혐오발언 검출 (Hatespeech Detection)

코드스테이츠 AI부트캠프 1기 이창우

1. 들어가며

HateSpeech란 특정한 인종이나 국적 · 종교 · 성별 등을 기준으로 다른 사람들에 대한 증오를 선동하는 발언이나 약자에게로 향하는 '혐오(증오) 발언'을 의미하며 이는 증오범죄(hate crime)로까지 이어질 수 있다.[1] 특히 SNS상에서의 혐오표현은 지속성, 확산성, 접근성, 익명성, 규제 부재로 혐오표현을 빠르고 광범위하게 확산시킬 수 있다.[2]

본 프로젝트에서는 채팅상황에서 적용할 수 있는 혐오표현 검출 모형을 제작하는데 목적이 있으며 과제에는 제약사항이 따른다. 실시간으로 반응할 수 있어야하기 때문에 (1)추론속도가 빨라야하고 (2)그럼에도 잘 분류할 수 있어야하며 (3)특정상황이 아닌 일반적으로 좋은 성능을 낼 수 있는 것에도 초점을 두었다. 또한 (4)문맥을 포함하여 검출하고자 시도하였다.

목적을 달성하고자 가볍고 빠른 고전적인 머신러닝기법부터 BERT기반의 경량화된 모델등을 사용하였고 모형의 검출속도가 문장 당 50ms 이내로 기준을 삼았다.

Dataset	abbr.
A Benchmark Dataset for Learning to Intervene in Online Hate Speech	Benchmark
COunter NArratives through Nichesourcing: a Multilingual Dataset of Responses to Fight Online Hate Speech	Conan
Automated Hate Speech Detection and the Problem of Offensive Language	DT
Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior	FOUNTA
Toxic Comment Classification Challenge	TCCC
Multi-hate-target knowledge-grounded hate countering dataset	Multi
Are You a Racist or Am I Seeing Things? Annotator Influence on Hate Speech Detection on Twitter	Wz_ls
tweets_hate_speech_detection (Huggingface Datasets)	THSD
SemEval-2019 Task 5: Multilingual Detection of Hate Speech Against Immigrants and Women in Twitter	HateEval
Offensive Language Identification dataset	OLID
Hate Speech Dataset from a White Supremacy Forum	White

〈표 1〉 수집한 데이터셋 목록

2. 관련연구

2.1 문맥의 영향[3]

행동실험결과를 통해 문맥이 인간의 판단에 미치는 영향을 분석하고, 문맥을 조건에 추가하였을 때 성능을 향상시킬 수 있는지에대한 연구가 진행되었다. 이 연구의 주된 두 가지 질문은 다음과 같다.

(가) 문맥이 사람의 결정에 영향을 미치는가?

(나) 상황에 따른 조건이 성능을 향상시키는가?

게시판에서 바로 이전 게시물까지만으로 제한시키는 방식을 통해서 문맥을 포함시켰다. 당연하게도 사람은 어떤 발화가 혐오발언인지 아닌지에 대해 판단을 내릴 때 문맥이 영향을 미쳤다. 하지만 혐오발언을 검출하는 데에서는 유의미한 성능향상을 이끌어내지 못했다.

인간의 판단에는 문맥이 중요한 역할을 했던 것으로보아 혐오발언 검출에 분명 필요한 조건이며 앞으로 더 연구가 필요한 주제임은 분명하지만 본 프로젝트에서는 시간을 이유로 모델 훈련에서 문맥을 포함시키지 않고 예측과정에서 문맥을 포함하였다.

---

1) Article19. 2015 'Hate Speech' Explained A Toolkit

2) 국가인권위원회. 2016. 혐오표현 실태조사 및 규제방안 연구

3) John Pavlopoulosy, Jeffrey Sorensenz, Lucas Dixonz, Nithum Thainz, Ion Androutsopoulosy. 2020.. Toxicity Detection: Does Context Really Matter?

2.2 데이터셋의 호환[4)]

온라인에서 어떤 데이터셋으로 훈련을 시켰느냐에 따라서 결과가 상이했다. 예를 들어, 트위터 데이터셋으로 훈련시킨 경우 레딧 데이터셋에서는 혐오 발언 검출을 잘 하지 못하는 경향이 있었다. 본 프로젝트는 이 문제를 해결하고 일반적인 성능을 나타내기 위해 다양한 데이터셋을 수집하고 검증하여 최선의 조합을 만들어내고자 하였다.

2.3 예측 소요시간[5)]

BERT모델을 효과적 경량화시키는 것으로 알려진 둔 SqueezeBERT를 제안하는 논문에서 기존 BERT가 각 단계별로 소요된 시간을 표로 정리해 놓았다. 이 표에서는 임베딩이 차지하는 시간은 0.26%, Self Attention module에서의 FC(Fully Connected)단계와 Softmax단계에서 차지한 시간은 총 30.2%였으며 Feed-forward Network Layers에서의 FC가 69.4%, Classifier에서 0.02%로 Feed-forward에서 매우 많은 시간을 소비한다는 실험결과를 확인하였다. 본 프로젝트에서는 Feed-forward에서 소요되는 시간을 단축하고자 Feed-forward부분을 CNN으로 바꾸어보려고 시도했으나 시간문제로 중단하였고 classifier를 Logistic으로 대신 사용하여 시간 단축을 시도해보았다.

2.4 BiLSTM, GloVe, BERT

GloVe[6)](Global Vectors for Word Representation)는 구글에서 개발한 Word2Vec의 단점을 보완한 임베딩방식이다. 기존의 Word2Vec은 중심단어로 주변단어를, 주변단어로 중심단어를 예측하는 과정에서 주어진 단어를 임베딩하여 벡터로 변환하는 방식이며, GloVe는 단어 상호 간 비율 정보를 말뭉치 전체를 놓고 한 번에 반영하는 방식이다. 단어벡터 간 유사도를 좀 더 쉽게 측정하면서 단어 전체에 대한 정보를 좀 더 잘 반영할 수 있다. 학습데이터를 대상으로 co-occurrence 행렬을 만들어 연산을 하는데 계산복잡성이 크다는 단점이 있으나 공식 홈페이지[7)]를 통해서 이미 훈련된 단어벡터를 받아 사용할 수 있다.

BiLSTM(Bidirectional Long short-term memory)은 게이트 기법을 통해 순환 신경 회로망의 한계를 극복한 모델인 LSTM을 이용하였다. 순방향 뿐만 아니라 역방향의 결과를 함께 반영하는 모델이다. 문맥)을 기반으로 하는 연관성 분석에 유리하며 속도를 고려한 다양한 자연어처리 문제에 널리 활용되고 있다.

BERT(Bidirectional Encoder Representations form Transformer)는 Transformer의 Encoder부분을 적극 활용한 모델이다. BERT는 Wiki와 Book data와 같은 대용량 데이터를 Masked Language Model(MLM)과 Next Sentence Prediction 기법을 통해 모델을 pre-training한다. BERT의큰 장점은 새로운 신경망 등을 붙일 필요 없이 BERT 모델 자체의 fine tuning을 통해서도 좋은 성능을 기대할 수 있다는 점이며 이미 여러 문제에서 fine tuning만으로도 state of the art를 달성했다.

---

4) Kunze Wang, Dong Lu1, Soyeon Caren Han, Siqu Long, Josiah Poon. 2020. Detect All Abuse! Toward Universal Abusive Language Detection Models, Kunze Wang1

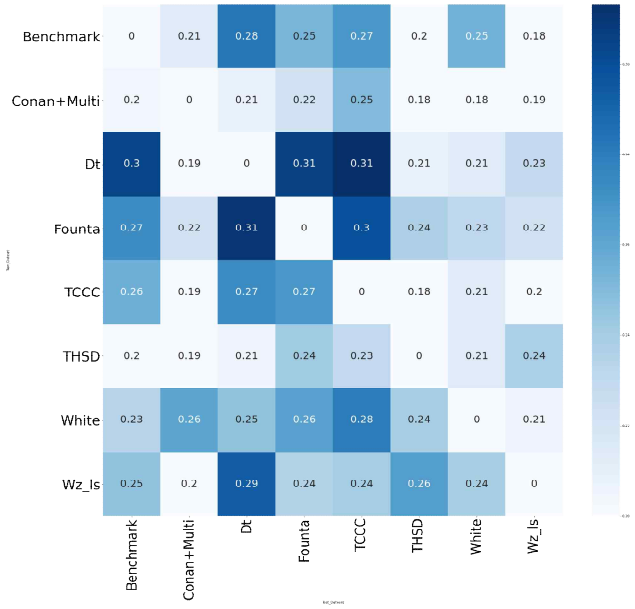
5) Forrest N. Iandola, Albert E. Shaw, Ravi Krishna, Kurt W. Keutzer. 2020. SqueezeBERT: What can computer vision teach NLP, about efficient neural networks? Table 1

6) Jeffrey Pennington, Richard Socher, Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation

7) https://nlp.stanford.edu/projects/glove/

3. 데이터셋

사용된 데이터셋은 1~11과 같으며 본 보고서에는 우측의 약어로 기재한다. 영어로 된 데이터 셋은 많이 구할 수 있었다. 트위터아이디만 포함되어있는 FOUNTA와 Wz-Ls 데이터 셋에서도 기존 트위터가 삭제되는 등 데이터가 소실되었음에도 불구하고 전처리 후 중복을 제거했을 때에 총 수집한 데이터는 약 40만개에 달한다.



〈그림 1〉 수집한 데이터셋 간의 검증결과

전처리과정에서는 먼저 트위터 특성상 RT(ReTweet)을 남기며 답변을 남기는데, RT뒤에는 아이디가 오고 이 정보는 의미가 크지 않을 거라고 판단하여 제거하였다. URL 주소 또한 의미가없다고 판단하여 제거하였다. 이모티콘은 감정을 드러내는데 유의하지만, 혐오발언 검출에 대해서는 영향이 적다고 판단하였다. 특수문자는 '!'과 '?'만 남기고 다 제거하였다. 해쉬태그는 '#'뒤에 정보를 담고있기 때문에 해쉬(#)만 제거하였다. 표제어추출(Lemmartization)과 어간추출(Stemming), 불용어(Stopwords)제거를 시도해보았고 평가결과도 좋았으나 시간에 지체가 생겨서 최종 기능에 포함하지 않기로 결정하였다.

오타를 고쳐주는 방법을 시도해 보았다. Jamspell[8]을 사용하여 'I love youuuuuuu'와 같은 발화를 'I love you'로 수정해주어 좀 더 발화에 의미를 남길 것이라 예상했다. cpu에서의 속도가 2.6ms[9]이므로 성능향상이 있다면 충분히 감내할 만한 소요시간이었다. 그러나 TF-IDF와 GloVe로 평가해본 결과 모든 경우에서 하락하였다. 이유는 'cunt'와 같이 혐오의 주요한 키워드가 되는 단어를 cant로 수정하는 경우가 있었기 때문이다. 이 때문에 성능이 오히려 하락한 것으로 추측된다.

데이터 검증을 수행하였다. Glove BiLSTM으로 한 데이터셋으로 훈련시키고 검증하였다. 적은 데이터셋으로 일반적인 성능이 뛰어나다면 매우 좋은 데이터셋일 것이라고 생각하여 Computer Vision 분야에서 파라미터의 효율성을 따질 때 사용하는 지표 중 하나인 Information Density이 컨셉을 차용하여 데이터셋의 양에따라서 페널티를 주었다. 방법은 ROC AUC 점수에서 데이터셋에 포함된 데이터의 갯수의 1/10승으로 나누었다. 결과는 〈그림 1〉으로 시각화하였다.

$$Factor_{Validation} = \frac{ROCAUC_{Test}}{\sqrt[10]{\mathrm{Len}_{\mathrm{Dataset}}}}$$

대부분의 데이터셋에서는 일반적인 성능을 보였으나 THSD와 conan+multi데이터 셋으로 훈련한 모형은 모든 검증 데이터 셋에서 0.25 이하로 나타났으므로 일반적인 성능이 없다고 판단하였고 최종 데이터셋에서 제외시키기로하였다.

8) https://github.com/bakwc/JamSpell
9) https://github.com/neuspell/neuspell, Performance

Train Dataset [count]	Validation Dataset									
	Bench -mark	Dt	Founta	Hate -Eval	OLID	TCCC	THSD	White	Wz_ls	mean
Benchmark+Dt+Founta+Wz_ls+TCCC+ White+HateEval+OLID [8]	0.967	0.983	0.957	0.868	0.886	0.988	0.741	0.921	0.893	0.912
Benchmark+Dt+Founta+Wz_ls+TCCC+ THSD+White+Conan+Multi+HateEval [10]	0.947	0.966	0.906	0.810	0.755	0.959	0.863	0.825	0.828	0.873
Benchmark+Founta+Wz_ls+TCCC+THSD+White +Conan+Multi+HateEval+OLID [10]	0.948	0.918	0.912	0.806	0.769	0.963	0.875	0.825	0.819	0.871
Dt+Founta+Wz_ls+TCCC+THSD+White+ Conan+Multi+HateEval+OLID [10]	0.883	0.967	0.908	0.811	0.778	0.964	0.867	0.814	0.818	0.868
Benchmark+Dt+Founta+Wz_ls+TCCC+ THSD+HateEval+OLID [8]	0.944	0.965	0.907	0.807	0.760	0.958	0.872	0.769	0.826	0.868
HateEval+OLID [2]	0.765	0.889	0.876	0.824	0.782	0.893	0.716	0.800	0.685	0.803
Founta+Wz_ls [2]	0.819	0.902	0.880	0.707	0.744	0.829	0.726	0.732	0.843	0.798
Dt+Wz_ls+Conan+Multi+HateEval+OLID [6]	0.853	0.956	0.777	0.799	0.720	0.776	0.549	0.711	0.816	0.773
Benchmark+Wz_ls [2]	0.950	0.881	0.775	0.629	0.638	0.799	0.653	0.770	0.829	0.769
White+Conan+Multi [3]	0.663	0.662	0.720	0.679	0.625	0.774	0.607	0.856	0.618	0.689

〈표 2〉 데이터 조합 검증 결과

데이터셋 조합을 44번 시행하여 평가하였다. THSD와 Conan, Multi 데이터셋을 제외한 모든 데이터 셋을 합쳐 1번 진행하였고, 데이터셋의 개수를 2개부터 10개까지 무작위로 선택하여 43번진행하였다. 각 데이터셋은 훈련데이터셋과 검증데이터셋으로 나눈 뒤에 진행하였으며 훈련시에는 훈련데이터셋만 사용하고 검증시에는 검증데이터셋을 사용하였다. 결론적으로 데이터와 일반적인 성능은 데이터의 양이 많아짐에 따라 높아졌고 THSD와 Conan+Multi를 포함하지 않은 데이터 셋이 성능이 가장 좋았으며 최종 데이터셋으로 선정하였다. 〈표 2〉에서는 상위 5개와 하위 5개에대한 평가결과를 첨부하였다.

언더샘플링을 통해 최종데이터 셋을 선정하였다. 위에서 선정한 데이터 셋은 총 352,541개이고 1의 클래스(혐오발언)를 가진 데이터는 77,140으로 21.88%의 비율이었으나 클래스의 불균형한 문제를 해결하고 프로젝트의 시간제한상 더 빠르게 훈련을 시키기 위해서 1의 개수만큼 0을 언더샘플링하였다. 추가적으로 1000글자가 넘는 데이터는 삭제하였다. 최종적으로 선정된 데이터 셋은 총 149,799개이며 모델의 검증을 위해 훈련데이터셋은 119,868개, 검증데이터셋 29,931개이다.

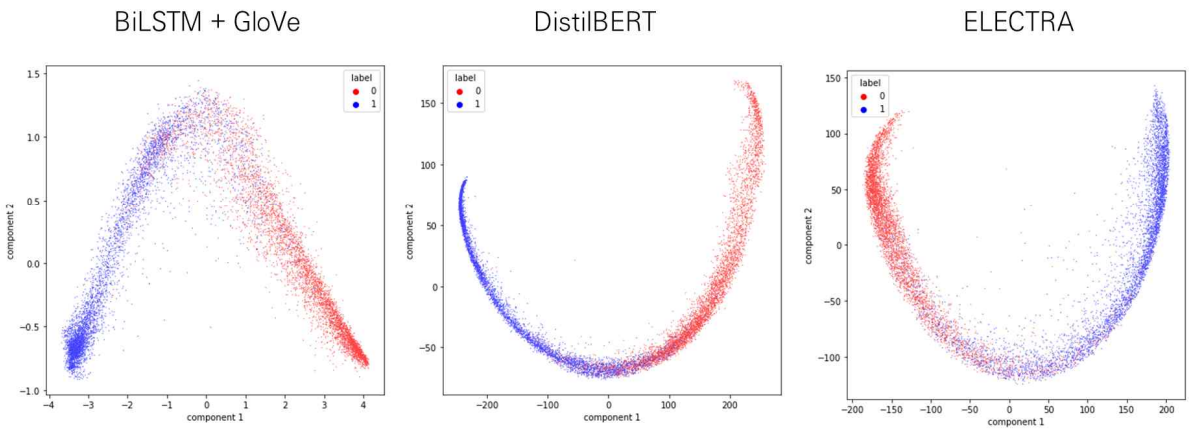
4. 모델링

3.1 모형의 훈련

결정하기까지 가장 오랜 시간이 걸린 문제이다. 이 결정이 데이터셋을 어떻게 구축할지에대한 문제이기 때문이다. BERT의 스페셜 토큰인 '[SEP]'토큰을 사용하여 부모문장과 자식문장을 넣어 훈련하는 방법을 고려하였다. 그러나 채팅이라는 상황을 고려할 때, 게시판의 글은 성격이 매우 다르다고 판단하였다. 채팅에 대한 데이터셋은 찾을 수가 없어 차선책을 선택하였다.

많은 고민 끝에 결정한 사항은 일반적인 방법으로 훈련하는 것이었다. 위에서 선정한 데이터셋을 사용하여 가장 흔한 방법으로 각 발화가 독립변수 X가되고 혐오발언 여부를 종속변수 y로 모델을 훈련시켰다. 혐오발화가 부분적으로 있더라도 발화 전체가 혐오표현이 된다는 것을 근거로 삼았다.

실제 서비스를 할 때에는 발화를 2개 내지 5개로 합친 뒤에 결과를 출력한다. 예를들어, 직전발화가 "Hello, My name is Changwoo"이고 현재 작성된 발화가 "Hey Monkey"라고 했을 때 모델에 입력되는 문장은 "Hello, My name is Changwoo Hey Monkey"가 된다. 이처럼 문장을 이어서 입력한다면 문맥을 포함하여 예측할 수 있다고 기대한다.



〈그림 2〉 각 모형의 마지막 레이어의 결과를 PCA를 통해 시각화한 것

4.2 사용한 모형

데이터 셋 조합 이전에 TF-IDF를 시도해보았고, 불용어를 제거하고 어간추출을 사용하여 Randomforest를 통해 분류하였을 때 F1 Score가 0.81정도로 준수한 성능을 보였고 개선여지가 있었다. 하지만 용어가 자주 바뀌는 혐오발언 특성 상 TF-IDF는 확장가능성이 낮다고 생각되어 추가적인 시도는 하지 않았다.

최종데이터 셋으로 시도한 모델은 크게 (1)GloVe + [Logistic , Randomforest, LGBM]과 (2)BERT기반 모델들, (3)BERT기반 모델을 통한 임베딩 + [Logistic, Randomforest, LGBM]로 나눌 수 있다.

모델에 따라서 내부적으로 데이터를 어떻게 분류하였는가를 살펴보기위해 훈련된 BiLSTM+LSTM, DistilBERT, ELECTRA모형의 마지막 Hidden state를 추출하였고, PCA를 통해 2차원으로 축소한 뒤 〈그림 2〉로 시각화를 진행하였다. 마지막 레이어에서 2차원으로 줄인 상태인데도 육안으로도 잘 분류가 된 것처럼 보이며, 시간이 많이 소요되는 FC layer를 붙이지 않고 Logistic과 같이 단순한 모형으로도 충분히 좋은 결과를 낼 것이라 예상된다.

4.3 성능평가

훈련된 ELECTRA 모형의 성능이 가장 좋았다. 의외인 것은 FC 레이어를 통해 결과를 낸 것이 오히려 성능이 낮아졌다는 점인데, 훈련데이터셋에 과적합된 것으로 추측된다. ELECTRA모형에 가장 단순한 모형인 Logistic도 두 번째로 우수한 성능을 나타내었다.

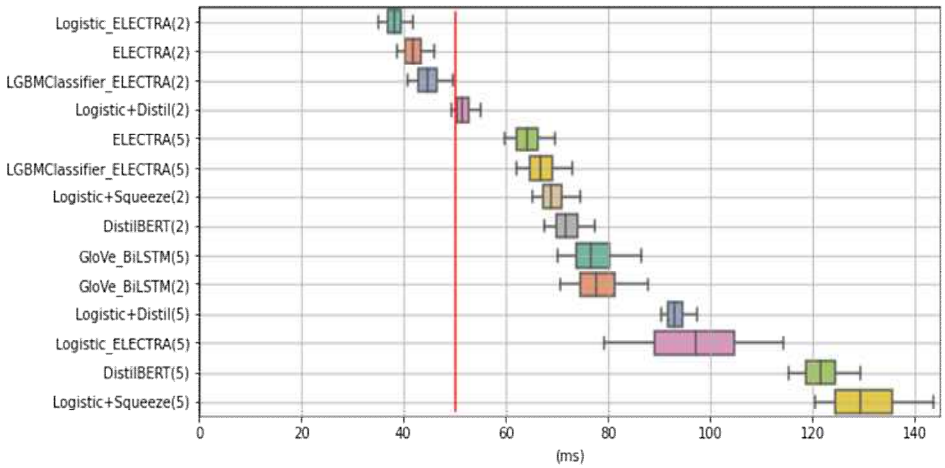
DistilBERT도 그 뒤를 이어서 좋은 성능을 나타내었고, 이 모형을 사용하여 임베딩을 한 결과는 상대적으로 좋지 않은데 이유는 Finetuning이 되지 않은 상태에서 임베딩을 진행하였기 때문이다. 그럼에도 GloVe를 통해 임베딩한 것과 비교적 좋은 성능을 보여주고 있어 개선가능성이 크게 보인다. DistilBERT의 Finetuing은 끝났으나 임베딩하여 다시 진행하기에는 프로젝트의 마감시간이 촉박하므로 넘어가기로한다. GloVe으로 임베딩 후 BiLSTM 모형으로 분류하였을 때도 ROC AUC Score이 0.0937로 예상보다 매우 좋은 결과를 나타냈다.

Model		Val_Score	
임베딩	Classifier	ROC_AUC	F1 Score
ELECTRA	LGBM	0.963	0.898
ELECTRA	Logistic	0.962	0.896
ELECTRA		0.961	0.896
ELECTRA	RandomForest	0.960	0.895
DistilBERT		0.953	0.879
GloVe	BiLSTM	0.937	0.865
DistilBERT*	Logistic	0.920	0.844
SqueezeBERT*	Logistic	0.920	0.844
DistilBERT*	LGBM	0.915	0.837
SqueezeBERT*	LGBM	0.915	0.837
SqueezeBERT*	RandomForest	0.903	0.824
DistilBERT*	RandomForest	0.903	0.824
GloVe	LGBM	0.879	0.782
GloVe	RandomForest	0.816	0.725

〈표 3〉 모델의 성능평가

* finetuning이 되지 않은 상태에서 임베딩을 하였음.

4.4 Benchmark



〈그림 3〉 각 모형 별 Benchmarks

벤치마크는 Colab에서 하드웨어가속기가 없는 상태에서 진행했으며 Intel(R) Xeon(R) CPU @ 2.00GHz이 배정되었다. 시간을 측정하는데 사용한 문장은 "*Hello, My Name is Changwoo. Hello, Mind Logic. Hello, CodeStates*"이다. 문맥을 포함하기 위하여 문장을 2개 혹은 5개를 이어서 총 5000번을 시행하였고, 이상측정을 지우기위해 양쪽으로 10%씩 제거하였다. 최종적으로 측정데이터는 4000개로 평가하였으며. 평균 및 오차는 〈테이블〉과 같다. 95% 신뢰구간(평균 ± 2표준편차)을 기준으로하여 〈표 4〉에 정리하였다.

〈표 4〉의 Model열에 괄호는 문맥을 포함하기 위하여 잇는 문장의 개수이다. '(2)'인 경우에는 2문장을 이어 예측하고, '(5)'인 경우에는 5문장까지 이어서 예측한다. 기존 BERT기반 모델에서 FC layer로 분류한 것과 비교하였을 때, 단순한 모형인 Logistic Regression이 가장 빠르게 검출하였다. 기준인 50ms의 기준을 통과한 것은 Logistic+ELECTRA, ELECTRA, LGBM+ELECTRA로 총 3가지이며 5문장을 이어서 예측한 모형은 모두 시간내에 들지 못했다. 따라서 최종적으로 선정한 모델은 성능평가에서도 2위를 차지하고 속도측면에서도 가장 우수했던 Logistic+ELECTRA (2) 모형으로 결정하였다.

ELECTRA 모형은 Transformers API를, Logistic Regression은 Scikit-learn API를 이용하였고 훈련된 모델은 BentoML을 통해 배포하였다.

Model (Number of Sentence)	Elapsed time (95%)
ELECTRA+Logistic (2)	38.16 ± 3.4 ms
ELECTRA (2)	41.83 ± 3.7 ms
ELECTRA+LGBM (2)	44.74 ± 4.5 ms
DistilBERT+Logistic (2)	51.53 ± 2.9 ms
ELECTRA (5)	64.17 ± 5.2 ms
ELECTRA+LGBM (5)	66.93 ± 5.8 ms
Squeeze+Logistic (2)	69.05 ± 4.7 ms
DistilBERT (2)	71.92 ± 5.3 ms
GloVe+BiLSTM (5)	77.13 ± 8.3 ms
GloVe+BiLSTM (2)	78.02 ± 8.8 ms
DistilBERT+Logistic (5)	93.16 ± 3.7 ms
ELECTRA+Logistic (5)	96.85 ±19.1 ms
DistilBERT(5)	121.68 ± 7.3 ms
Squeeze+Logistic (5)	130.16 ±12.9 ms

〈표 4〉 각 모형별 Benchmarks

〈그림 4〉 BentoML 작동 화면 :
문맥을 통해서 클래스 예측이 달라지는 경우

5. 마치며

5.1 DistilBERT와 SqueezeBERT에대한 추가시도

컴퓨터 비젼의 컨셉을 차용하여 가장 흥미롭게 생각하던 Squeeze 모델을 프로젝트의 제한된 시간내에 훈련하지 못하였다. 또한 훈련된 DistilBERT 모형을 통해 임베딩을 하지 못했다는 것이 아쉬운 점으로 남는다. 결과를 보면 Distil BERT와 SqueezeBERT 파라미터 수가 다르지만 pre-trained된 모형으로 마지막 hidden state를 사용하여 임베딩한 결과가 소수점 3자리까지 같았다. Fine-tuning된 DistilBERT모형은 Electra보다는 성능결과가 좋지 않았지만 매우 준수한 결과를 나타냈는데, SqueezeBERT에서도 좋은 성능이 기대되며 개선할 여지가 많다고 생각된다.

5.2 한국어 혐오표현 검출

온라인에서의 혐오발언은 피해자에게 정신적으로 큰 피해를 입히고 이는 사회적 충격으로 다가오기도한다. 연예인 등 유명인이 악플로 인하여 극단적인 선택을 할 때마다 공동체 전반에 우울감을 느끼는 현상이 확산되고 모방 시도가 발생할 가능성이있다. 최근 쿠팡이츠를 이용하는 식당의 점주가 새우튀김으로 비롯된 악성 댓글 및 갑질로 인하여 극단적인 선택을 하는 사건도 있었다. 이러한 부정적인 영향으로인하여 혐오발언 및 악성댓글 등을 막고자 인터넷 실명제 등 많은 방법이 제안되었지만 이를 멈추기에는 역부족이었다.
심지어 어린아이가 찍어올린 유튜브영상에도 상대적 열등감으로 가득찬 댓글들을 어렵지않게 확인할 수 있었다. 또한 인터넷 정치, 연예, 스포츠 기사에 달린 댓글들은 보기 불쾌할 정도이다. 이렇게 쉽게 혐오발언을 접할 수 있는 것 자체가 검열이 제대로 이루어지지 않고있다는 것을 반증한다.
이러한 검열서비스를 제공하기위한 시도도 이루어지고 있다. NAVER에서는 음절단위로 토큰화하고, Persona 임베딩, ELMO 전이학습을 통해 클린봇 2.0이라는 서비스를 제공하고 있다. 본 프로젝트에서 얻은 통찰을 기반으로 한국어 혐오표현 검출 프로젝트를 이어서 진행하고자한다. 데이터를 직접 수집하여 데이터 셋을 구축할 예정이며, 게시글의 제목과 내용, 게시글의 댓글, 댓글의 댓글로 데이터를 구성하고 모형은 BERT에서 FC layer대신 CNN을 이용하여 특징을 추출하는 방법을 구상중이다.

5.3 프로젝트를 끝내며

글이나 음성보다 이미지가 시각적인 요소를 포함하기 때문에 더 자극적이다. 그래서 컴퓨터 비전쪽에 관심이 더 갔었다. 예상하던 결과더라도 직접보면 너무나도 신기하기 때문이다. 또한 사람의 눈으로는 pixel 값이 조금 다르다고 하더라도 잘 인지하지 못한다. 그렇기에 방향만 맞다면, 비슷하게 결과를 낼 수 있다면 나름 만족스러운 결과를 얻을 수 있었다. 하지만 자연어처리는 정반대였다. '아'다르고 '어'다르다고, 벡터로는 매우 작은 수치의 차이더라도 사람이 인식하기에는 전혀 다른 의미를 지니게 된다. 이번 프로젝트에서 난항을 겪었던 가장 큰 이유다.
그렇기에 자연어처리는 더 매력적인 분야인 것 같다. 딥러닝의 모든 과제가 마찬가지이겠지만 매우 다양하고 많은 시도가 이루어지고 있는데, 자연어처리또한 놀랍도록 신비한 경험을 제공한다. 사람이 쓰는 글을 이해하고 사람처럼 글을 쓰며 심지어 소설까지 쓰며 철학적인 이야기도 논한다. 항상 느끼는 것이지만 0과 1이라는 두가지 숫자로 논리를 만들고 사람의 마음을 울리는 것은 매우 경이로운 일이다.
앞으로 인공지능은 산업 전반에 흩어져 적용될 것이고 사회 깊숙이 자리잡게 될 것이라 생각한다. 이러한 거대한 역사의 흐름에 몸을 담게되었다는 사실만으로도 흥분되는 일이며, 인공지능 기술을 통해 사람의 생활에 직간접적으로 도움을 주거나 사회적가치를 실현시키는 일을 하고자한다.