HABERTOR: An Efficient and Effective Deep Hate speech Detector

2022. 01. 05 이세영

Contents

HABERTOR 요약

HABERTOR: HAtespeech detectOR using BERT

1) 자체 어휘사전 생성, 가장 큰 규모의 hatesppech dataset을 사용하여 사전 훈련 됨.

2) quaternion 기반으로 factorized 된 컴포넌트로 구성되어 훨씬 적은 수의 파라미터, 더 빠른 training 및 inference, 더 적은 메모리 사용량

3) 분리된 input 소스들을 위한 pooling layer가 있는 multi-source 앙상블 헤드를 사용하여 그 효과를 강화 시킴

4) fine-grained and adaptive noise magnitude로 robust 하게 한다.

=> HABERTOR 가 BERT 보다 효율적이고 효과적이다.

Introduction

- Hatespeech 는 SNS의 발달로 더 빠르게, 더 많은 사람들이 접할 수 있게 됨.
- 온라인에서의 혐오감 표현이 현실세계에서의 물리적 폭력으로 이어질 수 있음으로 묵인하고 있어서는 안됨.

이전의 연구들 (ML, DL기반 방법)

- 단방향으로 읽음으로써 모든 important features 를 탐색하지 않거나 사전 훈련된 언어모델을 잘 활용하지 못함.
- BERT(transformer 기반 양방향 인코더)는 supervised task 에 대한 fine-tuning 을 위한 좋은 기초를 형성함.

Hatespeech에 대해 BERT 적용

- 이전 분류기보다 뛰어난 결과
- 한계1: hatespeech 가 일반적인 문장과 다른 언어/의미적 특성을 소유함
  ex) 은어, 오타, 줄임말 등 dick 은 일반적인 것이지만 d1ck 은 욕설
- 한계2: 매우 큰 파라미터 수 (최소 110M)
  매우 큰 계산 자원 필요

Introduction

본 연구의 목표

- BERT 기반의 모델로
- 매개 변수를 크게 줄여
- BERT보다 더 나은 hatespeech 예측 성능 달성

Main Contribution

1. Yahoo 에서 추출한 대규모 hatespeech dataset
2. 4분위 기반 인수분해 메커니즘 (quaternion-based factorization mechanism)
3. multi-source ensemble head fine-tuning architecture
4. target-based adversarial training

Problem Definition

w_n 은 words, n 은 시퀀스 s의 최대 길이

시퀀스 $s = [w_1, w_2, \ldots, w_n]$ 가 주어지면, 매핑함수 f를 구축하는 것을 목표로 함.

$$f : s = [w_1, w_2, \ldots, w_n] \longrightarrow \mathcal{R} \in [0, 1]$$

여기서 f는 s를 입력으로 하여 probability score $P(y = 1|s) \in [0, 1]$을 반환한다.
P는 input s가 hatespeech 로 분류될 가능성

- 언어 이해를 향상시키기 위해 unsupervised language modeling task 로 f를 사전 훈련시킨다.
- 그 다음, f를 classification task 로 훈련시킨다.

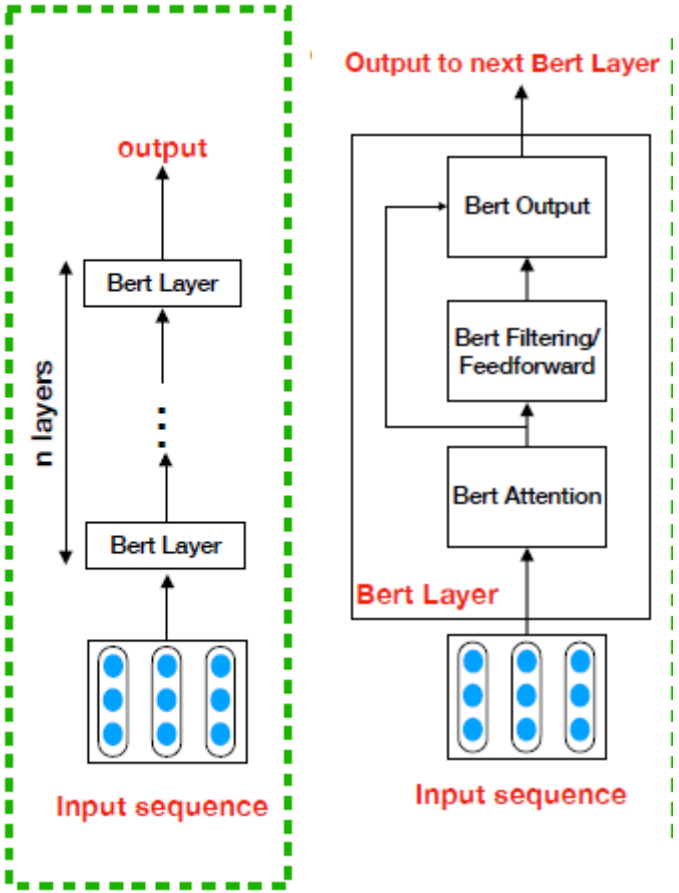
HABERT0R

1. Tokenization

- BERT 모델이 사용한 SentencePiece 사용하여 input texts 를 tokenize

2. Parameter Reduction with Quternion Factorization

  - V: vocab size, E: Embedding size, H: hidden size, L: number of Layers,
  - F: feed-forward/filter size
  - BERT에서 F=4H, E=H, 어텐션 heads 수=H/64,
  - Encoding the vocabs takes VH parameters
  - attention, filtering/feedforward and output layer 포함(각 $4H^2$ 파라미터 포함)
  - 즉, BERT는 $12H^2$ 파라미터가 있으며, BERT-base는 12개의 레이어로, VH+$144H^2$ 파라미터가 있음.



HABERTOR

- V: vocab size, E: Embedding size,
- H: hidden size, L: number of Layers,
- F: feed-forward/filter size

2. Parameter Reduction with Quternion Factorization
   - 파라미터를 크게 줄이기 위해 Quternion factorization(paper) 전략 제시
   - **Vocab Factorization(VF):**
     - V vocabs 를 Quternion representations 으로 인코딩 (임베딩 크기 E ⟪ H)
     - E를 다시 H로 변환하는데 Quternion transformation 을 적용
     - 실수차원의 임베딩 형식에 맞게 4개의 Quternion 파트 통합

     $O(V \times H)$에서 $O(V \times E + E \times H)$
     BERT ALBERT

     => VE+EH/4 parameters 가 됨. (ALBERT VE+EH 보다 적음)
   - **Attention Factorization(AF):**
     - input 시퀀스의 길이가 N이면 multi-head attention 의 output은 NxN이다. *H에 의존 X
     - 따라서 Query, Key, and Value 를 같은 H로 $3H^2$ 파라미터로 생성할 필요 X
     - 대신 Query, Key, Value 를 linear Quaternion transformation 을 사용하여 C 크기로 생성 C⟪H
     - 파라미터는 3CH/4 가 됨. 3H*H -> 3C*H

HABERTOR

- V: vocab size, E: Embedding size,
- H: hidden size, L: number of Layers,
- F: feed-forward/filter size
- C≪H

2. Parameter Reduction with Quternion Factorization
   - 파라미터를 크게 줄이기 위해 Quternion factorization 전략 제시
   - **Feedforward Factorization(FF):**
     - H -〉 4H로(즉, $4H^2$) 선형 변환하는 대신 H -〉 I, I-〉 4H 로 Quternion 변환을 한다. (I ≪ H)
     - 따라서 총 HI/4 + IH 파라미터가 생성된다.
   - **Output Factorization(OF):**
     - 4H -〉 I 에 Quternion transformation 적용, I-〉 H 가 된다.
     - HI + IH/4 가 된다.

위의 압축기법을 적용하여 전체 파라미터는 VE + EH/4 L(3CH/4 + H^2 + 5HI/2) 로 감소.

BERT-base 설정이 V=32k, H=768, L=12인 경우 E=128, C=192, I=128로 설정하면 파라미터의 총합이 110M에서 8.4M로 감소

HABERT0R

50-50 rule
M: input text sequences 의 수
N: 시퀀스의 words 수
p1: next pair 일 확률

$$M \times p_1 = (M+N) \times p_2 \text{ or } \frac{p_1}{p_2} = \frac{(M+N)}{M}. \text{ Since } p_1 + p_2 = 1, \text{ replacing } p_2 = 1 - p_1, \text{ we have: } M \times p_1 = (M+N)(1-p_1) \longrightarrow p_1 = \frac{(M+N)}{(2M+N)}.$$

3. Pretraining tasks

◆ **Masked token prediction task**

- BERT 는 매번 입력마다 시퀀스 중 하나만 masking 하여 학습한다.
- 본 연구에서는 masked position을 τ개를 랜덤하게 바꿔 τ 라는 training instances 를 생성한다.

masking factor

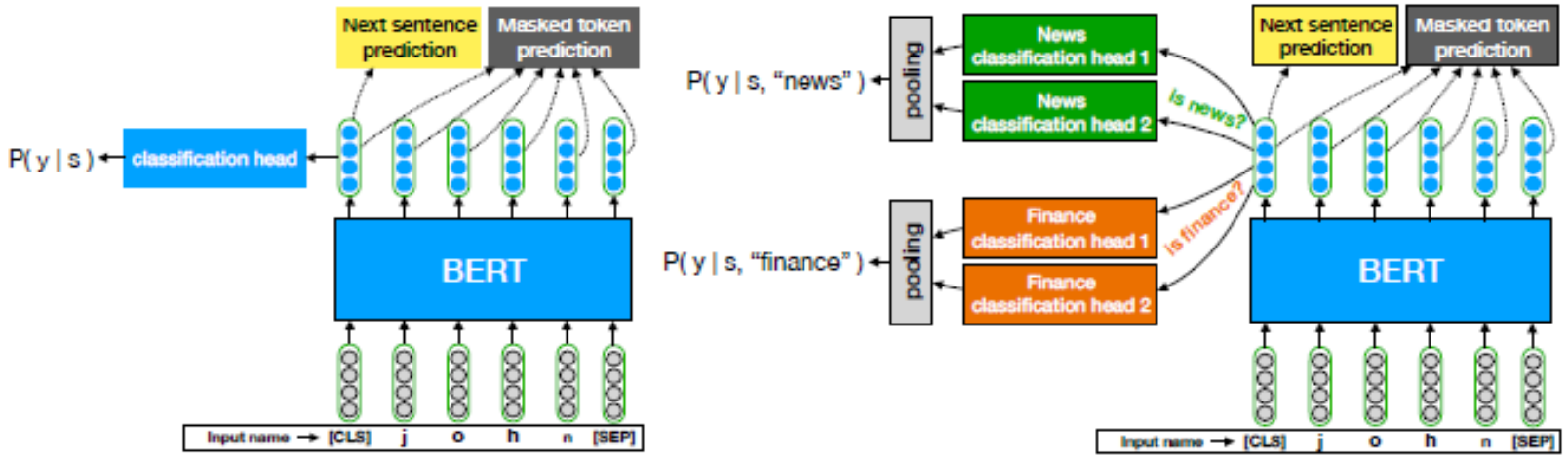
◆ **Next sentence prediction task**

- BERT 에서는 두 개의 입력 문장이 쌍을 이루고 있지만, 본 연구에서는 짝을 이룬 문장을 준비하기 위해 전처리과정을 거쳐야 한다.

1) NLTK 라이브러리에서 unsupervised 문장 tokenizer 를 학습시킴. 학습된 tokenizer 를 사용하여 각 입력 시퀀스를 토큰화, 문장으로 분할
2) BERT는 두 문장이 pair 일 확률이 50% 아닐 확률이 50%, 본 연구에서는 인풋이 1,2,3개 이상으로 분리될 수 있음. 따라서 not next 예제를 생성하기 위해 다른 무작위 문장과 짝을 맺어야 한다.

HABERTOR

4. Training the hatespeech prediction task
   - Multi-source multi-head HABERTOR classifier 제안



(a) Traditional Fine-tuning BERT. (b) HABERTOR with two sources and ensemble of 2 heads.

Figure 1: Architecture comparison of traditional fine-tuned BERT and HABERTOR multi-source ensemble heads.

   - 서로 다른 소스의 다른 입력 시퀀스에 대해 공유 언어 이해 지식을 가진 classification heads/nets 분리
   - h 개의 앙상블 헤드를 가져 데이터 분산을 더 잘 할 수 있다.

HABERTOR

5. Parameter Estimation

- hateful 예측 아래 cross-entropy loss 를 최소화하는 것을 목표로 함.

$$\mathcal{L}_{hs} = \underset{\theta}{\operatorname{argmin}} \; -\sum_{i=1}^{|S|} \boldsymbol{y}_i \log\left(\hat{\boldsymbol{y}}_i\right) + (1-\boldsymbol{y}_i)\log\left(1-\hat{\boldsymbol{y}}_i\right)$$

S: Collection of sequences
s_i: sequence
s_i 가 뉴스관련 문장인 경우 $\hat{y}_i = P(y_i|s_i, \text{"news"})$

- FGM(adversarial training) 을 사용하여 모델을 robust 하게 함.

$$\mathcal{L}_{adv} = \underset{\delta,\delta\in[a,b]}{\operatorname{argmin}} \; -\sum_{i=1}^{|S|} logP\left(\tilde{\boldsymbol{y}}_i \,|s_i+\delta_i;\hat{\theta}\right) \qquad (1)$$

$$\delta_i = -\epsilon \times \frac{\nabla_{s_i}\left(-logP\left(\tilde{\boldsymbol{y}}_i \,|s_i;\hat{\theta}\right)\right)}{\|\nabla_{s_i}\left(-logP\left(\tilde{\boldsymbol{y}}_i \,|s_i;\hat{\theta}\right)\right)\|_2} \qquad (2)$$

$$\mathcal{L} = \mathcal{L}_{hs} + \lambda_{adv}\mathcal{L}_{adv} - \lambda_\epsilon\|\epsilon\|_2, \qquad (3)$$

Empirical Study

Table 1: Statistics of the three datasets.

Statistics/Datasets	Yahoo	Twitter	Wiki
Total	1.4M	16K	115K
# Hateful	100K	5K	13K
% of hatespeech	7%	31%	12%

- **Dataset**
  - Yahoo News and Finance의 5년 동안의 댓글에서 추출, 약 140만개의 레이블링 된 댓글로 구성됨.
  - 야후 뉴스 – 약 94만개, 야후 파이낸스 – 약 48만개
  - 트위터: 약 5천개의 hatespeech, 총 16000개의 레이블링된 데이터로 구성
  - Wiki: 영어 위키백과 talk's 페이지의 115,000 개 레이블링된 토론 댓글, 약 13천개의 hate 포함
  - hatespeech 약 10만개
  - train/dev/test : 70%/10%/20%
- **Models**
  - 15 개의 state-of-the-art 모델 baseline
  - BoW, KimCNN, VDCNN, RoBERTa 등
- **Measurement**
  - AUC, AP(average precision), FPR(False Positive Rate), FNR(False Negative Rate), F1-score

Empirical Study

Table 3: Performance of all models that we train on Yahoo train data, test on Yahoo test data and report results on Yahoo News and Yahoo Finance separately. Best baseline is underlined, **better** results than best baseline are **bold**.

Model	Yahoo		Yahoo News					Yahoo Finance				
	AUC	AP	AUC	AP	FPR@ 5%FNR	FNR@ 5%FPR	F1	AUC	AP	FPR@ 5%FNR	FNR@ 5%FPR	F1
BOW	85.91	48.35	85.07	51.37	61.13	50.53	49.01	85.83	36.80	60.97	49.43	40.15
NGRAM	84.19	42.15	83.17	45.00	63.45	57.45	43.59	84.29	31.63	63.42	53.94	35.95
CNN	91.21	63.03	90.64	65.64	47.50	36.23	60.61	91.20	52.30	45.59	33.96	51.93
VDCNN	88.10	58.08	87.65	60.75	60.39	41.56	56.12	88.17	48.72	62.43	38.78	50.38
FastText	91.64	60.15	90.97	63.16	41.80	38.09	58.35	92.13	47.97	37.75	34.30	49.36
LSTM	91.83	64.17	91.14	66.59	43.81	35.09	60.96	92.38	54.44	38.26	31.45	53.36
att-LSTM	91.83	64.39	91.10	66.77	44.24	34.86	61.37	92.43	54.79	38.32	30.75	53.79
RCNN	91.17	63.34	90.52	65.72	48.49	36.37	60.29	91.32	53.77	49.40	32.17	52.73
att-BiLSTM	92.52	64.17	91.93	66.82	38.07	34.68	61.54	92.93	53.97	36.05	31.14	52.58
Fermi	86.53	41.52	86.10	45.16	53.33	55.60	45.65	85.45	27.53	56.60	56.48	33.27
Q-Transformer	92.34	64.43	91.81	67.06	39.12	34.17	61.82	92.64	54.41	37.71	29.74	53.51
Tiny-BERT	93.60	68.70	93.03	70.80	34.50	30.37	64.42	94.09	60.25	31.16	25.09	57.58
DistilBERT	93.68	69.15	93.13	71.25	34.33	30.05	64.69	94.12	60.56	29.23	24.94	58.01
ALBERT	93.50	67.99	92.93	70.28	34.56	31.15	63.82	93.94	58.73	30.12	25.87	56.37
BERT-base	94.14	70.05	93.56	71.65	32.15	28.91	65.30	94.60	62.34	29.14	22.81	59.72
HABERTOR	**94.77**	**72.35**	**94.12**	**73.79**	**29.26**	**27.12**	**67.09**	**95.72**	**65.93**	**22.03**	**18.99**	**62.38**
HABERTOR-VQF	**94.70**	**71.82**	**94.00**	**73.25**	**29.50**	**27.79**	**66.57**	**95.81**	**65.20**	**20.78**	**20.08**	**61.60**
HABERTOR-VAQF	**94.59**	**71.53**	**93.90**	**73.02**	**29.94**	**27.92**	**66.51**	**95.63**	**64.64**	**23.08**	**20.39**	**60.84**
HABERTOR-VAFQF	**94.43**	**70.75**	**93.72**	**72.37**	**31.86**	**28.58**	**65.81**	**95.42**	**63.07**	**22.87**	**21.43**	**60.11**
HABERTOR-VAFOQF	**94.18**	69.92	93.51	71.63	32.47	29.26	**65.35**	**95.00**	61.99	**24.95**	22.81	59.50

Best Baseline

Best

Contribution

Main Contributions

1. Yahoo, twitter, wiki talk's page 에서 추출한 대규모 hatespeech dataset
2. 4분위 기반 인수분해 메커니즘 (quaternion-based factorization mechanism)
   ① parameters 수를 크게 줄임
   ② 더 빠른 training 및 inference
   ③ 더 적은 메모리 사용량
3. multi-source ensemble head fine-tuning architecture
4. target-based adversarial training -> robust model

Limitation

- Hatespeech 가 일반적인 문장과 다른 언어/의미적 특성을 소유했다는 점 지적 -> 해결하지 못 함.
- Dataset 의 hate / not hate 기준을 명시해주지 않음.

Thank You