CRUSH: Contextually Regularized and User anchored Self-supervised Hate speech Detection

Souvic Chakraborty[*†], **Parag Dutta**[1*#], **Sumegh Roychowdhury**[*†], **Animesh Mukherjee**[†]
[#]Indian Institute of Science (IISc), Bangalore, KA, IN - 560012
paragdutta@iisc.ac.in
[†]Indian Institute of Technology (IIT), Kharagpur, WB, IN - 721302
{chakra.souvic,sumeghtech,animeshm}@gmail.com

Abstract

The last decade has witnessed a surge in the interaction of people through social networking platforms. While there are several positive aspects of these social platforms, their proliferation has led them to become the breeding ground for cyber-bullying and hate speech. Recent advances in NLP have often been used to mitigate the spread of such hateful content. Since the task of hate speech detection is usually applicable in the context of social networks, we introduce *CRUSH*, a framework for hate speech detection using User Anchored self-supervision and contextual regularization. Our proposed approach secures $\approx$ 1-12% improvement in test set metrics over best performing previous approaches on two types of tasks and multiple popular English language social networking datasets.

***Note**: This paper contains materials that may be offensive or upsetting to some people.*

1 Introduction

Today, the world is more connected than ever in the history of mankind. This can primarily be attributed to: (i) the technological advancements that have made affordable internet connections and devices available to people, and (ii) the social networking platforms that have hosted and connected these people. As a result, even people divided by geography can seamlessly interact in real-time without stepping outside their homes. In fact, social networks are an integral part of today's society.

We, however, are more concerned about the pitfalls of this global widespread use of social networks. The unprecedented and rapid explosion in social networking and social media use has left many people — particularly the youth, women, and those from ethnic and religious minority groups — vulnerable to the negative aspects of the online world like sexual harassment, fake news and hate speech (Jakubowicz, 2017). The number of toxic users dumping their radically biased views and polarising content onto these networks have burgeoned to such a level that they are causing political discord and communal disharmony. Therefore such posts must be intervened upon and filtered before they have the intended effect on the mass. With a huge number of posts on many popular social networking websites every second, manual filtering for hate speech does not scale. Hence, automating hate speech detection has been the primary focus of many researchers in recent years, both in academia and industry alike.

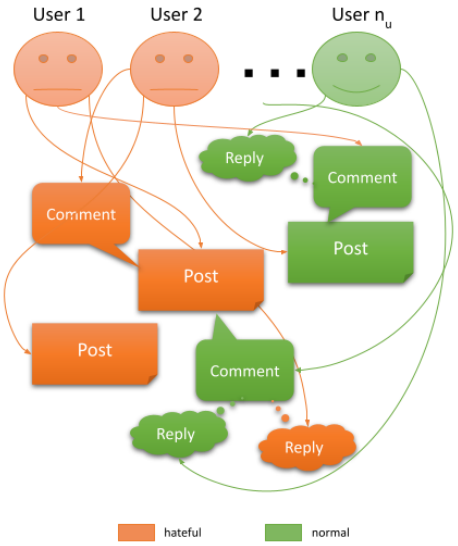

Figure 1: [Best viewed in color] Hateful content tend to cluster together in common threads and usually come from few hateful users in social media. We stress on this informed assumption to learn better representations using self supervision and contextual regularization. In the sub-graph shown in the pic, the textual content is in the form of posts, comments on the posts (optional), and replies to the comments (optional). n_u is the total number of users in the network (dynamic), and each of the text sequences can be attributed to one of the users.

Figure 1 depicts and describes a sub-graph of a typical social network. It is essential to leverage this structure within social networks for infusing

[*]Equal Contribution.
[1]Work done while at IIT Kharagpur.

Findings of the Association for Computational Linguistics: NAACL 2022, pages 1874 - 1886
July 10-15, 2022 ©2022 Association for Computational Linguistics

network context while identifying hate speech. We investigate and notice that the social graph context can be disentangled into two components: (i) *Post context*: the context in the neighborhood around the text sequence, i.e., the sub-graph consisting of posts, comments, and replies (see Figure 1) and (ii) *User context*: the context from all the existing text sequences in the network that originated from the same user (see, for instance, the connections emanating from users 1, 2, n_u etc. in the Figure 1). Relying on the echo-chamber effect, we accordingly propose a framework that uses self supervision from unlabelled data harnessed from the social networks (Gab & Reddit in our case), so that we can use contextual information (user & post context) to generate better representations from input sentences for hate speech related tasks. The main contributions of this paper are:

(i) First, we propose **UA (User Anchored self-supervision)**, a self-supervised contrastive pre-training objective. Essentially we try to incorporate the mapping from text sequences to users into the language model. In addition, we provide a Robust UA strategy that incorporates the hate speech downstream task information into our proposed UA pre-training approach.

(ii) Next, we propose **CR (Contextual Regularization)**, a regularization strategy based on the findings of Mathew et al. (2020). Here we introduce a loss based on the informed assumption that the neighboring comments and replies (in the social graph) of a hateful comment is more likely to be hateful than the comments/replies in the vicinity of a non-hateful comment. It helps us to regularize the supervised learning paradigm by learning better representations of text sequences in social network context.

(iii) We experiment with two types of hate speech tasks – classification and scoring – across three datasets. We show that our approach secures $\approx$ 1-4% improvement in F1-scores (for classification tasks) and $\approx 12\%$ improvement in mean absolute error (for scoring task) when compared to best competing baselines.

To the best of our knowledge, we are the first to use text sequence based self-supervision in hate speech using user characteristics. One of the key technical contribution that this paper makes is to show that it is more advantageous to use context to regularize a classification model than directly infusing the context into the model. Also, none of our proposed approaches require any additional annotation effort or introduce any extra parameter into the training pipeline, and are therefore scalable.

2 Related work

Hate speech is heavily reliant on linguistic complexity. Waseem (2016) showed that classification consensus is rare for certain text sequences even among human annotators. Automatic detection of hate speech is further strongly tied to the developments in machine learning based methods.

Until recently, feature engineering was one of the popularly used techniques. Gitari et al. (2015) designed several sentiment features and Del Vigna et al. (2017) used the sentimental value of words as the main feature to measure the hate constituted within a text sequence. Empirical evidence was provided by Malmasi and Zampieri (2017) indicating that n-gram features and sentiment features can be successfully applied to detect hate speech. Rodriguez et al. (2019) constructed a dataset of hate speech from Facebook, and consequently proposed a rich set of sentiment features, including negative sentiment words and negative sentiment symbols, for detecting hateful text sequences. As witnessed by the above works, it was widely believed that sentiment played an important role in detecting underlying hate.

More recently, deep learning based methods (Badjatiya et al., 2017) have garnered considerable success in detecting hate speech since they can extract the latent semantic features of text and provide the most direct cues for detecting hate speech. Zhang et al. (2018) developed a CNN+GRU based model to learn higher-level features. (Kshirsagar et al., 2018) passed pre-trained word embeddings through a fully connected layer, which achieved better performance than contemporary approaches despite its simplicity. Tekiroğlu et al. (2020) constructed a large-scale dataset for hate speech and its responses and used the pre-trained language models (LM) for detection. These methods demonstrated the considerable advantages of interpreting words in the context of given sentences over static sentiment based methods.

Self-supervision and auxiliary supervision have also been explored for hate speech detection. HateBERT (Caselli et al., 2021) used the Masked Language Modelling (MLM) objective to learn con-

textual hate semantics within text sequences from individual Reddit posts. HateXplain (Mathew et al., 2021) used human annotators to obtain rationale about the text containing hate speech and then applied the same for improving their language model. Researchers have previously tried context infusion in the inputs (Menini et al., 2021; Vidgen et al., 2021; Pavlopoulos et al., 2020). Del Tredici et al. (2019) showed that user context helps in downstream tasks. On the other hand, Menini et al. (2021) and Vidgen et al. (2021) show that contextual classification is harder given contextual annotations. A similar theme recurs when Pavlopoulos et al. (2020) shows that context infusion does not easily increase the performance of classifiers in context of toxicity detection in text. Alternatively instead of using context directly as input we use it as a regularizer to improve the classifier by learning better contextual representations while training. During inference, we use only the post text, thus adding no computation overhead.

We find that using self-supervision for hate speech detection systems leveraging the associated context within a social network is heavily underexplored. s demonstrated by Mathew et al. (2020), learning in a broader socio-personal context of the users and contemporary social situations is also very important. The authors showed the clustering tendency of hateful text sequences on specific hateful user timelines and after specific events (temporal clustering). However, to the best of our knowledge, no prior work in hate speech detection has explored self-supervision in these directions using the context information.

3 Proposed approaches

Assumptions about invariance(s) are required in all self-supervised learning objectives. We make the following two assumptions articulated below.

(i) In the masked language modeling (MLM) objective, the invariant is that the conditional probability distribution of the vocabulary tokens (or words) for each masked token can be reasonably estimated given the context (in the form of tokens) around the masked token.

(ii) In the User Anchored self-supervision (UA) objective, we assume that the users' writing style and bias (specifically cultural and in-group bias) are invariant (Hughes et al., 2012). Hence, the inverse mapping of a post to the corresponding user should be estimable subject to the language understanding capability.

We denote the model being trained for the downstream tasks as $\mathcal{M}$. $\mathcal{M}$ has two modules: **(a)** an encoder for encoding the input sentences, and **(b)** a classifier or regressor for mapping the hidden representations generated by the encoder into one of $\mathcal{K}$ classes and a single value respectively. For instance, the encoder in $\mathcal{M}$ can be modeled by a transformer (Vaswani et al., 2017). See Figure 2 for block diagram of our proposed approaches.

3.1 *Phase I*: Incorporating hateful content into the language model through continual pre-training (CP)

We start with pre-trained language models and continue pre-training them by minimizing the standard MLM objective using text sequences (posts) accumulated from a variety of social network datasets. The procedure for domain adaptation in LMs can be found in Gururangan et al. (2020) and a procedure tailored for hateful words can be found in HateBERT. We use the text sequences available on Pushshift[2] in addition to RAL-E from HateBERT for pre-training our Hate-infused Language Model (HateLM).

3.2 *Phase II*: User Anchored self-supervision (UA)

Some users are more biased than others and hence are more prone to post toxic content (Mathew et al., 2020). We employ a User Anchored self-supervision (UA) objective in the next pre-training phase to compare the users' writing style. Hence, given an LM is capable of language understanding, it should also be able to distinguish between users' writing styles from their posts with high probability, when the pool of posts from various users is not very large.

Here we use a self-supervised method based on *contrastive pre-training* with negative samples to efficiently incorporate UA into an LM. This negative sampling makes the approach highly scalable[3].

For the text sequence $\mathcal{S}$ corresponding to the i^{th} sampled post and user u_0, we try to decrease some distance metric between representation of $\mathcal{S}$ and another sentence by u_0 among the n_u available users while increasing the distance metric with sentences authored by other users in a contrastive fashion.

[2] https://files.pushshift.io/gab

[3] Owing to the extremely large number of users in a platform and the dynamic nature of the graph, it is infeasible to simply add a user classification network after the encoder

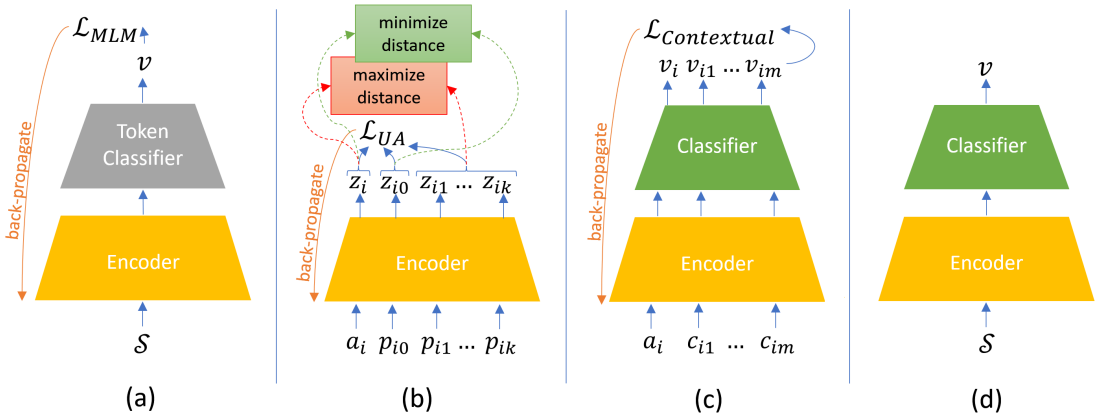

Figure 2: [Best viewed in color] An illustration of the various phases of training and inference for the classification task. The regression task will have a similar structure except for a regressor head instead of a classifier head in *(c)* and *(d)*. The blue arrows indicate the forward pass, and the orange arrows indicate the backward pass. *(a)* corresponds to Phase I, i.e. continual pre-training using self-supervised MLM objective on hateful sequences to incorporate contextual hate understanding. *(b)* corresponds to Phase II, i.e. the User Anchored self-supervised learning objective, it depicts the scalable contrastive objective model that does not add any additional parameters. *(c)* corresponds to Phase III, i.e. our contextual regularization procedure where we add post and user context. And finally *(d)* shows the inference phase which does not require any additional context.

Next, we sample $k << n_u$ number of users uniformly at random, without repetition, from among the list of all users in the social network. Adding u_0 along with the k sampled users in a set, we get:

$$U_i = \{u_{i0}, u_{i1}, ..., u_{ik} | \forall_{j_1, j_2 \in \{0,..,k\}} u_{ij_1} \neq u_{ij_2}\} \quad (1)$$

The set of posts P_i made by users in U_i.

$$P_i = \{p_{i0}, p_{i1}, ..., p_{ik} | p_{i0} \neq \mathcal{S}\} \quad (2)$$

p_{i0} cannot be same as $\mathcal{S}$ because our positive sample needs to be different from the original post. The remaining $\forall_{j \in [k]}\{p_{ij}\}$ become negative samples.

For Phase II pre-training and updating the parameters $\theta_\mathcal{E}$ of the encoder in $\mathcal{M}$, we start by sampling an anchor sequence $a_i \equiv \mathcal{S}$ from our dataset. We pass a_i through encoder in $\mathcal{M}$ to obtain z_i.

$$z_i = \mathcal{F}_{\theta_\mathcal{E}}(a_i) \quad (3)$$

Next, we generate P_i by sampling as described above. We pass all of the p_{ij}'s $\in P$ through our encoder as well, to get embeddings:

$$Z_i = \{z_{i0}, z_{i1}, ..., z_{ik}\} \quad (4)$$

We use a self-supervised contrastive loss inspired from Khosla et al. (2020) and modify it for our purposes:

$$\mathcal{L}_{\text{UA}} = \mathop{\mathbb{E}}_{i \sim \mathcal{U}([N])} \left[-log \left(\frac{\exp(z_i \cdot z_{i0})}{\sum_{j \in [k]} \exp(z_i \cdot z_{ij})} \right) \right] \quad (5)$$

Here, '$\cdot$' represents the inner product and $\mathcal{U}(.)$ represents discrete uniform distribution. Since there are no additional models, only $\theta_\mathcal{E}$ parameters need to be updated during backpropagation.

***Robust UA*:** During training UA objective exhibits the problem of over-fitting. After a certain number of epochs, the information in the language model accumulated during the CP phase gets overshadowed by information obtained from the UA phase. In order to handle the problem, the model needs to be fine-tuned after every epoch. Since this is not feasible, we propose a method for a more robust training method aligned to our downstream task of hate speech classification. Since we already have the annotations in the target dataset, we add an auxiliary task to align the model parameter updates towards the downstream task rather than diverge as a consequence of over-fitting.

For an anchor sequence $a_i \equiv \mathcal{S}$ sampled from the training set, we get the original class label of the example and get a positive example p by sampling a new post from the training set belonging to the same class. Similarly, we sample two posts from each of the remaining classes in order to get a set of $l = 2(n_c - 1)$ negative examples, where n_c is the number of classes in the downstream classification task. Hence similar to P_i in Equation 2, our auxiliary post set corresponding to the i^{th} sampled sequence is as follows:

$$\overline{P}_i = \{\overline{p}_{i0}, \overline{p}_{i1}, ..., \overline{p}_{il} | \overline{p}_{i0} \neq \mathcal{S}\} \quad (6)$$

Subsequently, we pass them through the encoder in $\mathcal{M}$ and generate auxiliary embedding set:

$$\overline{Z}_i = \{\overline{z}_{i0}, \overline{z}_{i1}, ..., \overline{z}_{il}\} \quad (7)$$

Similar to the supervised contrastive loss described in Equation 5, we add the following auxiliary loss:

$$\mathcal{L}_{\text{aux}} = \mathop{\mathbb{E}}_{i\sim\mathcal{U}([N])} \left[-log \left(\frac{\exp(z_i \cdot \overline{z}_{i0})}{\sum_{j\in[l]} \exp(z_i \cdot \overline{z}_{ij})} \right) \right] \tag{8}$$

Here, it is the same i that was being sampled in Equation 5. Finally we take a convex combination of both the losses to get:

$$\mathcal{L}_{\text{RobustUA}} = \lambda \mathcal{L}_{\text{UA}} + (1-\lambda)\mathcal{L}_{\text{aux}} \tag{9}$$

where, $0 < \lambda < 1$ is a hyper-parameter.

Note: We do not have classes for a regression task. Therefore, we group the labels into $\mathcal{K}$ clusters by using K-means clustering where $\mathcal{K}$ is a hyper-parameter. Then we use the associated cluster labels as a proxy for the class labels.

3.3 *Phase III*: Contextual Regularization (CR)

Our primary assumption in this approach is that: A post is influenced by its context. We demonstrate how to exploit the intuition "hate begets hate" (Mathew et al., 2020) to our advantage.

3.3.1 CR in classification

HateLM and UA, i.e., the methods described in section 3.1 and 3.2 (with the exception of Robust UA training) were self-supervised; hence, no information about the annotations available in the training set was used. After getting the pre-trained model in Phase II, we fine-tune using the annotations available for the dataset. We consider a pair of text sequence and its corresponding label $\langle \mathcal{S}, \text{y} \rangle$ sampled from the training set uniformly at random, where $\text{y} \in [\mathcal{K}]$ denotes the true class label of the sampled sequence $\mathcal{S}$. Usually, we would get the vector embedding z by passing $\mathcal{S}$ through the encoder. We would then have used the classifier in $\mathcal{M}$ to get the vector v of $\mathcal{K}$ dimensions as follows:

$$v = \mathcal{F}_{\mathcal{C}}(z; \theta_{\mathcal{C}}) \tag{10}$$

where $\theta_{\mathcal{C}}$ parameterizes the classifier in $\mathcal{M}$. According to the model, the probability of the sample belonging to j^{th} class among $\mathcal{K}$ classes would be as follows:

$$\mathbb{P}[class(\mathcal{S}) = j] = \frac{\exp(v_j)}{\sum_{k\in[\mathcal{K}]} \exp(v_k)} \tag{11}$$

and the most likely class assignment would be

$$\hat{\text{y}} = \arg\max_{j\in[\mathcal{K}]} \mathbb{P}[class(\mathcal{S}) = j] \tag{12}$$

The cross-entropy loss would be calculated as:

$$L_{\text{CE}} = \mathop{\mathbb{E}}_{i\sim\mathcal{U}([N])} [-log\, \mathbb{P}[class(\mathcal{S}) = \text{y}]] \tag{13}$$

We propose an additional method for regularization of the model using the contextual information during fine-tuning. For the given $\mathcal{S}$ we sample at most n_a posts from the same thread (all comments/replies concerning the parent post) where the comment is posted (post context) and at most n_b more posts from the timeline of the user who posted $\mathcal{S}$ (user context). Both post and user context is sampled without replacement. So, we generate a context set C of $\mathcal{S}$ for the i^{th} text sequence sampled from the dataset:

$$C_i = \{c_{i1}, ..., c_{in_a}, c_{i(n_a+1)}, ..., c_{i(n_a+n_b)}\} \tag{14}$$

We then generate the vector v using $\mathcal{S}$ as mentioned in Equation 10. Next we generate V_i from C_i:

$$V_i = \{v_{i1}, ..., v_{im} | \forall_{j\in[m]} v_{ij} = \mathcal{M}(c_{ij})\} \tag{15}$$

where $m = n_a + n_b$ and, $\mathcal{M}(.) \equiv \mathcal{F}_{\theta_{\mathcal{C}}}(\mathcal{F}_{\theta_{\mathcal{E}}}(.))$. Our contextual loss from V_i is as follows:

$$\mathcal{L}_{CCE,i} = \frac{-1}{m} \sum_{j\in[m]} log \left(\frac{\exp(v_{ijt})}{\sum_{k\in[\mathcal{K}]} \exp(v_{ijk})} \right) \tag{16}$$

where t is the true class of the original labelled post. Therefore the auxiliary contextual cross-entropy loss is calculated as:

$$\mathcal{L}_{\text{CCE}} = \mathop{\mathbb{E}}_{i\sim\mathcal{U}([N])} [\mathcal{L}_{CCE,i}] \tag{17}$$

Our final contextual regularization loss function is a linear combination of the losses mentioned in Equation 13 and 17 as described below:

$$\mathcal{L}^{\text{classification}}_{\text{Contextual}} = \lambda \mathcal{L}_{\text{CE}} + (1-\lambda)\mathcal{L}_{\text{CCE}} \tag{18}$$

Back-propagating this loss we update the parameters $\theta_{\mathcal{C}}$ and $\theta_{\mathcal{E}}$, corresponding to the classifier and the encoder in $\mathcal{M}$ respectively, as one might have done normally.

3.3.2 CR in regression

The regression task is almost similar to classification as mentioned in section 3.3.1 with the exception that the $\text{y} \in \mathbb{R}$ in the tuple $\langle \mathcal{S}, \text{y} \rangle$ sampled from the training set, and the output of the model $\mathcal{M}$ being a real value r rather than a vector v of $\mathcal{K}$ dimensions. Since we use a regressor, Equation 10 changes to

$$r = \mathcal{F}_{\mathcal{R}}(z; \theta_{\mathcal{R}}) \tag{19}$$

where $\theta_{\mathcal{R}}$ parameterizes the regressor in $\mathcal{M}$.

The cross-entropy loss in Equation 13 becomes

Dataset	Annotated samples	Total users	Labels	Max context sampled (per user)
HateXplain	10,000	3,300	Hate speech, Offensive, Normal	100
LTI-GAB	30,500	4,900	Toxic and Non-Toxic	50
Ruddit	6,000	4,200	Regression (-1 to +1)	20

Table 1: Dataset statistics for the datasets we used for performing the experiments along with the additional unsupervised data we collected. All datasets are in the English language.

squared loss as follows:

$$L_{\text{MSE}} = \mathop{\mathbb{E}}_{i \sim \mathcal{U}([N])} \left[(\mathrm{y}_i - r_i)^2 \right] \quad (20)$$

where y_i is the label associated with the ith example sampled from the training dataset, r_i is the predicted value by model $\mathcal{M}$.

The techniques for selection of the context discussed in section 3.3.1 remain unaltered since our post and user context is unsupervised. However, the set of vectors V_i becomes a set of real values R_i generated as follows:

$$R_i = \{r_{i1}, ..., r_{im} | \forall_{j \in [m]} r_{ij} = \mathcal{M}(c_{ij})\} \quad (21)$$

And our auxiliary contextual mean squared loss is calculated as the follows:

$$\mathcal{L}_{CMSE} = \mathop{\mathbb{E}}_{i \sim \mathcal{U}([N])} \left[\frac{-1}{m} \sum_{j \in [m]} (\mathrm{y}_i - r_{ij})^2 \right] \quad (22)$$

Our final contextual regularization loss function is thus a linear combination of the losses mentioned in Equation 20 and 22 as described below:

$$\mathcal{L}^{\text{regression}}_{\text{Contextual}} = \lambda \mathcal{L}_{\text{MSE}} + (1 - \lambda) \mathcal{L}_{\text{CMSE}} \quad (23)$$

4 Experiments

We experiment with the downstream task of hate speech detection. We establish the effectiveness of our proposed approaches by demonstrating that they are: **(i)** mutually independent of each other, **(ii)** independent of the base language model used, and **(iii)** applicable across various social network datasets. We validate these claims by comparing our approaches with the following baselines:

(i) To establish mutual independence among the training phases, we train a base language model (BERT from Devlin et al. (2019)) with each of the training phases separately. An improvement over the performance of the base model in each phase would indicate the advantage of our approach over naive training.

(ii) To establish independence from the base language model, we train multiple baseline language models during each of the above-mentioned separate training phases (HateBERT during Phase I; BERT and HateXplain during the remaining phases). Consistent performance improvement irrespective of the language model would again indicate the advantage of our approach.

(iii) We use three datasets from two different social networks and two types of tasks (classification and regression). This establishes the capability of our methods to solve a range of heterogeneous tasks across contrasting datasets.

We next describe the datasets and the implementation details of our approaches.

4.1 Datasets

Social networks: In particular we experiment with two popular social networks (a) `gab.ai` (GAB), and (b) `reddit.com` (Reddit). Our choice of the dataset was guided by the availability of the context graph and additional data collection time. Baumgartne (2018) had scraped GAB and made the network freely available for academic use. On the other hand, Reddit has an API[4] available to get the public domain data. Therefore, these websites were favorable for our experiments. Since Reddit involved additional data collection (a time consuming process), we chose a popular dataset that contains less than 10,000 datapoints.

Annotated hate speech data: We use the following english hate speech datasets for our experiments (See Table 1 for more information on dataset statistics) – (i) HateXplain-GAB dataset (Mathew et al., 2021) (contains data from GAB), (ii) LTI-GAB dataset (Qian et al., 2019) (contains data from GAB) and, (iii) Ruddit (Hada et al., 2021) (contains data from Reddit).

We use *only* the GAB subset of the annotated data from both the datasets (i) and (ii), because the social network context graph for GAB is publicly available. The GAB subset of dataset (i) has around 10K annotated data samples, which are already divided into 80-10-10 train-val-test split. Dataset (i) has three class annotations (hate speech, offensive, and normal). Dataset (ii) contains intervention sen-

[4] https://www.reddit.com/dev/api/

Phase	Model	HX-GAB[5] (3 class)		LTI-GAB (2 class)		Ruddit (regression)	
		Acc	Macro-F1	Acc	F1 (toxic)	MSE	MAE
Existing approaches as baselines	TF-IDF	0.6337	0.5604	0.8993	0.8824	0.1128	0.2651
	BERT	0.6763	0.6376	0.9141	0.9019	0.1041	**0.2521**
	HateXplain†[5]	**0.6905**	**0.6511**	**0.9177**	**0.9062**	**0.1036**	**0.2520**
Continual pre-training (CP phase)	HateBERT	0.6843	0.6458	0.9166	0.9040	0.1029	0.2516
	HateLM	**0.6882**	**0.6493**	**0.9174**	**0.9058**	**0.1018**	**0.2474**
User Anchored self-supervision (UA phase)	BERT + UA	0.6950	0.6632	0.9192	0.9089	0.0994	0.2367
	HateXplain + UA	0.7058	0.6680	0.9211	0.9105	0.0989	0.2329
	HateLM + UA	**0.7087**	**0.6711**	**0.9215**	**0.9117**	**0.0958**	**0.2229**
Contextual Regularization (CR phase)	BERT + CR	0.6819	0.6452	0.9176	0.9051	0.1019	0.2438
	HateXplain + CR	**0.6935**	0.6555	**0.9199**	0.9088	0.1011	0.2410
	HateLM + CR	0.6924	**0.6558**	**0.9200**	**0.9096**	**0.0995**	**0.2342**
UA and CR phase together	BERT + UA + CR	0.7017	0.6673	0.9211	0.9104	0.0968	0.2314
	HateXplain + UA + CR	0.7099	0.6702	0.9236	0.9122	0.0955	0.2291
	CRUSH*	***0.7133***	***0.6749***	***0.9234***	***0.9149***	***0.0921***	***0.2188***

Table 2: **Experimental outcomes of our approaches.** Our model **CRUSH** ≡ HateLM + UA + CR. For reporting results (except CR and UA+CR phases) the encoder models were attached with a classifier (refer section 4.2) and fine-tuned (using the loss mentioned in Equation 13). *Across the columns*: we show the results with three datasets. In HateXplain (HX)'s and Learning to Intervene (LTI)'s results, higher metrics are better. In Ruddit results, lower metrics are better. *Across the rows*: we show the various models grouped by the *phase* of training along with corresponding baselines. In each group, we have indicated the best-performing models' results corresponding to the evaluation metric in bold. The overall best performing models' results have additionally been italicized. **The models denoted by † (the best competing baseline) and ⁎ (CRUSH) are significantly different (M-W U test with $p < 0.05$) across all datasets.**

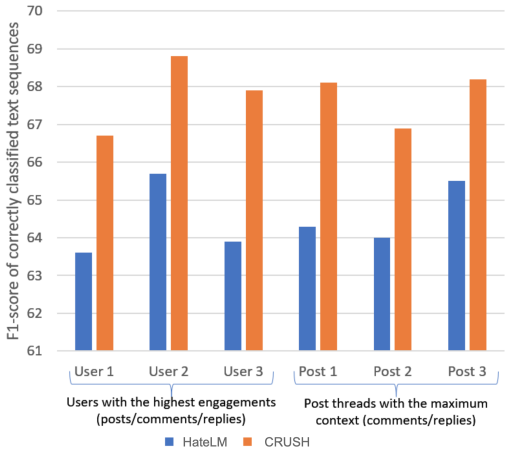

Figure 3: [Best viewed in color] F1-scores of correctly classified posts grouped by three user contexts and three post contexts. The improvement along the user contexts and post contexts demonstrate the utility of the CR phases (user & post context based).

tences along with which sentences to intervene and a binary label - hate and non-hate. We use a 90-10 train-test split with the random seed 2021, and among the training set we use 10% randomly sampled data for validation. Dataset (iii) was collected from Reddit and is labeled for regression task. The dataset contains ratings corresponding to the measure of hate within a sentence ranging between -1 and 1, with higher numbers corresponding to a higher extent of hate. There are about 6K examples in this dataset. We use a train test split similar to that we used in the dataset (ii).

Our pre-processing procedure for all the textual data, both labeled and unlabelled is exactly the same as that of HateXplain. (Mathew et al., 2021). **Unlabelled data for self-supervision**: We get the threads corresponding to the annotated Reddit datapoints using the Reddit API and use that as our unlabelled corpora for self-supervision in case of the regression task. For GAB, we use the whole network datadump available on Pushshift (Baumgartne, 2018) as mentioned in Section 3.2.

4.2 Implementation details

For our encoder, we use the pre-trained `bert-base-uncased` that is available on huggingface [6]. By continual pre-training on this model, we obtain our HateLM using the text sequences from GAB and RAL-E as described in section 3.1. Afterward, we obtain a 768 dimensional pooled output from our encoder which is then be passed onto the classifier/regressor head. For both our classifier and regressor model, we select a neural

[5]HateXplain refers to both a dataset and the corresponding model provided in the paper. However, since we only use the GAB subset of the HateXplain dataset, we require to fine-tune the HateXplain model on this subset, for it to be comparable to other models, and therefore it is considered a baseline.

[6]https://huggingface.co/bert-base-uncased

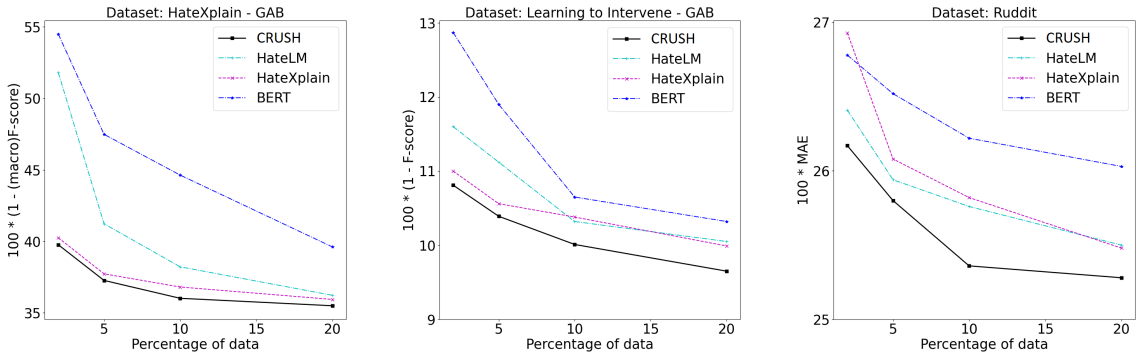

Figure 4: [Best viewed in color] Comparison of model error in low data regime across the three datasets. Our model (black solid line) consistently outperforms existing approaches that do not use the social network context. For plots showing comparisons with non-contextualized LMs (e.g., TF-IDF) as well, refer to Figure 5 in Appendix.

Word	BERT	HateLM
black	everywhere, free, not	racist, evil, stupid
muslim	accepted, optional, allowed	terrorist, evil, shit
jews	persecuted, excluded, present	idiots, bad, stupid
men	equal, free, citizens	evil, people, bad
women	excluded, only, bias	dumb, evil, stupid

Table 3: Top-3 next word predictions by each LM when the corresponding word is given as prompt. The hateful words as the output of the next word prediction demonstrate the effectiveness of the hate-infusion into LM.

network consisting of a 2 Fully-connected layers. The hidden layer for the same is chosen to have 128 dimensions along with a ReLU activation. The input dimension is dependent on the output of the encoder, and in our case is set to 768. The output dimension is 1, 2 or 3 corresponding to the regressor, binary and ternary classifier. We used *Adam* optimizer with batch size 48, max seq length 128 and learning rates $3e$-6 and $2e$-5 for our encoder and classifier/regressor respectively. Our approaches do not require additional context during inference (see Figure 2(d)).

4.3 Ablation studies

We frame the ablation studies to answer a series of interesting questions. These are:

Q1: How much does self-supervised learning objective during pre-training help in the downstream task (if at all)?

Q2: Similarly, how much does contextual regularization during fine-tuning help?

Q3: Do these two approaches work only in conjunction with each other?

In addition to the above, we perform the following ablation experiments to validate the usefulness of various components of our model:

Q4: How much of the performance can be attributed to the detection of isolated hateful words without any contextual information in place? To answer this question, we use the TF-IDF (Sammut and Webb, 2010) vector embedding of each text sequence. since TF-IDF computes representations solely based on the frequency of words without any contextual information.

Q5: Have the hateful words been contextually incorporated into the hate-infused language model? Here, we prompt our hate-infused language model and study the next word it predicts.

Q6: How do these approaches perform in a few-shot setting?

5 Results

The Table 2 compares the performance of our methods with related models and baselines. We report the performance metrics according to the test sets of the corresponding datasets.

[1] It is evident from Table 2 that both CP and UA phases (self-supervised phases) lead to improvements over the baselines. This answers **Q1**. To answer how useful regularization was we can look at the CR phase results which are again better than the baselines thus answering **Q2**.

Further, we can also see that UA and CR phase results individually beat the baseline. Furthermore, combined UA+CR phase outperforms all models. Hence, the conjunction hypothesis is valid thus answering **Q3**.

Post (along with ground-truth label)	BERT	HateLM	CRUSH
if your humor is based on racism homophobia sexism and rape you're not f**king funny go home (**non-hate**)	✗	✓	✓
<user> ah man i f**king hate you so much (**hateful**)	✗	✓	✓
the french government position is that france is made stronger by the immigrants it is bribing to leave porter (**non-hate**)	✗	✗	✓
i am presuming he means the standard left wing idiots lots of hypocritical women amongst them think maria ladenburger obviously you are against these muhammedans (**hateful**)	✗	✗	✓
a covington catholic a native american indian and a black hebrew israelite walk into a bar (**non-hate**)	✗	✗	✗
if money was grown on trees women would be dating monkeys oh wait never mind (**hateful**)	✗	✗	✗

Table 4: Qualitative results using text sequences from HateXplain dataset. ✓ indicates sentences were classified properly by the corresponding models while ✗ indicates incorrect classification. MLM based training of BERT and HateLM does not seem to capture the complexities of hate speech properly, thus CRUSH outperforms them.

[2] **Q4** is answered by the considerable improvement over the TF-IDF baseline as can be seen in Table 2. In addition, Figure 3 shows the advantages gained by using CRUSH over HateLM. It demonstrates both the user-level and the post-level contextual information from the network incorporated into our CRUSH model are individually helpful over models that do not have social network context information.

[3] Table 3 answers **Q5** by showing some of the prompt completion results for BERT vs our Hate-infused LM. It can be noticed that although HateLM definitely has been incorporated with racial and ethnic hate, it surprisingly does not discriminate between genders.

[4] Figure 4 shows the few-shot training results with small percentages of data sampled from our training datasets (2% to 20% data points) uniformly at random. Intuitively our model is already able to outperform all other baselines consistently even without context because of the self-supervised training procedures we propose during the CP and UA phases which captures the hate & users' style+bias. This is evident from the results of CP & UA phases in Table 3. So later adding context in few-shot setting simply further enhances our model thus answering **Q6** (see Appendix for full results).

Discussion of some qualitative examples: Table 4 presents a few text sequences from the HateXplain dataset and their corresponding results (classification/misclassification) for the models – BERT, HateLM, and CRUSH. It can be noticed that the first two sentences are properly classified by HateLM and CRUSH but not by vanilla BERT. That leads us to believe that BERT predictions are heavily dependent on the token occurrences rather than the context in the sentence where the tokens have occurred. As for the next two sentences, it is evident that the sentence representations learnt by CRUSH (due to UA + CR phases of training, see Section 3.2, 3.3) are superior to those learnt by the HateLM, hence it classifies these text sequences better than simple MLM based training. Finally, the last two sentences indicate that CRUSH (along with the other approaches) still lacks the ability to identify humor, sarcasm, and implicit hate speech, which are known to be difficult problems.

6 Conclusion

In this paper, we provide approaches to infuse social network context using the self-supervised user-attribution pre-training objective combined with the contextual regularization objective on top of traditional MLM for hate speech tasks. We empirically demonstrate the advantage of our methods by improving over existing competitive baselines in hate speech detection and scoring tasks across three different datasets. We also show that our method performs superior in the low data regime as well when compared to existing approaches. We also do ablations to understand the benefits of each objective separately. Future work include exploiting the relations among users, using different base models capable of incorporating longer contexts, and trying to address hard problems like sarcasm and implicit hate speech detection in social networks.

7 Ethical considerations

All the datasets that we use are publicly available. We report only aggregated results in the paper. For context mining, we have used data either available in the public domain or available through official APIs of the corresponding social media. Neither have we, nor do we intend to share any personally identifiable information with this paper. We also make our codebase publicly available here - https://github.com/parag1604/CRUSH.

Our model CRUSH helps advance the state-of-the-art in hate speech detection by incorporating continual pre-training, capturing user writing biases and leveraging both user & post context (which is publicly available as well). This in turn should be able limit the amount of hateful threads in social networks/media by better detection of hate speech, thus promoting a more friendly and welcoming environment for people from all race, religion, ethnicity, gender etc.

Unfortunately, these models are not completely free from all potential negative impact. One such example being that our models have hate knowledge infused within them during the CP phase (refer Section 3.1) of our training pipeline. As shown in Table 3, these language models could be used potentially used for generating hateful words/sentences given an initial prompt.

However, the hateful words generated by the HateLM can be also identified by the platform if the platform uses a hate speech detection algorithm like ours or uses a set of hate lexicons to directly filter out such keywords generated (Gitari et al., 2015). Moreover, our final model - CRUSH - is not exactly suitable for toxic language generation as it is further pre-trained on the user style discrimination task and fine-tuned on the hate speech classification task making its classification capabilities (potential use by the social media platforms to detect hate speech) stronger than the hate speech generation capabilities (the potential of the model being abused with malicious intent). So, the platforms using our better-informed model will easily detect any hate speech generated by the model.

Moreover, large-scale technologies for better generation of hate speech using GPT-2 and other generative models (Wullach et al., 2021; Hartvigsen et al., 2022) (with the intention to train hate speech classifiers better) already exist and as a model trained only for classification, our model is likely to be far weaker than these models for generation of harmful content. While these language models can be used both to benefit or disrupt our lifestyle just like any other technology (Douglas, 2014), we urge the researchers to exercise ultimate caution while using them, including ours. For the same reason, we do not make the trained model parameters publicly available (except for the code to promote reproducibility).

Also, one of the factors increasing bias in the classification model is the data it is trained upon. Hence, any potential bias in the datasets that are manually annotated by human beings (who are not free from bias) can result the model being biased for/against some specific target groups.

To incorporate better inductive bias into the model, we have trained the model to initially map text sequences posted by the same user to vectors that are close in the embedding space. This helps the model to identify the dialects of the various users and help the model perform better. However, this inductive bias may cluster the linguistic characteristics of some groups which in turn might potentially increase the chances of that particular language style being classified as hate speech if the annotated data contains only hateful instances from that particular dialect. This may make the model biased if those particular dialects are used predominantly by some protected category (Blacks, Jews, Women, Mexican etc.).

The most simple and effective solution here would be to annotate data for each dialect/vernacular group, potentially stratifying the dataset into clusters using the pre-trained language model. If there are balanced examples from each of these dialect/vernacular or at least each protected category (Blacks, Jews, Women, Mexican etc.) for both the hate and non-hate categories, such biases can be well mitigated.

Further, our model like any other hate speech model is not suitable for in-the-wild deployment without explicit human scrutiny. Language use is often very specific to each platform. So, the distribution of words and data-points may not match the training data distribution of the model. Thus, it is extremely important to first test the model on the platform, check for potential biases against the protected categories and individual linguistic groups/dialects and deploy it as a preliminary filter assisting the human experts detecting hate speech.

8 Acknowledgements

We thank Facebook (Meta Platforms, Inc.) for sponsoring the stay of PD at IIT Kharagpur through the Ethics in AI Research grant and Tata Consultancy Services (TCS) for funding SC with the TCS PhD fellowship.

References

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, WWW '17 Companion, page 759–760, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Jason Baumgartne. 2018. gab.ai corpora on pushshift.io. https://files.pushshift.io/gab/. Added: 2018-11-30, Announcement: https://twitter.com/jasonbaumgartne/status/1068610394992926720.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Marco Del Tredici, Diego Marcheggiani, Sabine Schulte im Walde, and Raquel Fernández. 2019. You shall know a user by the company it keeps: Dynamic representations for social media users in NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4707–4717, Hong Kong, China. Association for Computational Linguistics.

Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook. In *Proceedings of the First Italian Conference on Cybersecurity, ITASEC 2017*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Douglas. 2014. The dual-use problem, scientific isolationism and the division of moral labour. *Monash bioethics review*, 32:86–105.

Njagi Dennis Gitari, Zhang Zuping, Zuping Zhang, Hanyurwimfura Damien, and Jun Long. 2015. A lexicon-based approach for hate speech detection. In *International Journal of Multimedia and Ubiquitous Engineering*, volume 10, pages 215–230.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Rishav Hada, Sohi Sudhir, Pushkar Mishra, Helen Yannakoudakis, Saif M. Mohammad, and Ekaterina Shutova. 2021. Ruddit: Norms of offensiveness for English Reddit comments. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2700–2717, Online. Association for Computational Linguistics.

Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. 2022. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv preprint arXiv:2203.09509*.

James M. Hughes, Nicholas J. Foti, David C. Krakauer, and Daniel N. Rockmore. 2012. Quantitative patterns of stylistic influence in the evolution of literature. *Proceedings of the National Academy of Sciences*, 109(20):7682–7686.

Andrew Jakubowicz. 2017. Alt_right white lite: Trolling, hate speech and cyber racism on social media. *Cosmopolitan Civil Societies: An Interdisciplinary Journal*, 9:41.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Rohan Kshirsagar, Tyrus Cukuvac, Kathy McKeown, and Susan McGregor. 2018. Predictive embeddings for hate speech detection on twitter. In *Abusive Language Online Workshop, EMNLP 2018*, pages 26–32.

Shervin Malmasi and Marcos Zampieri. 2017. Detecting hate speech in social media. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 467–472, Varna, Bulgaria. INCOMA Ltd.

Binny Mathew, Anurag Illendula, Punyajoy Saha, Soumya Sarkar, Pawan Goyal, and Animesh Mukherjee. 2020. Hate begets hate: A temporal study of hate speech. *Proc. ACM Hum.-Comput. Interact.*, 4(CSCW2).

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):14867–14875.

Stefano Menini, Alessio Palmero Aprosio, and Sara Tonelli. 2021. Abuse is contextual, what about nlp? the role of context in abusive language annotation and detection. *ArXiv*, abs/2103.14916.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.

Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A benchmark dataset for learning to intervene in online hate speech. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4755–4764, Hong Kong, China. Association for Computational Linguistics.

Axel Rodriguez, Carlos Argueta, and Yi-Ling Chen. 2019. Automatic detection of hate speech on facebook using sentiment and emotion analysis. In *International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2019*, pages 169–174.

Claude Sammut and Geoffrey I. Webb, editors. 2010. *TF–IDF*, pages 986–987. Springer US, Boston, MA.

Serra Sinem Tekiroğlu, Yi-Ling Chung, and Marco Guerini. 2020. Generating counter narratives against online hate speech: Data and strategies. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1177–1190, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Bertie Vidgen, Dong Nguyen, Helen Margetts, Patricia Rossini, and Rebekah Tromble. 2021. Introducing CAD: the contextual abuse dataset. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2289–2303, Online. Association for Computational Linguistics.

Zeerak Waseem. 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

Tomer Wullach, Amir Adler, and Einat Minkov. 2021. Fight fire with fire: Fine-tuning hate detectors using large samples of generated hate speech. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4699–4705, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ziqi Zhang, D. Robinson, and Jonathan Tepper. 2018. Detecting hate speech on twitter using a convolution-gru based deep neural network. In *Extended Semantic Web Conference, ESWC 2018*.

A Appendix

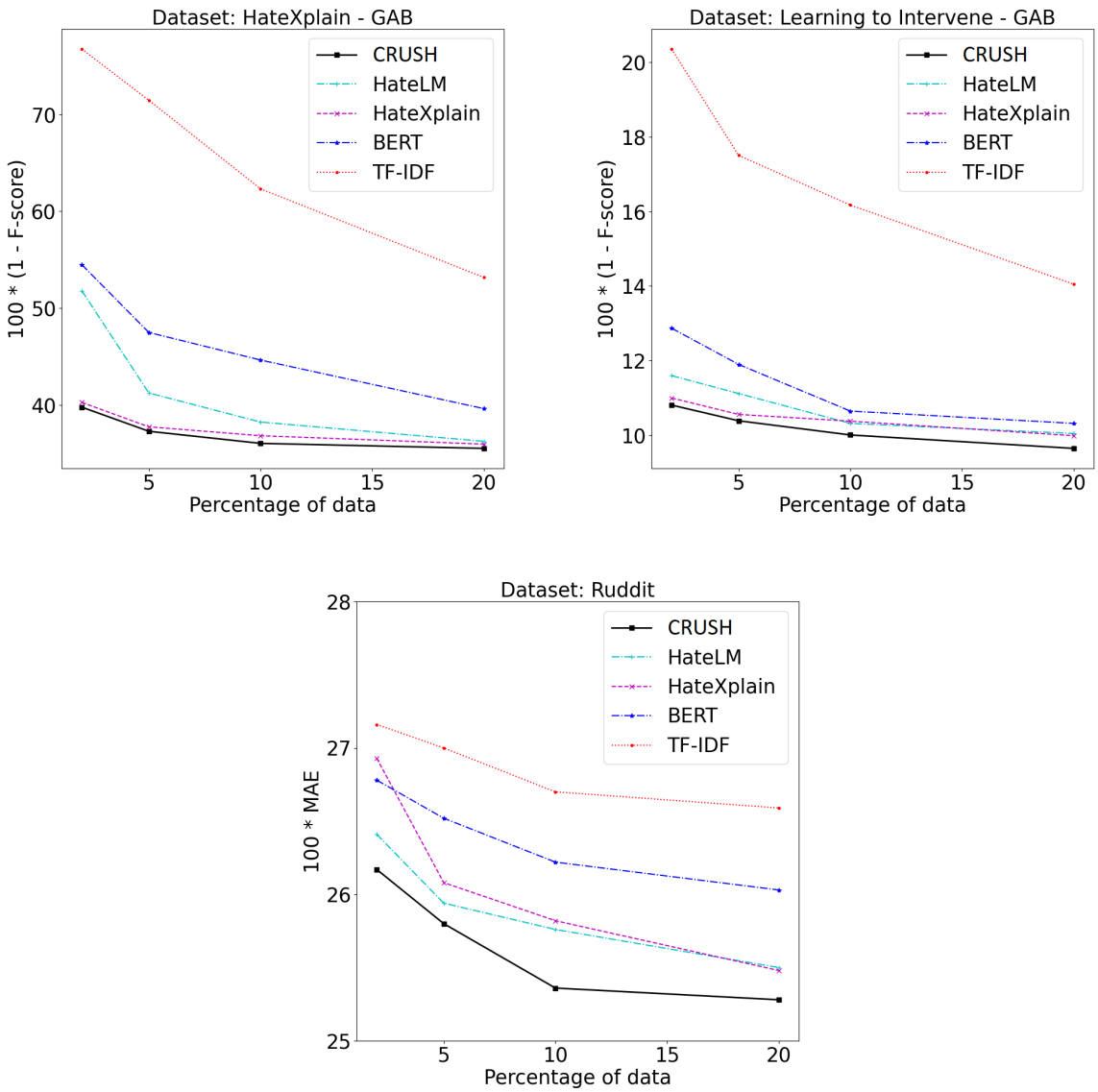

Figure 5: Full comparison of model errors in low data regime. This includes a comparison of our CRUSH model with baseline models built on non-contextualized and contextualized embeddings.