

Hierarchical CVAE for Fine-Grained Hate Speech Classification

Jing Qian, Mai ElSherief, Elizabeth Belding, William Yang Wang

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106 USA

{jing-qian, mayelsherief, ebelding, william}@cs.ucsb.edu

Abstract

Existing work on automated hate speech detection typically focuses on binary classification or on differentiating among a small set of categories. In this paper, we propose a novel method on a fine-grained hate speech classification task, which focuses on differentiating among 40 hate groups of 13 different hate group categories. We first explore the Conditional Variational Autoencoder (CVAE) (Larsen et al., 2016; Sohn et al., 2015) as a discriminative model and then extend it to a hierarchical architecture to utilize the additional hate category information for more accurate prediction. Experimentally, we show that incorporating the hate category information for training can significantly improve the classification performance and our proposed model outperforms commonly-used discriminative models.

1 Introduction

The impact of the vast quantities of user-generated web content can be both positive and negative. While it improves information accessibility, it also can facilitate the propagation of online harassment, such as hate speech. Recently, the Pew Research Center¹ reported that “roughly four-in-ten Americans have personally experienced online harassment, and 63% consider it a major problem. Beyond the personal experience, two-thirds of Americans reported having witnessed abusive or harassing behavior towards others online.”

In response to the growth in online hate, there has been a trend of developing automatic hate speech detection models to alleviate online harassment (Warner and Hirschberg, 2012; Waseem and Hovy, 2016). However, a common problem with these methods is that they focus on coarse-grained

¹<http://www.pewinternet.org/2017/07/11/online-harassment-2017/>

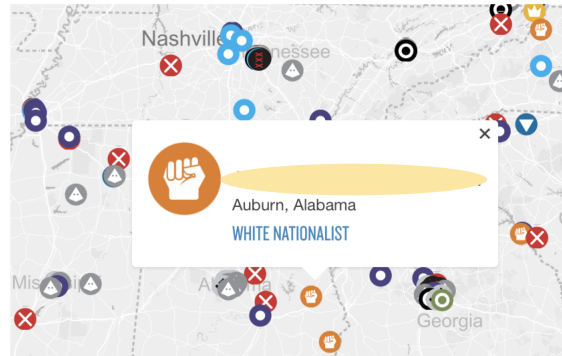


Figure 1: A portion of the hate group map published by the Southern Poverty Law Center (SPLC). Each marker represents a hate group. The markers with the same pattern indicate the corresponding hate groups share the same ideology. The white box shows an example of a hate group in Auburn, Alabama under the category of “White Nationalist”. Due to the sensitivity of the data, we mask the name of the group.

classifications with only a small set of categories. To the best of our knowledge, the existing work on hate speech detection formulates the task as a classification problem with no more than seven classes. Building a model for more fine-grained multiclass classification is more challenging since it requires the model to capture finer distinctions between each class.

Moreover, fine-grained classification is necessary for fine-grained hate speech analysis. Figure 1 is a portion of the hate group map published by the Southern Poverty Law Center (SPLC)², where a hate group is defined as “an organization that based on its official statements or principles, the statements of its leaders, or its activities has beliefs or practices that attack or malign an entire class of people, typically for their immutable characteristics.” The SPLC divides the 954 hate groups in the United States into 17 categories ac-

²<https://www.splcenter.org>

ording to their ideologies (e.g. Racist Skinhead, Anti-Immigrant, and others). The SPLC monitors hate groups throughout the United States by a variety of methodologies to determine the activities of groups and individuals: reviewing hate group publications and reports by citizens, law enforcement, field sources and the news media, and conducting their own investigations. Therefore, building automatic classification models to differentiate between the social media posts from different hate groups is both challenging and of practical significance.

In this paper, we propose a fine-grained hate speech classification task that separates tweets posted by 40 hate groups of 13 different hate group categories. Although CVAE is commonly used as a generative model, we find it can achieve competitive results and tends to be more robust when the size of the training dataset decreases, compared to the commonly used discriminative neural network models. Based on this insight, we design a Hierarchical CVAE model (HCVAE) for this task, which leverages the additional hate group category (ideology) information for training. Our contributions are three-fold:

- This is the first paper on fine-grained hate speech classification that attributes hate groups to individual tweets.
- We propose a novel Hierarchical CVAE model for fine-grained tweet hate speech classification.
- Our proposed model improves the Micro-F1 score of up to 10% over the baselines.

In the next section, we outline the related work on hate speech detection, fine-grained text classification, and Variational Autoencoder. In Section 3, we explore the CVAE as a discriminative model, and our proposed method is described in Section 4. Experimental results are presented and discussed in Section 5. Finally, we conclude in Section 6.

2 Related Work

2.1 Hate Speech Detection

An extensive body of work has been dedicated to automatic hate speech detection. Most of the work focuses on binary classification. Warner and Hirschberg (2012) differentiate between anti-semitic or not. Gao et al. (2017), Zhong et al. (2016), and Nobata et al. (2016) differentiate between abusive or not. Waseem and Hovy (2016),

Burnap and Williams (2016), and Davidson et al. (2017) focus on three-way classification. Waseem and Hovy (2016) classify each input tweet as racist hate speech, sexist hate speech, or neither. Burnap and Williams (2016) build classifiers for hate speech based on race, sexual orientation or disability, while Davidson et al. (2017) train a model to differentiate among three classes: containing hate speech, only offensive language, or neither. Badjatiya et al. (2017) use the dataset provided by Waseem and Hovy (2016) to do three-way classification. Our work is most closely related to (Van Hee et al., 2015), which focuses on fine-grained cyberbullying classification. However, this study only focuses on seven categories of cyberbullying while our dataset consists of 40 classes. Therefore, our classification task is much more fine-grained and challenging.

2.2 Fine-grained Text Classification

Our work is also related to text classification. Convolutional Neural Networks (CNN) have been successfully applied to the text classification task. Kim (2014) applies CNN at the word level while Zhang et al. (2015) apply CNN at the character level. Johnson and Zhang (2015) exploit the word order of text data with CNN for accurate text categorization. Socher et al. (2013) introduces the Recursive Neural Tensor Network for text classification. Recurrent Neural Networks (RNN) are also commonly used for text classification (Tai et al., 2015; Yogatama et al., 2017). Lai et al. (2015) and Zhou et al. (2015) further combine RNN with CNN. Tang et al. (2015) and Yang et al. (2016) exploit the hierarchical structure for document classification. Tang et al. (2015) generate from the sentence representation to the document representation while Yang et al. (2016) generate from the word-level attention to the sentence-level attention. However, the division of the hierarchies in our HCVAE is according to semantic levels, rather than according to document compositions. We generate from the category-level representations to the group-level representations. Moreover, the most commonly used datasets by these works (Yelp reviews, Yahoo answers, AGNews, IMDB reviews (Diao et al., 2014)) have no more than 10 classes.

2.3 Variational Autoencoder

Variational Autoencoder (VAE) (Kingma and Welling, 2013) has achieved competitive results in

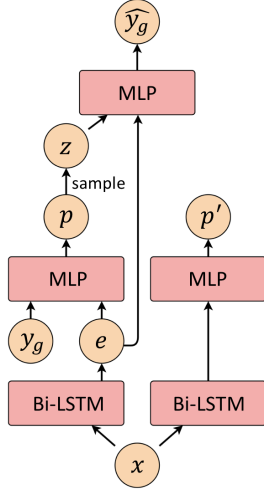


Figure 2: The structure of the baseline CVAE model during training. Bi-LSTM is a bidirectional LSTM layer. MLP is a Multilayer Perceptron. x is the embedded input text. y_g is the ground truth hate group label. \hat{y}_g is the output prediction of the hate group. p is the posterior distribution of the latent variable z while p' is the prior distribution of z . Note that this structure is used for training. During testing, the posterior network is replaced by the prior network to compute \hat{y}_g and thus y_g is not available during testing. Refer to Section 3 for detailed explanation.

many complicated generation tasks, such as handwritten digits (Kingma and Welling, 2013; Salimans et al., 2015), faces (Kingma and Welling, 2013; Rezende et al., 2014), and machine translation (Zhang et al., 2016). CVAE (Larsen et al., 2016; Sohn et al., 2015) is an extension of the original VAE framework that incorporates conditions during generation. In addition to image generation, CVAE has also been successfully applied to several NLP tasks, such as dialog generation (Zhao et al., 2017). Although so far CVAE has always been used as a generative model, we explore the performance of the CVAE as a discriminative model and further propose a hierarchical CVAE model, which exploits the hate group category (ideology) information for training.

3 CVAE Baseline

We formulate our classification task as the following equation:

$$Obj = \sum_{(x, y_g) \in X} \log p(y_g|x) \quad (1)$$

where x , y_g are the tweet text and hate group label respectively, X is the dataset. Instead of directly

parameterizing $p(y_g|x)$, it can be further written as the following equation:

$$p(y_g|x) = \int_z p(y_g|z, x)p(z|x)dz \quad (2)$$

where z is the latent variable. Since the integration over z is intractable, we instead try to maximize the corresponding evidence lower bound (ELBO):

$$ELBO = E[\log p(y_g|z, x)] - D_{KL}[q(z|x, y_g)||p(z|x)] \quad (3)$$

where D_{KL} is the KullbackLeibler (KL) divergence. $p(y_g|z, x)$ is the likelihood distribution, $q(z|x, y_g)$ is the posterior distribution, and $p(z|x)$ is the prior distribution. These three distributions are parameterized by $p_\varphi(y_g|z, x)$, $q_\alpha(z|x, y_g)$, and $p_\beta(z|x)$. Therefore, the training loss function is:

$$\begin{aligned} \mathcal{L} &= L_{REC} + L_{KL} \\ &= E_{z \sim p_\alpha(z|x, y_g)}[-\log p_\varphi(y_g|z, x)] + D_{KL}[q_\alpha(z|x, y_g)||p_\beta(z|x)] \end{aligned} \quad (4)$$

The above loss function consists of two parts. The first part L_{REC} is the reconstruction loss. Optimizing L_{REC} can push the predictions made by the posterior network and the likelihood network closer to the ground truth labels. The second part L_{KL} is the KL divergence loss. Optimizing it can push the output distribution of the prior network and that of the posterior network closer to each other, such that during testing, when the ground truth label y_g is no longer available, the prior network can still output a reasonable probability distribution over z .

Figure 2 illustrates the structure of the CVAE model during training (the structure used for testing is different). The likelihood network $p_\varphi(y_g|z, x)$ is a Multilayer Perceptron (MLP). The structure of both the posterior network $q_\alpha(z|x, y_g)$ and the prior network $p_\beta(z|x)$ is a bidirectional Long Short-Term Memory (Bi-LSTM) (Hochreiter and Schmidhuber, 1997) layer followed by an MLP. The Bi-LSTM is used to encode the tweet text. The only difference between the posterior and the prior network is that for the posterior network the input for the MLP is the encoded text concatenated with the group label while for the prior network only the encoded text is fed forward. During testing, the posterior network will be replaced by the prior network to generate the distribution over the latent variable (i.e. p' will replace

p). Thus during testing, the ground truth labels will not be used to make predictions.

We assume the latent variable z has a multivariate Gaussian distribution: $p = \mathcal{N}(\mu, \Sigma)$ for the posterior network, and $p' = \mathcal{N}(\mu', \Sigma')$ for the prior network. The detailed computation process is as follows:

$$e = f(x) \quad (5)$$

$$\mu, \Sigma = s(y_g \oplus e) \quad (6)$$

$$\mu', \Sigma' = s'(f'(x)) \quad (7)$$

where f is the Bi-LSTM function and e is the output of the Bi-LSTM layer at the last time step. s is the function of the MLP in the posterior network and s' is that of the MLP in the prior network. The notation \oplus means concatenation. The latent variables z and z' are randomly sampled from $\mathcal{N}(\mu, \Sigma)$ and $\mathcal{N}(\mu', \Sigma')$, respectively. During training, the input for the likelihood network is z :

$$\hat{y}_g = w(z) \quad (8)$$

where w is the function of the MLP in the likelihood network. During testing, the prior network will substitute for the posterior network. Thus for testing, the input for the likelihood is z' instead of z :

$$\hat{y}_g = w(z') \quad (9)$$

4 Our Approach

One problem with the above CVAE method is that it utilizes the group label for training, but ignores the available hate group category (ideology) information of the hate speech. As mentioned in Section 1, hate groups can be divided into different categories in terms of ideologies. Each hate group belongs to a specific hate group category. Considering this hierarchical structure, the hate category information can potentially help the model to better capture the subtle differences between the hate speech from different hate groups. Therefore, we extend the baseline CVAE model to incorporate the category information. In this case, the objective function is as follows:

$$\begin{aligned} Obj &= \sum_{(x, y_c, y_g) \in X} \log p(y_c, y_g | x) \\ &= \sum_{(x, y_c, y_g) \in X} \log p(y_c | x) + \log p(y_g | x, y_c) \end{aligned} \quad (10)$$

where y_c is the hate group category label and

$$p(y_c | x) = \int_{z_c} p(y_c | z_c, x) p(z_c | x) dz_c \quad (11)$$

$$p(y_g | x, y_c) = \int_{z_g} p(y_g | z_g, x, y_c) p(z_g | x, y_c) dz_g \quad (12)$$

where z_c and z_g are latent variables. Therefore, the ELBO of our method is the sum of the ELBOs of $\log p(y_c | x)$ and $\log p(y_g | x, y_c)$:

$$\begin{aligned} ELBO &= E[\log p(y_c | z_c, x)] - \\ &\quad D_{KL}[q(z_c | x, y_c) || p(z_c | x)] + \\ &\quad E[\log p(y_g | z_g, x, y_c)] - \\ &\quad D_{KL}[q(z_g | x, y_c, y_g) || p(z_g | x, y_c)] \end{aligned} \quad (13)$$

During testing, the prior networks will substitute the posterior networks and the ground truth labels y_c and y_g are not utilized. Hence the prior $p(z_g | x, y_c)$ in the above equation cannot be parametrized as a network that directly takes the ground truth label y_c and x as inputs. Instead, we parameterize it as shown in the right part of Figure 3. We assume that u' is trained to be a latent representation of y_c , so we use u' and x as inputs for this prior network.

According to the ELBO above, the corresponding loss function \mathcal{L} is the combination of the loss function for the category classification (\mathcal{L}_c) and that for the group classification (\mathcal{L}_g).

$\mathcal{L} = \mathcal{L}_c + \mathcal{L}_g$, where

$$\begin{aligned} \mathcal{L}_c &= E_{z_c \sim q_\alpha(z_c | x, y_c)} [-\log p_\varphi(y_c | z_c, x)] + \\ &\quad D_{KL}[q_\alpha(z_c | x, y_c) || p_\beta(z_c | x)], \text{ and} \\ \mathcal{L}_g &= E_{z_g \sim q_\eta(z_g | x, y_c, y_g)} [-\log p_\theta(y_g | z_g, x, y_c)] + \\ &\quad D_{KL}[q_\eta(z_g | x, y_c, y_g) || p_\gamma(z_g | x, u)] \end{aligned} \quad (14)$$

Figure 3 shows the structure of our model for training. By assuming the latent variables z_c and z_g have multivariate Gaussian distributions, the actual outputs of the posterior and prior networks are the mean and variance: $p_c = \mathcal{N}(\mu_c, \Sigma_c)$, $p_g = \mathcal{N}(\mu_g, \Sigma_g)$ for the two posterior networks, and $p'_c = \mathcal{N}(\mu'_c, \Sigma'_c)$, $p'_g = \mathcal{N}(\mu'_g, \Sigma'_g)$ for the two prior networks. Note that in addition to these four distributions, there is another distribution $p_x = \mathcal{N}(\mu_x, \sigma_x)$ as shown in Figure 3. This distribution is generated only with the input text x , so it provides the basic distribution (in order to avoid confusion, we call it the basic distribution instead

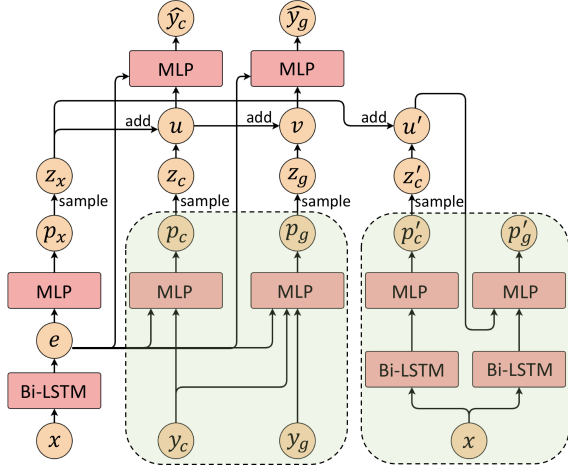


Figure 3: The structure of the HCVAE during training. Bi-LSTM is a bidirectional LSTM layer. MLP is a Multilayer Perceptron. x is the embedded input text. y_c and y_g are the ground truth hate category label and hate group label. \hat{y}_c and \hat{y}_g are the output predictions of the hate category and hate group. z_x , z_g , and z'_c are latent variables. In the left dotted box are two posterior networks. In the right dotted box are two prior networks. Note that this structure is used for training. During testing, the posterior networks will be substituted by the posterior networks (i.e. the left dotted box will be substituted with the right one) to compute \hat{y}_c and \hat{y}_g . Thus y_c and y_g are not available during testing. Refer to Section 4 for detailed explanation.

of the prior distribution) for both two posterior networks and two prior networks. With this basic distribution, the two posterior networks only need to capture the additional signals learned from x and labels. Similarly, the prior networks only need to learn the additional signals learned by the posterior networks. The detailed computation process during training is shown as the following equations:

$$e = f(x) \quad (15)$$

$$\mu_x, \Sigma_x = s_x(e) \quad (16)$$

$$\mu_c, \Sigma_c = s_c(y_c \oplus e) \quad (17)$$

$$\mu_g, \Sigma_g = s_g(y_c \oplus y_g \oplus e) \quad (18)$$

$$\mu'_c, \Sigma'_c = s'_c(f'_c(x)) \quad (19)$$

where f is the Bi-LSTM function and e is the output of the Bi-LSTM layer at the last time step. s_x , s_c , s_g , and s'_c are the functions of four different MLPs. f'_c is the Bi-LSTM function in the prior network $p_\beta(z_c|x)$. The latent variables z_x , z_c , z_g , and z'_c are randomly sampled from the Gaussian distributions $\mathcal{N}(\mu_x, \Sigma_x)$, $\mathcal{N}(\mu_c, \Sigma_c)$, $\mathcal{N}(\mu_g, \Sigma_g)$, and $\mathcal{N}(\mu'_c, \Sigma'_c)$ respectively. As mentioned above,

Algorithm 1 Train & Test Algorithm

```

1: function TRAIN( $X$ )
2:   randomly initialize network parameters;
3:   for  $epoch = 1, E$  do
4:     for ( $text, category, group$ ) in  $X$  do
5:       get embeddings  $x$  and one-hot vectors  $y_c, y_g$ ;
6:       compute  $e$  with the Bi-LSTM;
7:       compute  $\mu_x, \Sigma_x, \mu_c, \Sigma_c, \mu_g, \Sigma_g$ ;
8:       sample  $z_x = \text{reparameterize}(\mu_x, \Sigma_x)$ ;
9:       sample  $z_c = \text{reparameterize}(\mu_c, \Sigma_c)$ ;
10:      sample  $z_g = \text{reparameterize}(\mu_g, \Sigma_g)$ ;
11:       $u = z_x + z_c$ ;
12:       $v = u + z_g$ ;
13:      compute  $\hat{y}_c$  and  $\hat{y}_g$  according to Eq. 24- 25;
14:       $\mathcal{L}_{REC} = BCE(\hat{y}_c, y_c) + BCE(\hat{y}_g, y_g)$ ;
15:      compute  $\mu'_c, \Sigma'_c$ ;
16:      sample  $z'_c = \text{reparameterize}(\mu'_c, \Sigma'_c)$ ;
17:       $u' = z_x + z'_c$ ;
18:      compute  $\mu'_g, \Sigma'_g$ ;
19:       $\mathcal{L}_{KL} = D_{KL}(p_c||p'_c) + D_{KL}(p_g||p'_g)$ ;
20:       $\mathcal{L} = \mathcal{L}_{KL} + \mathcal{L}_{REC}$ ;
21:      update network parameters on  $\mathcal{L}$ ;
22:     end for
23:   end for
24: end function
25:
26: function TEST( $X$ )
27:   for  $text$  in  $X$  do
28:     get embeddings  $x$ ;
29:     compute  $e$  with the Bi-LSTM;
30:     compute  $\mu_x, \Sigma_x, \mu'_c, \Sigma'_c$ ;
31:     sample  $z_x = \text{reparameterize}(\mu_x, \Sigma_x)$ ;
32:     sample  $z'_c = \text{reparameterize}(\mu'_c, \Sigma'_c)$ ;
33:      $u' = z_x + z'_c$ ;
34:     compute  $\mu'_g, \Sigma'_g$ ;
35:     sample  $z'_g = \text{reparameterize}(\mu'_g, \Sigma'_g)$ ;
36:      $v' = u' + z'_g$ ;
37:     compute  $\hat{y}_c$  and  $\hat{y}_g$  according to Eq. 26- 28;
38:   end for
39: end function

```

p_x is the basic distribution while p_c , p_g , p'_c , and p'_g are trained to capture the additional signals. Therefore, z_x is added to z_c and z'_c , then the results u and u' are further added to z_g and z'_g , respectively, as shown in the following equations:

$$u' = z_x + z'_c \quad (20)$$

$$\mu'_g, \Sigma'_g = s'_g(u' \oplus f'_g(x)) \quad (21)$$

$$u = z_x + z_c \quad (22)$$

$$v = u + z_g \quad (23)$$

where f'_g is the Bi-LSTM function and s'_g is the function of the MLP in the prior network $p_\gamma(z_g|x, u)$. \oplus is element-wise addition. During training, u and v are fed into the likelihood networks:

$$\hat{y}_c = w_c(e \oplus u) \quad (24)$$

$$\hat{y}_g = w_g(e \oplus v) \quad (25)$$

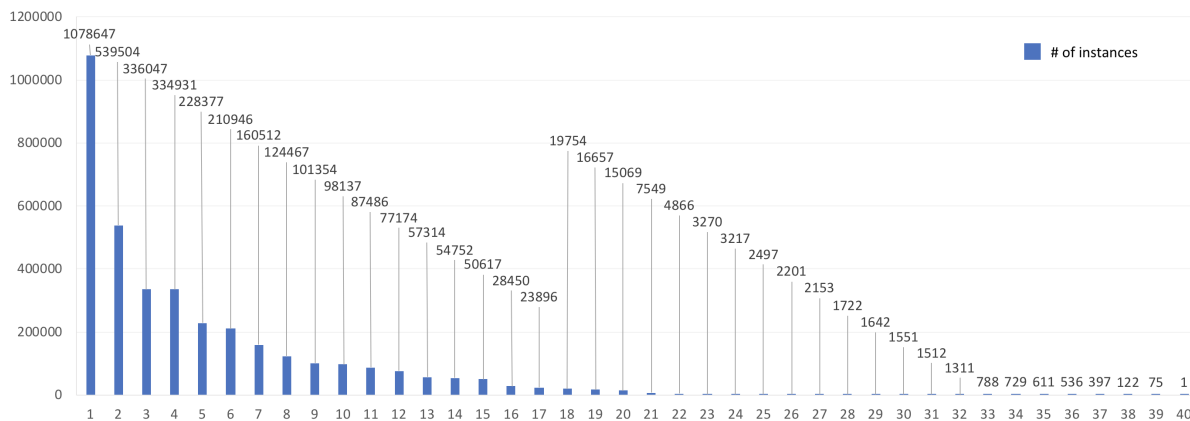


Figure 4: The distribution of the data for 40 hate groups. The X-Axis is each hate group. The Y-Axis is the number of tweets collected for each hate group.

where w_c and w_g are the functions of the MLPs in two likelihood networks. During testing, the prior networks will substitute the posterior networks, so the latent variable z'_g is randomly sampled from the Gaussian distributions $\mathcal{N}(\mu'_g, \Sigma'_g)$, and the last four equations above (Equation 22 - 25) will be replaced by the following ones:

$$\hat{y}_c = w_c(e \oplus u') \quad (26)$$

$$v' = u' + z'_g \quad (27)$$

$$\hat{y}_g = w_g(e \oplus v') \quad (28)$$

Algorithm 1 illustrates the complete training and testing process. *BCE* refers to the Binary Cross Entropy loss. Note that during training, the ground truth category labels and group labels are fed to the posterior network to generate latent variables. But during testing, the latent variables are generated by the prior network, which only utilizes the texts as inputs.

5 Experiments

5.1 Dataset

We collect the data from 40 hate group Twitter accounts of 13 different hate ideologies, e.g., white nationalist, anti-immigrant, racist skinhead, among others. The detailed themes and core values behind each hate ideology are discussed in the SPLC ideology section.³ For each hate ideology, we collect a set of Twitter handles based on hate groups identified by the SPLC center.⁴ For

³<https://www.splcenter.org/fighting-hate/extremist-files/ideology>

⁴<https://www.splcenter.org/fighting-hate/extremist-files/groups>

each hate ideology, we select the top three handles in terms of the number of followers. Due to ties, there are four different groups in several categories of our dataset. The dataset consists of all the content (tweets, retweets, and replies) posted with each account from the group's inception date, as early as 07-2009, until 10-2017. Each instance in the dataset is a tuple of (tweet text, hate category label, hate group label). The complete dataset consists of approximately 3.5 million tweets. Note that due to the sensitive nature of the data, we anonymously reference the Twitter handles for each hate group by using IDs throughout this paper. The distribution of the data is illustrated in Figure 4.

5.2 Experimental Settings

In addition to the discriminative CVAE model described in Section 3, we implement for other baseline methods and an upper bound model as follows.

Support Vector Machine (SVM): We implement an SVM model with linear kernels. We use L2 regularization and the coefficient is 1. The input features are the Term Frequency Inverse Document Frequency (TF-IDF) values of up to 2-grams.

Logistic Regression (LR): We implement the Logistic Regression model with L2 regularization. The penalty parameter C is set to 1. Similar to the SVM model, we use TF-IDF values of up to 2-grams as features.

Character-level Convolutional Neural Network (Char-CNN): We implement the character-level CNN model for text classification as described in (Zhang et al., 2015). It is 9 layers deep with 6 convolutional layers and 3 fully-connected layers.

Dataset	Complete			Subset		
	Macro-F1	Micro-F1	Weighted-F1	Macro-F1	Micro-F1	Weighted-F1
upper bound	.697	.904	.881	–	–	–
SVM + tf-idf	.653	.834	.835	.521	.771	.772
LR + tf-idf	.586	.787	.792	.494	.727	.736
Char-CNN	.604	.840	.853	.457	.730	.744
Bi-LSTM	.627	.767	.745	.353	.599	.570
CVAE	.520	.799	.830	.453	.774	.784
HCVAE (our)	.664	.844	.858	.469	.787	.799

Table 1: Experimental results. Complete: The performance achieved when 90% of the entire dataset is used for training. Subset: The performance achieved when only 10% of the dataset is used for training. The best results are in bold.

The configurations of the convolutional layers are kept the same as those in (Zhang et al., 2015).

Bi-LSTM: The model consists of a bidirectional LSTM layer followed by a linear layer. The embedded tweet text x is fed into the LSTM layer and the output at the last time step is then fed into the linear layer to predict \hat{y}_g .

Upper Bound: The upper bound model also consists of a bidirectional LSTM layer followed by a linear layer. The difference between this model and Bi-LSTM is that it takes the tweet text x along with the ground truth category label y_c as input during both training and testing. The LSTM layer is used to encode x . The encoding result is concatenated with the ground truth category label and then fed into the linear layer to give the prediction of the hate group \hat{y}_g . Since it utilizes y_c for prediction during testing, this model sets an upper bound performance for our method.

For the baseline CVAE, Bi-LSTM, the upper bound model, and the HCVAE, we use randomly initialized word embeddings with size 200. All the neural network models are optimized by the Adam optimizer with learning rate $1e-4$. The batch size is 20. The hidden size of all the LSTM layers in these models is 64 and all the MLPs are three-layered with non-linear activation function Tanh. For the CVAE and the HCVAE, the size of the latent variables is set to 256.

All the baseline models and our model are evaluated on two datasets. We first use the complete dataset for training and testing. 90% of the instances are used for training and the rest for testing. In order to evaluate the robustness of the baseline models and our model, we also randomly selected a subset (10%) of the original dataset for training while the testing dataset is fixed. Since the upper bound model is used to set an upper bound on performance, we do not evaluate it on

the smaller training dataset. We use Macro-F1, Micro-F1, and Weighted-F1 to evaluate the classification results. As shown in Figure 4, the dataset is highly imbalanced, which causes problems for training the above models. In order to alleviate this problem, we use a weighted random sampler to sample instances from the training data. However, the testing dataset is not sampled, so the distribution of the testing dataset remains the same as that of the original dataset. This allows us to evaluate the models’ performance on the data with a realistic distribution.

5.3 Experimental Results

The experimental results are shown in Table 1. The testing dataset keeps the imbalanced distribution of the original data, so the Macro-F1 score of each method is significantly lower than the Micro-F1 score and the Weighted-F1 score. Comparing the performance of the Bi-LSTM model and that of the baseline CVAE model shows that although CVAE is traditionally used as a generative model, it can also achieve competitive performance as a discriminative model. All the methods achieve lower F1 scores when using the smaller training dataset. Due to the imbalanced distribution of our dataset, half of the 40 groups has less than 1k tweets in the smaller training dataset, which leads to the sharp decline in the Macro-F1 scores of all the methods. However, the performance of both CVAE-based models degrades less than that of the other two neural network models (the Bi-LSTM model and the Char-CNN model) according to the Micro-F1 Weighted-F1 scores. These two CVAE-based models tend to be more robust when the size of the training dataset is smaller. The difference between the CVAE-based models and the other two models is that both the Bi-LSTM model and the Char-CNN model directly compress the input text into a fixed-length latent

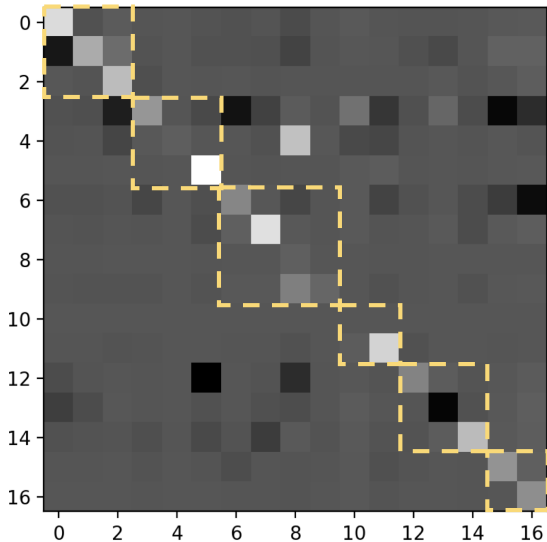


Figure 5: This figure compares a subset (17 hate groups of 6 categories) of the predictions made by the baseline CVAE and the HCVAE. The X and Y axes are the index of each hate group. The hate groups of the same category are grouped together as shown in the dashed squares. The color of the grid (i, j) is mapped from $r_{hcvae}(i, j) - r_{cvae}(i, j)$, where $r(i, j)$ is the fraction of the group i 's instances among the instances predicted as the group j . A higher difference value corresponds to a lighter color. A grid darker than the background is mapped from a negative value while a grid lighter than the background is mapped from a positive one.

variable while the CVAE model explicitly models the posterior and likelihood distributions over the latent variable. The result is that, during testing, the inference strategy of the Bi-LSTM model and the Char-CNN model is actually comparing the compressed text to the compressed instances in the training dataset while the strategy of the CVAE-based models is to compare the prior distributions over the latent variable. Compared with the compressed text, prior distributions may capture higher level semantic information and thus enable better generalization.

By further utilizing the hate category label for training, the HCVAE outperforms the baseline CVAE on all three metrics. Figure 5 illustrates the difference between the predictions made by the HCVAE and the CVAE. As shown in the figure, most of the lighter grids are in the dashed squares while most of the darker grids are outside. This indicates that the HCVAE model tends to correct the prediction of the CVAE's misclassified instances to be closer to the ground truth. In most cases, the misclassified instances can be directly corrected to the ground truth (the diago-

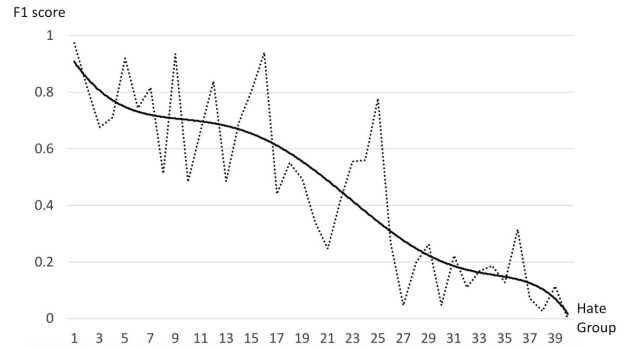


Figure 6: The F1 score achieved by our method on each hate group. The Y-axis is the F1 score. The X-axis is the hate group sorted by the number of instances in the dataset from larger to smaller. The dashed line shows the F1 scores and the solid line is the corresponding trendline.

nal). In other cases, they are not corrected to the ground truth but are corrected to the hate groups of the same categories as the ground truth (the dashed squares). This shows that additional ideology information is useful for the model to better capture the subtle differences among tweets posted by different hate groups.

Although our method outperforms the baseline methods, there is still a gap between its performance and the upper bound performance. We analyze this in the following section.

5.4 Error Analysis

As mentioned above, in some cases our model misclassified the tweet as a group under the same category as the ground truth. Take, for instance, the tweet from *Group1*: *The only good #muslim is a #dead #muslim*. Our HCVAE model predicts it as from *Group2*. But both *Group1* and *Group2* are under the category “neo nazi”. One possible reason for this kind of error is that the imbalance of the dataset adversely affects the performance of our method. Figure 6 shows the F1 scores achieved by the HCVAE on each group. The performance of the model tends to be much lower when the number of the group's training instances decreases. Although we use the weighted random sampler to alleviate the problem of the imbalanced dataset, repeating the very limited data instances (less than 3k) of a group during training cannot really help the posterior and prior networks to give a reasonable distribution that can generalize well on unseen inputs. Therefore, when the model comes

into the instances in the testing dataset, the performance can be less satisfactory. This is a common problem for all the methods, which is the cause of the significantly lower Macro-F1 scores.

Another type of error is caused by the noisy dataset. Take, for instance, the tweet from *Group3* under the category “ku klux klan”: *we must secure the existence of our people and future for the White Children !!* Our model classifies it as from *Group4* under the category “neo nazi”, which makes sense but is an error.

6 Conclusion

In this paper, we explore the CVAE as a discriminative model and find that it can achieve competitive results. In addition, the performance of the CVAE-based models tend to be more stable than that of the others when the dataset gets smaller. We further propose an extension of the basic discriminative CVAE model to incorporate the hate group category (ideology) information for training. Our proposed model has a hierarchical structure that mirrors the hierarchical structure of the hate groups and the ideologies. We apply the HCVAE to the task of fine-grained hate speech classification, but this Hierarchical CVAE framework can be easily applied to other tasks where the hierarchical structure of the data can be utilized.

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*, pages 759–760. International World Wide Web Conferences Steering Committee.
- Pete Burnap and Matthew L Williams. 2016. Us and Them: Identifying Cyber Hate on Twitter across Multiple Protected Characteristics. *EPJ Data Science*, 5(1):11.
- Thomas Davidson, Dana Warmusley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*.
- Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM.
- Lei Gao, Alexis Kuppersmith, and Ruihong Huang. 2017. Recognizing explicit and implicit hate speech using a weakly supervised two-path bootstrapping approach. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 774–782.
- Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.
- Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112.
- Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.
- Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, volume 333, pages 2267–2273.
- Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. 2016. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1558–1566. JMLR. org.
- Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. 2016. Abusive language detection in online user content. In *Proceedings of the 25th international conference on world wide web*, pages 145–153. International World Wide Web Conferences Steering Committee.
- Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.
- Tim Salimans, Diederik Kingma, and Max Welling. 2015. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

- Kihyuk Sohn, Honglak Lee, and Xinchun Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*.
- Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1556–1566.
- Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432.
- Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Véronique Hoste. 2015. Detection and Fine-grained Classification of Cyberbullying Events. In *RANLP'15: International Conference Recent Advances in Natural Language Processing*, pages 672–680.
- William Warner and Julia Hirschberg. 2012. Detecting Hate Speech on the World Wide Web. In *ACL'12: Proceedings of the 2nd Workshop on Language in Social Media*, pages 19–26. Association for Computational Linguistics.
- Zeeraq Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. In *Proceedings of the NAACL student research workshop*, pages 88–93.
- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.
- Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. *arXiv preprint arXiv:1703.01898*.
- Biao Zhang, Deyi Xiong, Hong Duan, Min Zhang, et al. 2016. Variational neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 521–530.
- Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.
- Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 654–664.
- Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J Miller, and Cornelia Caragea. 2016. Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *IJCAI'16: Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 3952–3958.
- Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.