다치셔서 다행이네요. 응급차를 부르셨나요?
안녕하냐 —— 나 씹냥이 6 마리나 키운다 하;	안녕하세요. 저는 고양이 6 마리 키워요.
뭔 병신같은 장식을 생각하고 있길래 그려	무슨 장식을 생각하고 계세요?
쳐발렸는데 밥까지 사면 좆같지 않것냐? 이기는 팀이 사는 걸루 혀	팀이 졌는데 밥까지 사는건 너무 슬프지 않을까요? 이기는 팀이 사는 것으로 합시다.

해당 방법론의 핵심적인 문제로는, 실제 댓글에서 나타나는 인물 비방, 극단적 편견 등은 해당 데이터셋을 통해 학습할 수 없었다. 일부 직접적인 욕설을 담은 문장을 제외하고, 비꼬거나 특정 사건, 단체를 비하하는 표현을 포함한 데이터는 존재하지 않았다.

3-3. Parallel : 순화 대응 데이터셋 수집

Pre-training 및 fine-tuning 과정에서 모델은 욕설 및 혐오 표현에 대한 지식을 습득하지 못했다. 우리 task 의 특수성 때문에, 위의 방법들을 사용해 좋은 순화 결과를 얻기에는 어려움이 컸다.

팀 내 데이터 수집 부담을 줄이고, 실제로 많은 사용자가 만족할 수 있는 순화 결과를 생성하도록, 사용자로부터 데이터를 수집하였다. 누구나 손쉽게 혐오 표현을 순화할 수 있도록, 데이터 수집을 위한 웹 페이지를 개발하고 배포했다.

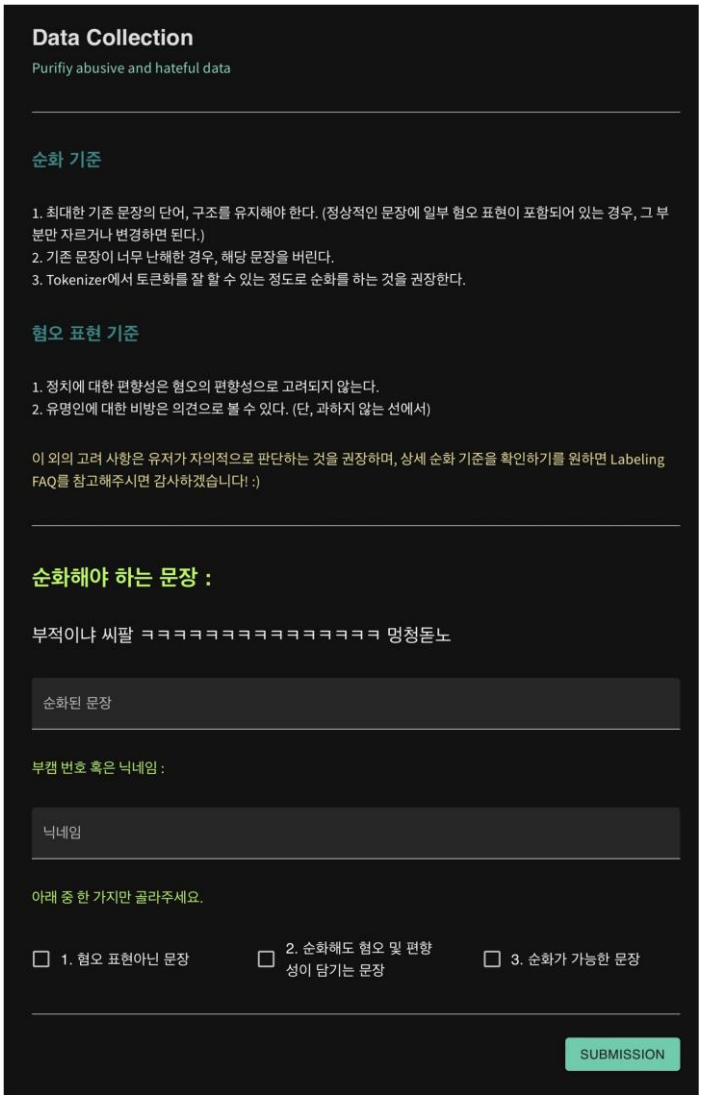


그림 6 데이터 수집 페이지. 사용자에게 간략한 순화 가이드라인을 먼저 제시하고 순화하도록 함. FAQ 페이지에 기준과 함께 예시를 제시.

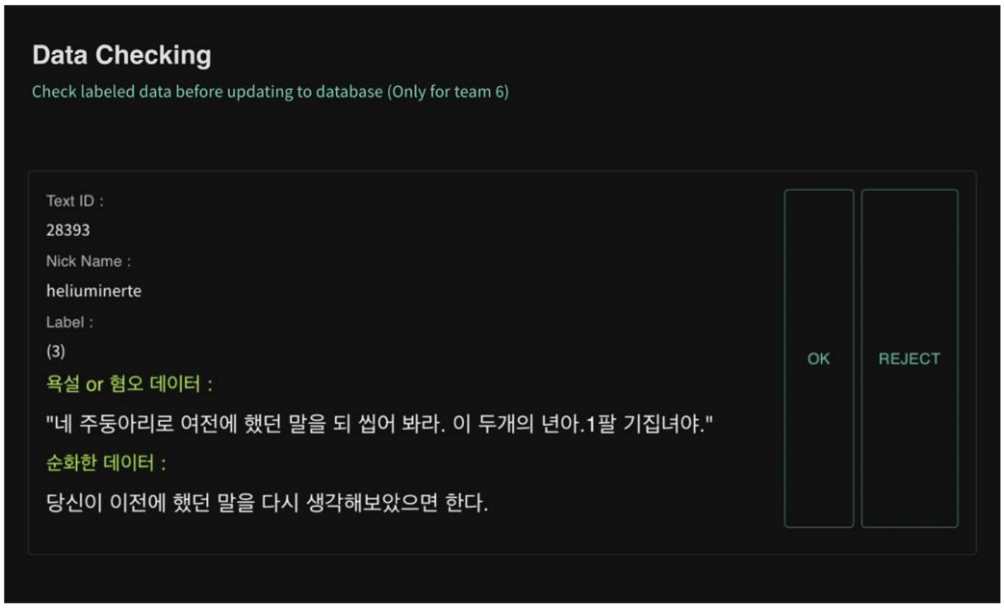


그림 7 데이터 검토 페이지. 웹페이지에서 수집된 데이터를 확인하고, 승인 여부를 DB에 반영하도록 함.

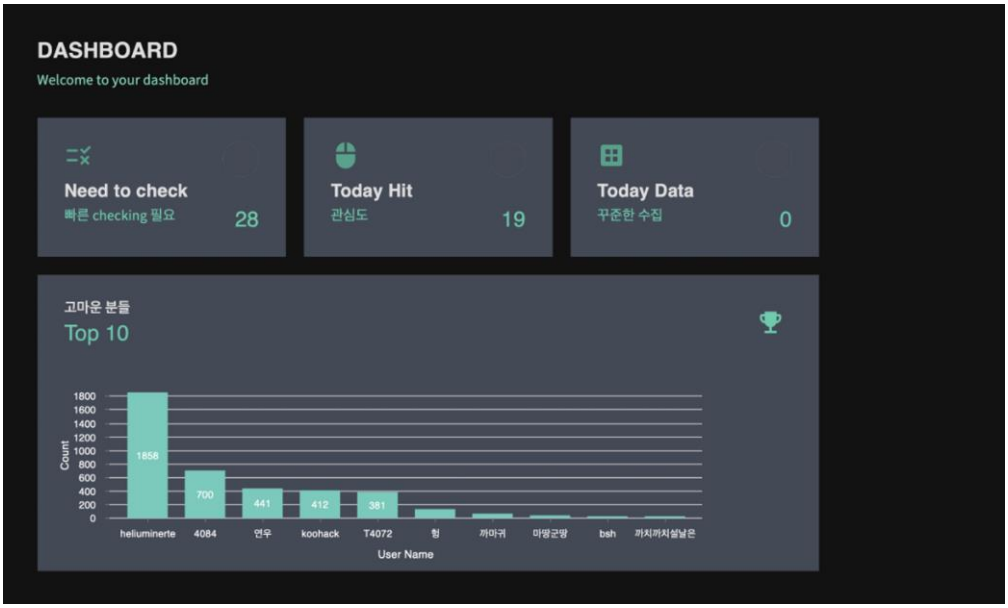


그림 8 대시보드 페이지. 대시보드를 통해 데이터 수집 및 검토와 관련한 수치 확인.

총 28명이 참여해 5,625개의 순화 문장을 생성했고, 팀 내 검토를 거쳐 최종 3,133개의 혐오 문장 - 순화 문장 쌍으로 이루어진 parallel dataset을 구축했다. 해당 데이터셋은 이어지는 generation model을 학습시키는 데에 사용되었다.

표 7 parallel dataset 예시

혐오 문장	사용자 순화 문장
자기 생각대로 하는거지 꼰대마냥 이래라저래라 존나 강요하네	자기 생각대로 하는 건데 자신의 생각을 너무 강요를 한다.
IT 는 쫏도 모르면서 틀딱 새끼가 나대네. ㅋ	IT 에 대해 잘 알고 의견을 공유했으면 좋겠다.
나라도 같은 재단 믿을만한 재단에 계속 기부할거같은데? 선행해도 ㅈㄹ이네?	나라도 같은 재단 믿을만한 재단에 계속 기부할 거 같은데? 선행에 과도하게 참견하지 말자.
할게없어서 감옥소재로 드라마를찍냐 다음엔 범죄 저지른 연예인들 나오는 감빵 예능도 만들어라 ㅅㅂ	감옥 소재의 드라마는 적절하지 않다고 생각한다.

4. Generation Model

4-1. 모델 선정

생성 모델 중 GPT, BART, T5에 대해서 어떤 pretrained 모델이 한국어를 지원하는지 확인하였고, 그 중에는 KoBART, KoGPT, MBART, mT5, KcT5 등의 모델이 존재하였다. 사전학습된 모델을 불러와 혐오 표현 데이터 셋에 대한 간단한 생성 실험을 통해 성능을 확인한 결과, MBART와 KcT5의 성능이 가장 좋아 이를 기반으로 다양한 실험을 진행하였다.

4-2. 순화 모델의 평가

순화의 경우, label 문장과 완벽하게 동일하게 생성하는 것보다는, label 문장과 동일하지 않더라도 사용자의 입장에서 납득 가능하도록 생성하는 것이 우선된다. Cross-entropy loss 는 label 과 같은 token 을 생성하는 데에는 도움을 주지만, 다른 token 으로 이루어진 비슷한 문장을 만드는 경우에 해당 문장이 얼마나 좋은 순화 결과인지 판단하기에는 한계가 명확하다.

따라서 모델의 순화 정도를 평가할 때, Human evaluation 과 혐오 표현 classifier 을 통해 성능 평가를 하고자 하였다. Human evaluation 의 경우 "처음 보는 사람에게 이런 말을 들었을 때 기분이 나쁜가?" 와 같은 평가 기준을 세워 평가자가 쉽고 명확하게 문장을 평가할 수 있도록 하였고, 혐오 표현 classifier 의 경우 생성된 문장을 classifier 에 넣었을 때 얼마나 혐오 표현이라고 판단하는지를 통해 평가하였다.

4-3. Reward

생성 모델의 성능을 향상시키기 위한 시도로 Thank you BART! 논문을 기반으로 Reward 을 적용한 학습을 진행하였다. reward 는 2 종류로

1. 비혐오 표현으로 순화되었는지를 판별하는 Style Classification Reward 와,
2. 기존 문장의 의미를 유지하고 있는지를 판별하는 BLEU Reward

를 loss 에 반영하였다.

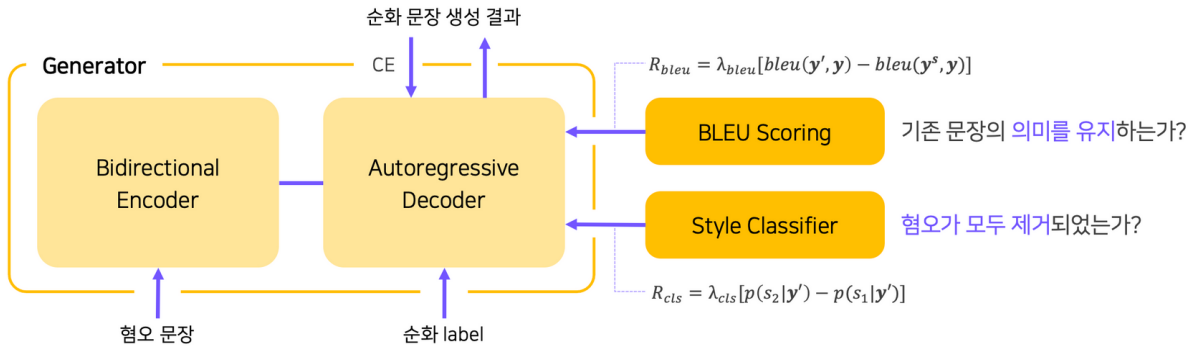


그림 9 Rewarding model architecture. Generator backbone model은 facebook/mbart-large-50, beomi/KcT5-dev.

reward를 통해 다소의 성능 향상이 있었으며, 이는 뒤의 User evaluation에 정리되어 있다.

4-4. Prompt

모델의 input 에 우리가 원하는 학습과 관련된 말을 추가하여 학습에 도움이 되도록 시도하였다.

Prompt 로 앞에 Prefix 로 "순화된 표현으로 변환: "을 붙이는 방식과 문장 앞에 "순화할 표현: ", 뒤에 "순화된 표현: " 을 붙이는 방식을 시도해 보았다. 결과는 둘 다 붙이지 않았을 때 보다 성능이 좋았고 둘 중에서는 "순화된 표현으로 변환: "을 붙이는 쪽이 좋은 결과를 내었다. Prefix 로 "순화된 표현으로 변환: "을 붙이는 결과는 뒤의 User evaluation 에 정리되어 있다.

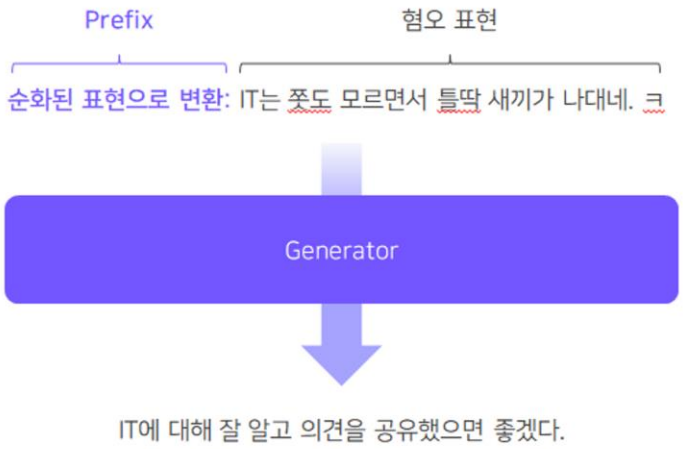


그림 10 Prompt Engineering

4-5. Instructions with Human Feedback

비꼬는 뉘앙스와 필터링이 어려운 문장들을 충분히 제거해보기 위해 유저들의 순화 기호가 들어가면 좋을 것이라 판단해 새로운 방식을 도입하고자 했다. InstructGPT는 ChatGPT에서 사용한 방식으로써 human feedback을 통해 model에 자연스러움을 추가한 방식이다. 이 방식을 통해 유저의 기호를 추가하고자 했다. 본 프로젝트에는 욕설-순화 쌍 3,133개가 사용됐으며, supervised fine-tuning에는 1,400개, reward model 구축에는 600개, 나머지는 reinforcement learning 하는데 사용됐다.

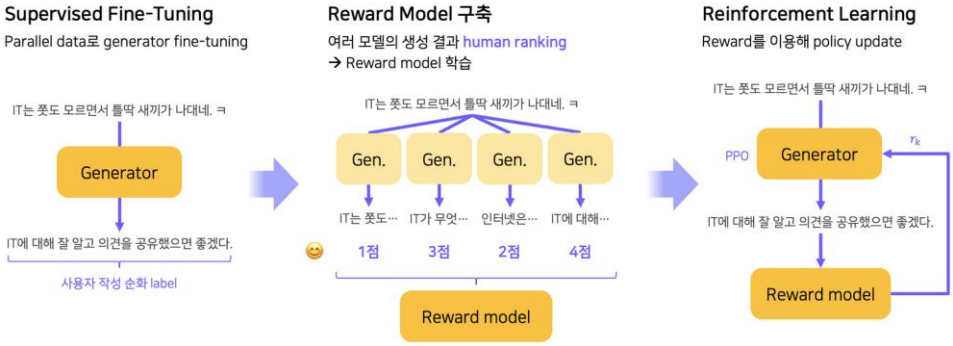


그림 11 Instructions with Human Feedback steps

1. Supervised Fine-Tuning

   human feedback 을 적용하기 앞서, user 들이 작성한 데이터들로 fine-tuning 한 모델이 필요하다. fine-tuning 하는 과정에서 엄청 많은 데이터로 학습시키지 않더라도 나쁘지 않은 성능을 보였기에 1,400 개의 데이터만을 가지고 학습을 진행했다.

2. Reward model 구축

   human feedback 을 반영하기 위한 reward model 은 필수이다. reward model 을 구축하기 위해 우선 user 의 순화 선호도를 수집하기로 했다. 600 개 혐오 데이터 셋과 다른 방식으로 fine-tuning 시킨 3 가지 모델을 사용해 600 개 문장에 대한 순화 문장 쌍을 도출했다. 최종적으로, 유저가 labeling 한 순화 문장을 포함해 한 욕설 문장 당 네 가지의 순화 문장을 매칭시킬 수 있었다.

   유저는 욕설 문장과 4 가지 순화 방식을 보고 각 순화 방식에 reward 를 채점했다. 채점된 reward 를 모두 종합하여 regression model 를 제작했다. validation dataset 으로 reward model 의 성능을 판단하기는 어렵기에 validation 은 진행하지 않았으며, reward model 이 적절한 reasonable 한 reward 를 도출할 수 있도록 최소한의 epoch 만으로 학습을 진행했다

3. PPO Reinforcement Learning

RLHF 논문에서 설명한 것과 같이 순화 문장 하나를 1 trajectory 로 설정하고 generate 된 policy 를 1 timestep 이라고 가정했다. 또한, reward 는 문장 생성이 끝났을 때에만 적용이 된다. 최종 reward 는 아래와 같은 식으로 표현할 수 있으며, objective function 또한 아래와 같은 식으로 표현 가능하다. 이런 방식을 사용해 model 이 next token prediction 하는 형식으로 학습을 진행하도록 설정했다.

$$r_{\text{total}} = r_{\text{PM}} - \lambda_{\text{KL}} D_{\text{KL}}(\text{policy} \parallel \text{policy}_0)$$

그림 12 Reward Function Equation

$$\begin{aligned}\text{objective}\,(\phi) = & E_{(x,y)\sim D_{\pi_\phi^{\text{RL}}}}\left[r_\theta(x,y) - \beta \log\left(\pi_\phi^{\text{RL}}(y \mid x)/\pi^{\text{SFT}}(y \mid x)\right)\right] + \\ & \gamma E_{x\sim D_{\text{pretrain}}}\left[\log(\pi_\phi^{\text{RL}}(x))\right]\end{aligned}$$

그림 13 Object Function Equation

실험에서는 mBART를 사용해 fine-tuning을 진행했고 다른 학습 방식을 사용해 2가지 모델을 제작했다. 한 모델을 policy를 비교하는 용도로 사용되고, 나머지는 update하기 위해 존재하는 모델이다.

해당 방법의 한계점은 아래와 같다.

- Reward model size의 한계로 인해 train loss가 수렴하지 않음

  실험 중 train loss가 수렴하지 못하고 발산하는 상황이 많이 발생했다. 다양한 이유가 있겠지만, 우선 논문에서 언급한 바에 따르면 reward 모델의 크기가 충분히 크지 않으면 reward model의 역할을 제대로 할 수 없다고 한다. 그러나, 리소스의 한계로 인해 base 모델로 실험을 진행했으며, 데이터 셋도 충분하지 않아 reward model이 제대로 된 값을 도출하지 못했을 것으로 생각된다.

- Multilingual model의 단점

  위에서 언급한 바와 같이 reward 모델의 성능에 따라 결과가 불안정해지는 모습을 볼 수 있다. train이 불안정하면 당연히 도출해내는 값들도 불안정해질 수밖에 없다. 불안정한 모델에서 순화를 진행할 때 다른 언어가 출현하는 경우가 종종 존재했다. 따라서, 다른 T5 혹은 BART 모델을 사용했으면 좋았을 것 같다.

- Dataset 부족

  단순하게 fine-tuning하는 방식은 매우 큰 데이터 셋을 요구하지 않는다. 그러나, 강화학습을 적용하는 경우 다양한 경우의 수를 고려하면서 학습이 진행되야 강화학습의 강점을 가져갈 수 있는데, 이번 프로젝트에서는 dataset을 자체적으로 제작해야 하는 부분이 매우 컸기에 데이터셋 부족을 겪을 수밖에 없었다.

4-6. User Evaluation

실제 서비스를 이용하는 건 사람이기에, human evaluation을 통해 각 모델의 순화 결과를 수치화 했다. 사용자에게 모델에 대한 정보를 알려주지 않고, 아래 4개 모델이 각각 생성한 결과에 대해 아래와 같은 기준으로 평가하도록 했다.

[기준]

1) 원본 문장에 없던 욕설이 생성되거나 문장이 그대로인가?
2) 의미가 변질되었는가?
3) 의미가 사라졌는가?
4) 직접적인 욕설이 남아있는가?
5) 초면인 사람에게 들었을 때 기분이 나쁜가?

원본 문장	순화문장1	기준	점수(0 or 1)	순화문장 2	점수(0 or 1)	순화문장3	점수(0 o
뭔가 천박 한 이미지 왜 천 박하게 보일까?못 배운 부류도 아닌데 희한해 묘해	외모가 부족한 면이 있어보인다.	없던 욕설이 생성되거나, 문장이 그대로인가?	0	뭔가 부족한 느낌		뭔가 고상한 이미지는 아닌 것 같다. 잘 알지 못하는 분야도 아닌데 이상하게 보인다	0
		의미가 변질되었는가?	0				1
		의미가 사라졌는가?	1		1		0
		직접적인 욕설이 남아 있는가?	0		0		0
		초면인 사람에게 들었을 때, 기분이 나쁜가?	0		0		0
			3		3		3

그림 14 User Evaluation 예시 사진

User evaluation 결과 Rewarding model에 prompt-based learning을 적용한 모델이 가장 높은 점수를 받았고 이를 최종 모델로 채택하였다.

	Raw 학습	Reward	Reward + Prompt	Instruct + Prompt
평가 평균 점수	59	64	**66**	63
가장 높은 평가 비율	12%	26%	**38%**	24%
가장 낮은 평가 비율	48%	20%	**10%**	22%
평가 점수 표준편차	0.87	0.9	**0.84**	0.86

그림 15 모델별 user evaluation 결과

5. Architecture

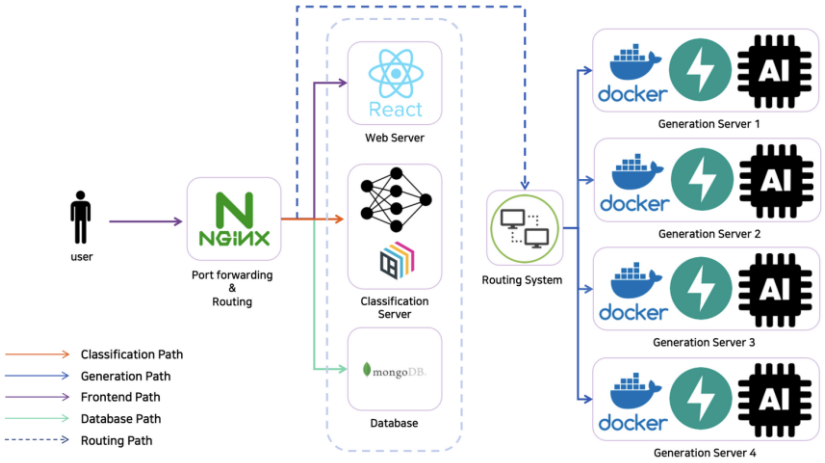
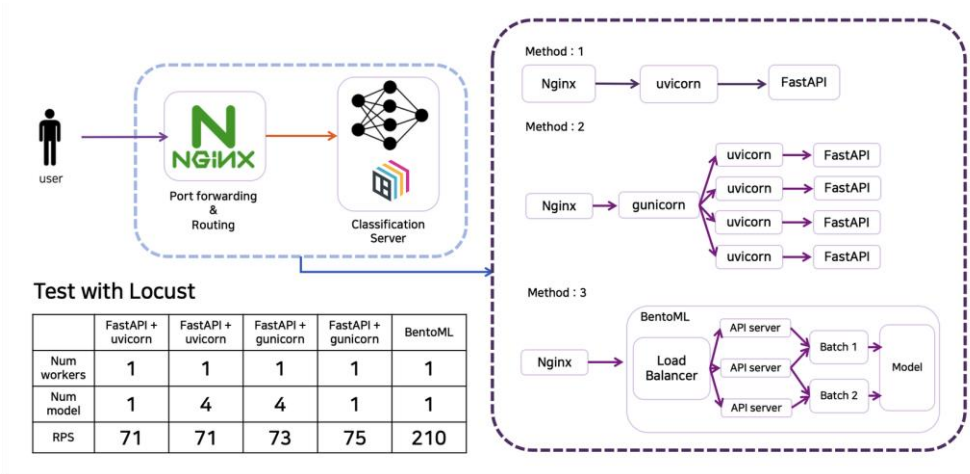


그림 16 전체 서버 구조

보다 효율적인 서비스 환경을 구성하기 위해 간단한 서버 구조를 사용하지 않고 다양한 실험을 통해 서버 구조를 최적화하고자 했다. 대략적으로 classification server와 generation server로 나눠서 볼 수 있다. classification과 generation은 다른 것으로 생각할 수 있기에 이 API의 성능을 비교 분석하기 위해 따로 구분하여 locust로 stress test를 진행했다.

5-1. Classification 서버 구조



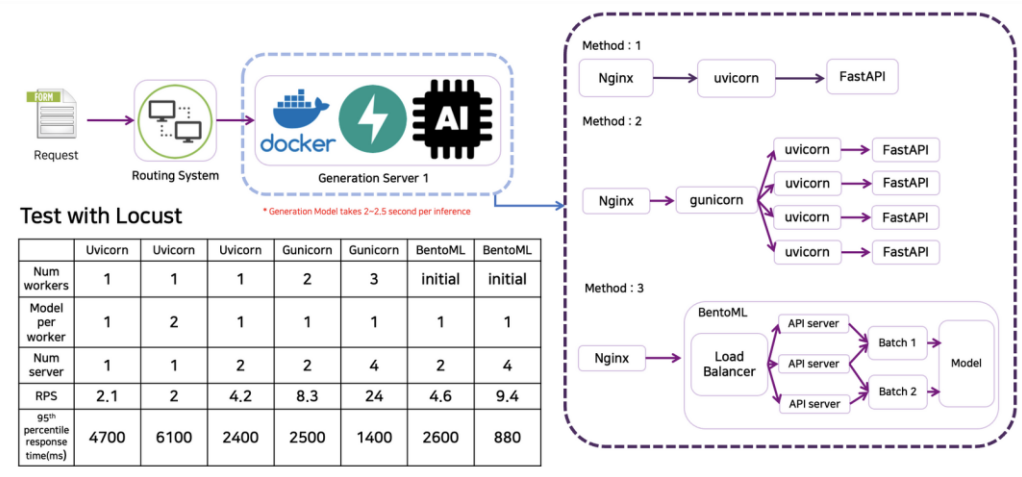
	FastAPI + uvicorn	FastAPI + uvicorn	FastAPI + gunicorn	FastAPI + gunicorn	BentoML
Num workers	1	1	1	1	1
Num model	1	4	4	1	1
RPS	71	71	73	75	210

그림 17 Classification 서버 구조

classifcation은 inference 속도가 매우 빠르므로 worker의 갯수를 늘려준다면 더 많은 request를 처리할 수 있고, batch servicing할 경우 이 성능은 배로 향상되는 것을 알 수 있었다. 특히 bentoML을 사용해 batch serving을 진행하는 경우에는 비약적인 성능 향상

이 있었다. classification은 가장 많은 request를 처리해야 하기에 좋은 성능의 API 서버가 필요하다. 따라서, bentoML을 classification에 적용하기로 했다.

5-2. Generation 서버 구조



Test with Locust

	Uvicorn	Uvicorn	Uvicorn	Gunicorn	Gunicorn	BentoML	BentoML
Num workers	1	1	1	2	3	initial	initial
Model per worker	1	2	1	1	1	1	1
Num server	1	1	2	2	4	2	4
RPS	2.1	2	4.2	8.3	24	4.6	9.4
95th percentile response time(ms)	4700	6100	2400	2500	1400	2600	880

그림 18 Generation 서버 구조

이에 반해, generation model은 추론하는데 많은 시간이 소요되기에 batch 형식이 아닌 낱개로 Inference할 경우가 성능 조금 더 좋다. 또한, 비혐오 표현인 경우 generation을 수행하지 않기 때문에 batch serving이 조금 더 좋지 못하다고 판단했다. 그 이유는 대부분의 사람들이 욕설 및 혐오 표현을 작성하는 것이 아니기 때문에 batch가 되는 것을 기다리는 것보다 먼저 generation을 시작하는 것이 더 좋다고 판단되기 때문이다. 따라서 최종적으로 gunicorn을 연동하는 방식으로만 generation server를 구성했다.

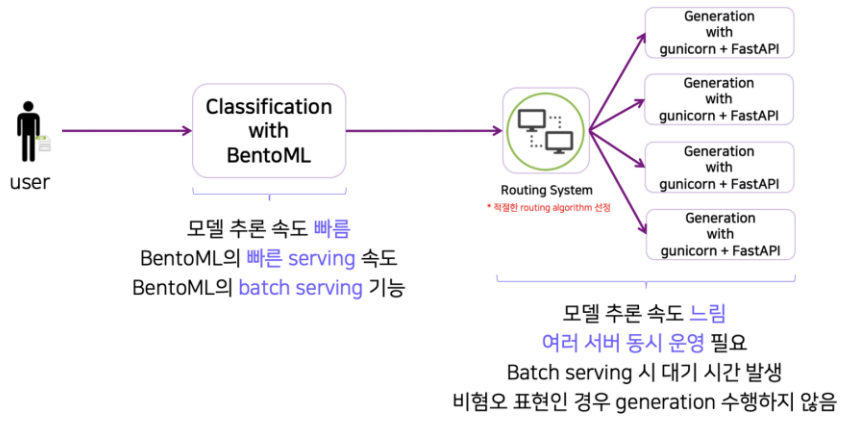


그림 19 서비스 도식화

6. 회고

6-1. 팀 회고

프로젝트 장점

- 건전한 온라인 소통을 위해 새로운 방식을 제안했다
- 혐오표현-순화표현 문장 쌍 데이터셋을 구축 및 공개하였다.

프로젝트 한계 및 개선방향

- 대량의 트래픽 처리하기

우리 서비스는 1초당 최대 500개의 트래픽을 수용할 수 있었다. 트래픽이 많아지는 경우, 사용자는 댓글 작성 결과를 1초 이상 기다려야 받을 수 있다. 네이버 클린 봇 같은 경우 댓글을 저장한 이후 classification을 진행한다. 따라서, 우리 서비스도 모든 댓글을 수용하되, classification만 먼저 진행하여 욕설을 분류하고 반 실시간으로 classified된 욕설 및 혐오 표현을 순화해주면 더 좋은 서비스가 될 것으로 기대한다.

- 데이터 수집

한정된 프로젝트 기간으로 공개되어 있는 혐오 표현 데이터셋을 사용하여 우리만의 혐오 기준을 적용하기 어려웠다. 이로 인해 모델 간의 데이터 통일성이 떨어질 수 있다. 앞으로 개선해 나간다면, 혐오 표현 기준을 다시 정립하고 데이터셋을 새로 라벨링하여 일관된 기준으로 분류 모델과 생성 모델을 학습시킬 것이다.

순화 데이터 쌍이 부족하여 생성 모델의 퀄리티를 높이는데 한계가 있었다. 보다 세부적인 가이드라인을 작성하고, 라벨러들을 고용하여 더 양질의 데이터를 더 많이 수집하는 방향으로 개선할 수 있을 것이다.

- 새롭게 생겨나는 혐오 반영하기

혐오 표현은 날마다 다른 주제와 다른 표현으로 생성되고 있다. 하지만 과거 시점의 데이터셋으로 학습한 우리 모델은, 새롭게 생겨나는 혐오 표현을 알지 못한다. Continual learning을 통해, 새롭게 작성되는 댓글들을 데이터로 모델을 지속적으로 학습시킬 수 있다.

- Pretrained model의 한계

사전 학습된 모델에서 학습된 Tokenizer가 우리 task에 적합하지 않았다. 혐오 댓

글의 경우 유명인의 이름이 자주 등장하는데, 모델이 이를 몰라서 다른 것으로 바꾸는 경우가 있었다. 예를 들어, '문가영'이라는 인명에 대한 정보를 모델이 가지고 있지 않아, 가지고 있는 지식 중 가장 유사하다고 판단한 '문재인영'을 생성하는 경우가 존재했다. 우리 task에 최적화된 Tokenizer를 사용하여 새로 학습시킬 수 있다. 또한, rule-based로 모델에 넣기 전에 헷갈릴 수 있는 일부 표현을 처리할 수 있다.