Learning to Decipher Hate Symbols

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,mayelsherif,ebelding,william}@cs.ucsb.edu

Abstract

Existing computational models to understand hate speech typically frame the problem as a simple classification task, bypassing the understanding of hate symbols (e.g., *14 words*, *kigy*) and their secret connotations. In this paper, we propose a novel task of deciphering hate symbols. To do this, we leverage the Urban Dictionary and collected a new, symbol-rich Twitter corpus of hate speech. We investigate neural network latent context models for deciphering hate symbols. More specifically, we study Sequence-to-Sequence models and show how they are able to crack the ciphers based on context. Furthermore, we propose a novel Variational Decipher and show how it can generalize better to unseen hate symbols in a more challenging testing setting.

1 Introduction

The statistics are sobering. The Federal Bureau of Investigation of United States[1] reported over 6,000 criminal incidents motivated by bias against race, ethnicity, ancestry, religion, sexual orientation, disability, gender, and gender identity during 2016. The most recent 2016 report shows an alarming 4.6% increase, compared with 2015 data[2]. In addition to these reported cases, thousands of Internet users, including celebrities, are forced out of social media due to abuse, hate speech, cyberbullying, and online threats. While such social media data is abundantly available, the broad question we are asking is—What can machine learning and natural language processing do to help and prevent online hate speech?

The vast quantity of hate speech on social media can be analyzed to study online abuse. In recent years, there has been a growing trend of developing computational models of hate speech. However, the majority of the prior studies focus solely on modeling hate speech as a binary or multiclass classification task (Djuric et al., 2015; Waseem and Hovy, 2016; Burnap and Williams, 2016; Wulczyn et al., 2017; Pavlopoulos et al., 2017).

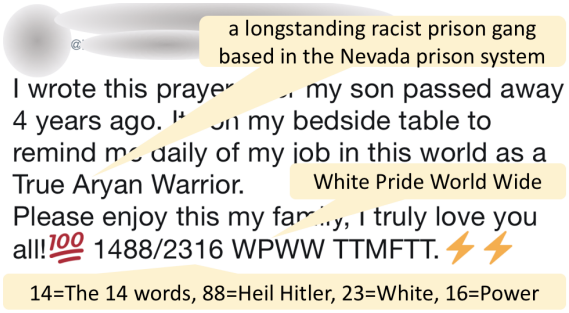

Figure 1: An example tweet with hate symbols.

While developing new features for hate speech detection certainly has merits, we believe that understanding hate speech requires us to design computational models that can decipher hate symbols that are commonly used by hate groups. Figure 1 shows an example usage of hate symbols in an otherwise seemingly harmless tweet that promotes hate. For example, *Aryan Warrior* is a longstanding racist prison gang based in the Nevada prison system. *WPWW* is the acronym for *White Pride World Wide*. The hate symbols *1488* and *2316* are more implicit. *14* symbolizes the 14 words: *"WE MUST SECURE THE EXISTENCE OF OUR PEOPLE AND A FUTURE FOR WHITE CHILDREN"*, spoken by members of the Order neo-Nazi movement. *H* is the 8th letter of the alphabet, so *88=HH=Heil Hitler*. Similarly, *W* is the 23rd and *P* is the 16th letter of the alphabet, so *2316=WP=White Power*.

[1] https://www.fbi.gov/news/stories/2016-hate-crime-statistics
[2] https://www.fbi.gov/news/stories/2015-hate-crime-statistics-released

Proceedings of NAACL-HLT 2019, pages 3006–3015
Minneapolis, Minnesota, June 2 - June 7, 2019.

In this work, we propose the first models for deciphering hate symbols. We investigate two families of neural network approaches: the Sequence-to-Sequence models (Sutskever et al., 2014; Cho et al., 2014) and a novel Variational Decipher based on the Conditional Variational Autoencoders (Kingma and Welling, 2014; Sohn et al., 2015; Larsen et al., 2016). We show how these neural network models are able to guess the meaning of hate symbols based on context embeddings and even generalize to unseen hate symbols during testing. Our contributions are three-fold:

- We propose a novel task of learning to decipher hate symbols, which moves beyond the standard formulation of hate speech classification settings.
- We introduce a new, symbol-rich tweet dataset for developing computational models of hate speech analysis, leveraging the Urban Dictionary.
- We investigate a sequence-to-sequence neural network model and show how it is able to encode context and crack the hate symbols. We also introduce a novel Variational Decipher, which generalizes better in a more challenging setting.

In the next section, we outline related work in text normalization, machine translation, conditional variational autoencoders, and hate speech analysis. In Section 3, we introduce our new dataset for deciphering hate speech. Next, in Section 4, we describe the design of two neural network models for the decipherment problem. Quantitative and qualitative experimental results are presented in Section 5. Finally, we conclude in Section 6.

2 Related Work

2.1 Text Normalization in Social Media

The proposed task is related to text normalization focusing on the problems presented by user-generated content in online sources, such as misspelling, rapidly changing out-of-vocabulary slang, short-forms and acronyms, punctuation errors or omissions, etc. These problems usually appear as out-of-vocabulary words. Extensive research has focused on this task (Beaufort et al., 2010; Liu et al., 2011; Gouws et al., 2011; Han and Baldwin, 2011; Han et al., 2012; Liu et al., 2012; Chrupała, 2014). However, our task is different from the general text normalization in social media in that instead of the out-of-vocabulary words, we focus on the symbols conveying hateful meaning. These hate symbols can go beyond lexical variants of the vocabulary and thus are more challenging to understand.

2.2 Machine Translation

An extensive body of work has been dedicated to machine translation. Knight et al. (2006) study a number of natural language decipherment problems using unsupervised learning. Ravi and Knight (2011) further frame the task of machine translation as decipherment and tackle it without parallel training data. Machine translation using deep learning (Neural Machine Translation) has been proposed in recent years. Sutskever et al. (2014) and Cho et al. (2014) use Sequence to Sequence (Seq2Seq) learning with Recurrent Neural Networks (RNN). Bahdanau et al. (2015) further improve translation performance using the attention mechanism. Google's Neural Machine Translation System (GNMT) employs a deep attentional LSTM network with residual connections (Wu et al., 2016). Recently, machine translation techniques have been also applied to explain non-standard English expressions (Ni and Wang, 2017). However, our deciphering task is not the same as machine translation in that hate symbols are short and cannot be modeled as language.

Our task is more closely related to (Hill et al., 2016) and (Noraset et al., 2017). Hill et al. (2016) propose using neural language embedding models to map the dictionary definitions to the word representations, which is the inverse of our task. Noraset et al. (2017) propose the definition modeling task. However, in their task, for each word to be defined, its pre-trained word embedding is required as an input, which is actually the prior knowledge of the words. However, such kind of prior knowledge is not available in our decipherment task. Therefore, our task is more challenging and is not simply a definition modeling task.

2.3 Conditional Variational Autoencoder

Unlike the original Seq2Seq model that directly encodes the input into a latent space, the Variational Autoencoder (VAE) (Kingma and Welling, 2014) approximates the underlying probability distribution of data. VAE has shown promise in multiple generation tasks, such as handwritten

digits (Kingma and Welling, 2014; Salimans et al., 2015), faces (Kingma and Welling, 2014; Rezende et al., 2014), and machine translation (Zhang et al., 2016). Conditional Variational Autoencoder (Larsen et al., 2016; Sohn et al., 2015) extends the original VAE framework by incorporating conditions during generation. In addition to image generation, CVAE has been successfully applied to some NLP tasks. For example, Zhao et al. (2017) apply CVAE to dialog generation, while Guu et al. (2018) use CVAE for sentence generation.

2.4 Hate Speech Analysis

Closely related to our work are Pavlopoulos et al. (2017); Gao et al. (2017). Pavlopoulos et al. (2017) build an RNN supplemented by an attention mechanism that outperforms the previous state of the art system in user comment moderation (Wulczyn et al., 2017). Gao et al. (2017) propose a weakly-supervised approach that jointly trains a slur learner and a hate speech classifier. While their work contributes to the automation of harmful content detection and the highlighting of suspicious words, our work builds upon these contributions by providing a learning mechanism that deciphers suspicious hate symbols used by communities of hate to bypass automated content moderation systems.

3 Dataset

In this section, we describe the dataset we collected for hate symbol decipherment.

3.1 Hate Symbols

We first collect hate symbols and the corresponding definitions from the Urban Dictionary. Each term with one of the following hashtags: *#hate, #racism, #racist, #sexism, #sexist, #nazi* is selected as a candidate and added to the set S_0. We collected a total of 1,590 terms. Next, we expand this set by different surface forms using the Urban Dictionary API. For each term s_i in set S_0, we obtain a set of terms R_i that have the same meaning as s_i but with different surface forms. For example, for the term *brown shirt*, there are four terms with different surface forms: *brown shirt, brown shirts, Brownshirts, brownshirt*. Each term in R_i has its own definition in Urban Dictionary, but since these terms have exactly the same meaning, we select a definition d_i with maximum upvote/downvote ratio for all the terms in R_i. For example, for each term in the set R_i=$\{$*brown shirt, brown shirts, Brownshirts, brownshirt*$\}$, the corresponding definition is *"Soldiers in Hitler's storm trooper army, SA during the Nazi regime..."* After expanding, we obtain 2,105 distinct hate symbol terms and their corresponding definitions. On average, each symbol consists of 9.9 characters, 1.5 words. Each definition consists of 96.8 characters, 17.0 words.

3.2 Tweet Collection

For each of the hate symbols, we collect all tweets from 2011-01-01 to 2017-12-31 that contain exactly the same surface form of hate symbol in the text. Since we only focus on hate speech, we train an SVM (Cortes and Vapnik, 1995) classifier to filter the collected tweets. The SVM model is trained on the dataset published by Waseem and Hovy (2016). Their original dataset contains three labels: Sexism, Racism, and None. Since the SVM model is used to filter the non-hate speech, we merge the instances labeled as sexism and racism, then train the SVM model to do binary classification. After filtering out all the tweets classified as non-hate, our final dataset consists of 18,667 (tweet, hate symbol, definition) tuples.

4 Our Approach

We formulate hate symbol deciphering as the following equation:

$$Obj = \sum_{(u,s,d^*)\in X} \log p(d^*|(u,s)) \qquad (1)$$

X is the dataset, (u, s, d^*) is the (tweet, symbol, definition) tuple in the dataset. The inputs are the tweet and the hate symbol in this tweet. The output is the definition of the symbol. Our objective is to maximize the probability of the definition given the (tweet, symbol) pair. This objective function is very similar to that of machine translation. So we first try to tackle it based on the Sequence-to-Sequence model, which is commonly used in machine translation.

4.1 Sequence-to-Sequence Model

We implement an RNN Encoder-Decoder model with attention mechanism based on Bahdanau et al. (2015). We use GRU (Cho et al., 2014) for decoding. However, instead of also using GRU for encoding, we found that LSTM (Hochreiter

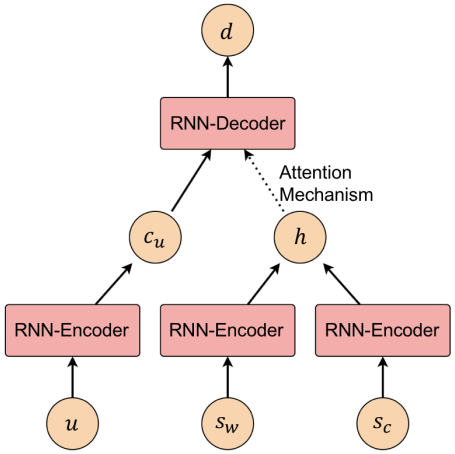

Figure 2: Our Seq2Seq model. u, s_w are the word embeddings of the tweet text and hate symbol. s_c is the character embedding of the symbol. c_u is the encoded tweet and h is the concatenated hidden states. d is the generated text. Detailed explanation is in section 4.1.

and Schmidhuber, 1997) performs better on our task. Therefore, our Seq2Seq model uses LSTM encoders and GRU decoders. An overview of our Seq2Seq model is shown in Figure 2. The computation process is shown as the following equations:

$$c_u, h_u = f_u(u) \tag{2}$$
$$c_{sw}, h_{sw} = f_{sw}(s_w) \tag{3}$$
$$c_{sc}, h_{sc} = f_{sc}(s_c) \tag{4}$$

u is the word embedding of the tweet text, s_w is the word embedding of the hate symbol, s_c is the character embedding of the symbol. f_u, f_{sw}, and f_{sc} are LSTM functions. c_u, c_{sw}, c_{sc} are the outputs of the LSTMs at the last time step and h_u, h_{sw}, h_{sc} are the hidden states of the LSTMs at all time steps. We use two RNN encoders to encode the symbol, one encodes at the word level and the other one encodes at the character level. The character-level encoded hate symbol is used to provide the feature of the surface form of the hate symbol while the word-level encoded hate symbol is used to provide the semantic information of the hate symbol. The hidden states of the two RNN encoders for hate symbols are concatenated:

$$h = h_{sw} \oplus h_{sc} \tag{5}$$

c_u is the vector of encoded tweet text. The tweet text is the context of the hate symbol, which provides additional information during decoding. Therefore, the encoded tweet text it is also fed into the RNN decoder. The detailed attention mechanism and decoding process at time step t are as follows:

$$w_t = \sigma(l_w(d_{t-1} \oplus e_{t-1})) \tag{6}$$
$$a_t = \sum_{i=1}^{T} w_{ti} h_i \tag{7}$$
$$b_t = \sigma(l_c(d_{t-1} \oplus a_t)) \tag{8}$$
$$o_t, e_t = k(c_u \oplus b_t, e_{t-1}) \tag{9}$$
$$p(d_t|u, s) = \sigma(l_o(o_t)) \tag{10}$$

w_t is the attention weights at time step t and w_{ti} is the i_{th} weight of w_t. d_{t-1} is the generated word at last time step and e_{t-1} is the hidden state of the decoder at last time step. h_i is the i_{th} time step segment of h. l_w, l_c, and l_o are linear functions. σ is a nonlinear activation function. k is the GRU function. o_t is the output and e_t is the hidden state of the GRU. $p(d_t|u, s)$ is the probability distribution of the vocabulary at time step t. The attention weights w_t are computed based on the decoder's hidden state and the generated word at time step $t-1$. Then the computed weights are applied to the concatenated hidden states h of encoders. The result a_t is the context vector for the decoder at time step t. The context vector and the last generated word are combined by a linear function l_c followed by a nonlinear activation function. The result b_t is concatenated with the encoded tweet context c_u, and then fed into GRU together with the decoder's last hidden state e_{t-1}. Finally, the probability of each vocabulary word is computed from o_t.

4.2 Variational Decipher

The Variational Decipher is based on the CVAE model, which is another model that can be used to parametrize the conditional probability $p(d^*|(u, s))$ in the objective function (Equation 1). Unlike the Seq2Seq model, which directly parametrizes $p(d^*|(u, s))$, our variational decipher formulates the task as follows:

$$\begin{aligned} Obj &= \sum_{(u,s,d^*)\in X} \log p(d^*|(u, s)) \\ &= \sum_{(u,s,d^*)\in X} \log \int_z p(d^*|z)p(z|(u, s))dz \end{aligned} \tag{11}$$

where z is the latent variable. $p(d^*|(u, s)$ is written as the marginalization of the product of two terms over the latent space. Since the integration

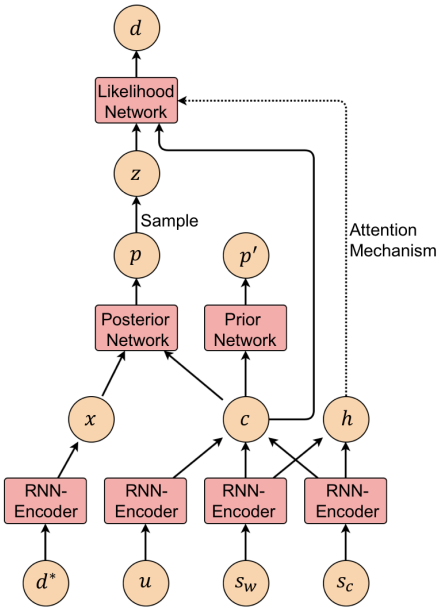

Figure 3: The Variational Decipher. Note that this structure is used during training. During testing, the structure is slightly different. d^* is the word embeddings of the definition. x is the encoded definition. c is the concatenation of the encoded tweet and hate symbol. p and p' are output distributions. z is the latent variable. The definitions of other variables are the same as those in Figure 2. Detailed explanation is in section 4.2.

over z is intractable, we instead try to maximize the evidence lower bound (ELBO). Our variational lower bound objective is in the following form:

$$\begin{aligned} Obj =& E[\log p_\varphi(d^*|z,u,s)] - \\ & D_{KL}[p_\alpha(z|d^*,u,s)||p_\beta(z|u,s)] \end{aligned} \tag{12}$$

where $p_\varphi(d^*|z,u,s)$ is the likelihood, $p_\alpha(z|d^*,u,s)$ is the posterior, $p_\beta(z|u,s)$ is the prior, and D_{KL} is the Kullback-Leibler (KL) divergence. We use three neural networks to model these three probability distributions. An overview of our variational decipher is shown in Figure 3. We first use four recurrent neural networks to encode the (tweet, symbol, definition) pair in the dataset. Similar to what we do in the Seq2Seq model, there are two encoders for the hate symbol. One is at the word level and the other is at the character level. The encoding of symbols and tweets are exactly the same as in our Seq2Seq model (see Equations 2-4). The difference is that we also need to encode definitions for the Variational Decipher.

$$x, h_d = f_d(d^*) \tag{13}$$

Here, f_d is the LSTM function. x is the output of the LSTM at the last time step and h_d is the hidden state of the LSTM at all time steps. The condition vector c is the concatenation of the encoded symbol words, symbol characters, and the tweet text:

$$c = c_u \oplus c_{sw} \oplus c_{sc} \tag{14}$$

We use multi-layer perceptron (MLP) to model the posterior and the prior in the objective function. The posterior network and the prior network have the same structure and both output a probability distribution of latent variable z. The only difference is that the input of the posterior network is the concatenation of the encoded definition x and the condition vector c while the input of the prior network is only the condition vector c. Therefore, the output of the posterior network $p = p_\alpha(z|d^*,u,s)$ and the output of the prior network $p' = p_\beta(z|u,s)$. By assuming the latent variable z has a multivariate Gaussian distribution, the actual outputs of the posterior and prior networks are the mean and variance: (μ, Σ) for the posterior network and (μ', Σ') for the prior network.

$$\mu, \Sigma = g(x \oplus c) \tag{15}$$

$$\mu', \Sigma' = g'(c) \tag{16}$$

g is the MLP function of the posterior network and g' is that of the prior network. During training, the latent variable z is randomly sampled from the Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ and fed into the likelihood network. During testing, the posterior network is replaced by the prior network, so z is sampled from $\mathcal{N}(\mu', \Sigma')$. The likelihood network is modeled by an RNN decoder with attention mechanism, very similar to the decoder of our Seq2Seq model. The only difference lies in the input for the GRU. The decoder in our Variational Decipher model is to model the likelihood $p_\varphi(d^*|z,u,s)$, which is conditioned on the latent variable, tweet context, and the symbol. Therefore, for the Variational Decipher, the condition vector c and the sampled latent variable z are fed into the decoder.

$$o_t, e_t = k(z \oplus c \oplus b_t, e_{t-1}) \tag{17}$$

e_{t-1} is the hidden state of the RNN decoder at the last time step. k is the GRU function. o_t is its output and e_t is its hidden state. Detailed decoding process and explanations are in section 4.1.

According to the objective function in Equation 12, the loss function of the Variational Decipher is as follows:

$$\begin{aligned}\mathcal{L} =& L_{REC} + L_{KL} \\ =& E_{z\sim p_\alpha(z|d^*,u,s)}[-\log p_\varphi(d^*|z,u,s)] + \\ & D_{KL}[p_\alpha(z|d^*,u,s)||p_\beta(z|u,s)]\end{aligned} \quad (18)$$

It consists of two parts. The first part L_{REC} is called reconstruction loss. Optimizing L_{REC} can push the sentences generated by the posterior network and the likelihood network closer to the given definitions. The second part L_{KL} is the KL divergence loss. Optimizing this loss can push the output Gaussian Distributions of the prior network closer to that of the posterior network. This means we teach the prior network to learn the same knowledge learned by the posterior network, such that during testing time, when the referential definition d^* is no longer available for generating the latent variable z, the prior network can still output a reasonable probability distribution over the latent variable z. The complete training and testing process for the Variational Decipher is shown in Algorithm 1. M is the predefined maximum length of the generated text. BCE refers to the Binary Cross Entropy loss.

Algorithm 1 Train & Test Variational Decipher

function TRAIN(U)
  randomly initialize network parameters φ, α, β;
  for $epoch = 1, E$ **do**
    for $(tweet, symbol, definition)$ in U **do**
      get embeddings u, s_w, s_c, d^*;
      compute x, c and h with RNN encoders;
      compute μ, Σ with the posterior network;
      compute μ', Σ' with the prior network;
      compute KL-divergence loss L_{KL};
      sample $z = reparameterize(\mu, \Sigma)$;
      initialize the decoder state $e_0 = c$;
      $L_{REC} = 0$;
      for $t = 1, M$ **do**
        compute attention weights w_t;
        compute o_t, e_t and $p(d_t|z,u,s)$;
        $d_t = indmax(p(d_t|z,u,s))$;
        $L_{REC} += BCE(d_t, d_t^*)$;
        if d_t==EOS **then**
          break;
        end if
      end for
      update φ, α, β on $\mathcal{L} = L_{REC} + L_{KL}$;
    end for
  end for
end function

function TEST(V)
  for $(tweet, symbol, definition)$ in V **do**
    get embeddings u, s_w, s_c;
    compute c and h with RNN encoders;
    compute μ', Σ' with the prior network;
    sample $z = reparameterize(\mu', \Sigma')$;
    initialize the decoder state $e_0 = c$;
    for $t = 1, M$ **do**
      compute attention weights w;
      compute o_t, e_t and $p(d_t|z,u,s)$;
      $d_t = indmax(p(d_t|z,u,s))$;
      if d_t==EOS **then**
        break;
      end if
    end for
  end for
end function

5 Experiments

5.1 Experimental Settings

We use the dataset collected as described in section 3 for training and testing. We randomly selected 2,440 tuples for testing and use the remaining 16,227 tuples for training. Note that there are no overlapping hate symbols between the training dataset U and the testing dataset D.

We split the 2,440 tuples of the testing dataset D into two separate parts, D_s and D_d. D_s consists of 1,681 examples and D_d consists of 759 examples. In the first testing dataset D_s, although each hate symbol does not appear in the training dataset, the corresponding definition appears in the training dataset. In the second testing dataset D_d, neither the hate symbols nor the corresponding definitions appear in the training dataset. We do this split because deciphering hate symbols in these two cases has different levels of difficulty.

This split criterion means that for each hate symbol in D_s, there exists some symbol in the training dataset that has the same meaning but in different surface forms. For example, the hate symbol *wigwog* and *Wig Wog* have the same definition but one is in the training dataset, the other is in the first testing dataset. We assume that such types of hate symbols share similar surface forms or similar tweet contexts. Therefore, the first testing dataset D_s is to evaluate how well the model captures the semantic similarities among the tweet contexts in different examples or the similarities among different surface forms of a hate symbol.

Deciphering the hate symbols in the second testing dataset D_d is more challenging. Both the unseen hate symbols and definitions require the model to have the ability to accurately capture the semantic information in the tweet context and then make a reasonable prediction. The second testing dataset D_d is used to evaluate how well the model generalizes to completely new hate symbols.

Dataset	Method	BLEU	ROUGE-L	METEOR
D_s	Seq2Seq	**37.80**	**41.05**	**36.67**
	VD	34.77	32.96	31.03
D_d	Seq2Seq	25.44	12.96	**5.54**
	VD	**28.38**	**14.01**	5.41
D	Seq2Seq	**33.96**	**32.32**	**26.98**
	VD	32.75	27.00	23.16

Table 1: The BLEU, ROUGE-L and METEOR scores on testing datasets. VD refers to the Variational Decipher. D is the entire testing dataset. D_s is the first part of D and D_d is the second part. The better results are in bold.

For the Seq2Seq model, we use negative log-likelihood loss for training. Both models are optimized using Adam optimizer (Kingma and Ba, 2015). The hyper-parameters of two models are exactly the same. We set the maximum generation length $M = 50$. The hidden size of the encoders is 64. The size of the word embedding is 200 and that of character embedding is 100. The word embeddings and character embeddings are randomly initialized. Each model is trained for 50 epochs. We report the deciphering results of two models on three testing datasets D, D_s and D_d.

5.2 Experimental Results

Quantitative Results: We use equally weighted BLEU score for up to 4-grams (Papineni et al., 2002), ROUGE-L (Lin, 2004) and METEOR (Banerjee and Lavie, 2005) to evaluate the decipherment results. The results are shown in Table 1. Figure 4 shows the BLEU score achieved by the two models on three testing datasets D, D_s and D_d during the training process. Both our Seq2Seq model and Variational Decipher achieve reasonable BLEU scores on the testing datasets. The Seq2Seq model outperforms the Variational Decipher on D_s while Variational Decipher outperforms Seq2Seq on D_d. Note that D_s is more than twice the size of D_d. Therefore, Seq2Seq outperforms Variational Decipher on the entire testing dataset D. The different performance of the two models on D_s and D_d is more obvious in Figure 4. The gap between the performance of the Seq2Seq model on D_s and D_d is much larger than that between the performance of the Variational Decipher on these two datasets.

Human Evaluation: We employed crowdsourced workers to evaluate the deciphering results of two models. We randomly sampled 100 items of deciphering results from D_s and another

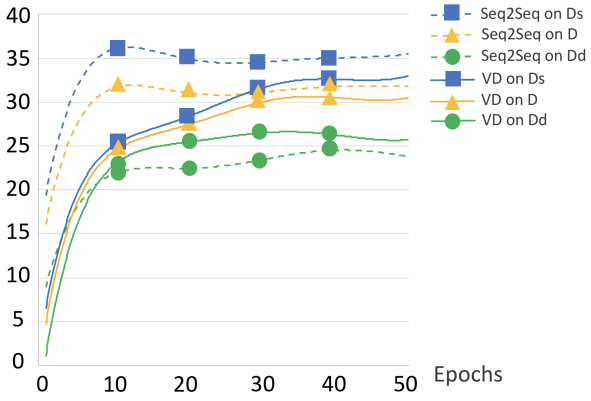

Figure 4: BLEU scores of two models on the testing dataset D, D_s and D_d. The three dotted curves represent the performance of the Seq2Seq model while the three solid curves represent the performance of the Variational Decipher.

Dataset	Seq2Seq Lose	Seq2Seq Win	Tie
D_s	31.0%	32.0%	37.0%
D_d	30.5%	22.0%	47.5%

Table 2: The results of human evaluation on two separate testing datasets D_s and D_d.

100 items from D_d. Each item composes a choice question and each choice question is assigned to five workers on Amazon Mechanical Turk. In each choice question, the workers are given the hate symbol, the referential definition, the original tweet and two machine-generated plain texts from the Seq2Seq model and Variational Decipher. Workers are asked to select the more reasonable of the two results. In each choice question, the order of the results from the two models is permuted. Ties are permitted for answers. We batch five items in one assignment and insert an artificial item with two identical outputs as a sanity check. The workers who fail to choose "tie" for that item are rejected from our test. The human evaluation results are shown in Table 2, which coincide with the results in Table 1 and Figure 4.

Discussion: When deciphering the hate symbols that have the same definitions as in the training dataset, the model can rely more on the surface forms of hate symbols than the tweet context to make a prediction because usually the hate symbols that share the same definitions also have similar surface forms. However, when it comes to the hate symbols with unseen definitions, simply relying on the surface forms cannot lead to a reasonable deciphering result. Instead, the model should learn the relationships between the con-

Dataset	Symbol	tweet	Referential definition	Result of Seq2Seq	Result of VD
D_s	*macaca*	What an ugly thing! And it's not because she's black, purple etc, it's because she's soooooooo ugly piece of *macaca*	Common French racist slur	Common French a racist	Common term black slur
D_s	*closet homosexuals*	I wish the stupid Imams would just admit already that Muslim men are sex addicts & *closet homosexuals.*	A Homosexual who hasn't told anyone about his/her sexuality	A Homosexual who hasn't told anyone about and her sexuality	A Homosexual who hasn't told told anyone about his her sexuality
D_s	*confederate flag*	done w/ this stupid sh*t. I support the *confederate flag* & will wear it. Will post pictures of it. Will fly it...	A flag that's usually flown in the south, most of the time flown to represent southern pride and heritage, but sometimes is flown to represent white power and racism	When s is or a any of but but	The from is and is or is or their south their to is or to
D_d	*niggering*	More like call of n*gger: advanced *niggering*	The act of being a n*gger	A thing is to black to to to to	A thing but a n*gger n*gger n*gger
D_d	*Heil Hitler*	*Heil Hitler*!!! RT @Am_Yuggio: 41.) Who is your role model ? :)	Its a gesture used by raising up the right hand with a straighten hand	A Schizophrenic of the the and Nazi Nazi by racist of Nazi more	Leader person and and Nazi and shit and

Figure 5: Some example errors in the generated results of our Seq2Seq model and Variational Decipher.

text information and the definition of the symbol. Therefore, the different performances of two models on the two testing datasets D_s and D_d indicate that the Seq2Seq model is better at capturing the similarities among different surface forms of a hate symbol, while the Variational Decipher is better at capturing the semantic relationship between the tweet context and the hate symbol. The Sequence-to-Sequence model tries to capture such kinds of relationships by compressing all the context information into a fixed length vector, so its deciphering strategy is actually behavior cloning. On the other hand, the Variational Decipher captures such relationships by explicitly modeling the posterior and likelihood distributions. The modeled distributions provide higher-level semantic information compared to the compressed context, which allows the Variational Decipher to generalize better to the symbols with unseen definitions. This explains why the gap between the performance of the Seq2Seq model on two datasets is larger.

5.3 Error Analysis

Figure 5 shows some example errors of the deciphering results of our Seq2Seq model and Variational Decipher. One problem with the deciphering results is that the generated sentences have poor grammatical structure, as shown in Figure 5. This is mainly because the size of our dataset is small, and the models need a much larger corpus to learn the grammar. We anticipate that the generation performance will be improved with a larger dataset.

For the hate symbols in D_s, the deciphering results are of high quality when the length of referential definitions are relatively short. An example is *macaca*, a French slur shows in Figure 5. The deciphering result of the Seq2Seq model is close to the referential definition. As to the Variational Decipher, although the result is not literally the same as the definition, the meaning is close. *closet homosexuals* in Figure 5 is another example. However, when the length of the referential definition increases, the performance of both models tends to be unsatisfactory, as the third example *confederate flag* shows in Figure 5. Although there exists the symbol *Confederate Flag* with the same definition in the training set, both models fail on this example. One possible reason is that the complexity of generating the referential definition grows substantially with the increasing length, so when the tweet context and the symbol itself cannot provide enough information, the generation model cannot learn the relationship between the symbol and its definition.

Deciphering hate symbols in D_d is much more challenging. Even for humans, deciphering completely new hate symbols is not a simple task. The two examples in Figure 5 show that the models have some ability to capture the semantic similarities. For the symbol *niggering*, the Variational Decipher generates the word *nigger* and Seq2Seq model generates *black*. For *Heil Hitler*, the Variational Decipher generates *leader person* and *Nazi*, while Seq2Seq also generates *Nazi*. Although these generated words are not in the definition, they still make some sense.

6 Conclusion

We propose a new task of learning to decipher hate symbols and create a symbol-rich tweet dataset. We split the testing dataset into two parts to analyze the characteristics of the Seq2Seq model and the Variational Decipher. The different performance of these two models indicates that the models can be applied to different scenarios of hate symbol deciphering. The Seq2Seq model outperforms the Variational Decipher for deciphering the hate symbols with similar definitions to that in the training dataset. This means the Seq2Seq model can better explain the hate symbols when Twitter users intentionally misspell or abbreviate common slur terms. On the other hand, the Variational Decipher tends to be better at deciphering hate symbols with unseen definitions, so it can be applied to explain newly created hate symbols on Twitter. Although both models show promising deciphering results, there still exists much room for improvement.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Richard Beaufort, Sophie Roekhaut, Louise-Amélie Cougnon, and Cédrick Fairon. 2010. A hybrid rule/model-based finite-state framework for normalizing SMS messages. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 770–779. Association for Computational Linguistics.

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.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Grzegorz Chrupała. 2014. Normalizing tweets with edit scripts and recurrent neural embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 680–686.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Nemanja Djuric, Jing Zhou, Robin Morris, Mihajlo Grbovic, Vladan Radosavljevic, and Narayan Bhamidipati. 2015. Hate speech detection with comment embeddings. In *Proceedings of the 24th ACM International Conference on World Wide Web*, pages 29–30.

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*, pages 774–782.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 368–378. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics.

Felix Hill, KyungHyun Cho, Anna Korhonen, and Yoshua Bengio. 2016. Learning to understand phrases by embedding the dictionary. *Transactions of the Association of Computational Linguistics*, 4:17–30.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations*.

Kevin Knight, Anish Nair, Nishit Rathod, and Kenji Yamada. 2006. Unsupervised analysis for decipherment problems. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 499–506.

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 33nd International Conference on Machine Learning*, pages 1558–1566.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044. Association for Computational Linguistics.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 71–76. Association for Computational Linguistics.

Ke Ni and William Yang Wang. 2017. Learning to explain non-standard english words and phrases. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 413–417.

Thanapon Noraset, Chen Liang, Larry Birnbaum, and Doug Downey. 2017. Definition modeling: Learning to define word embeddings in natural language. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1125–1135.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 12–21. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on 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 *Proceedings of the 32nd International Conference on Machine Learning*, pages 1218–1226.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*, pages 3483–3491.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112.

Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on Twitter. In *Proceedings of the Student Research Workshop, SRW@HLT-NAACL 2016*, pages 88–93.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399.

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.

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.