Coded Hate Speech Detection via Contextual Information

Depeng Xu[1], Shuhan Yuan[2](✉), Yueyang Wang[3], Angela Uchechukwu Nwude[1], Lu Zhang[1], Anna Zajicek[1], and Xintao Wu[1]

[1] University of Arkansas, Fayetteville, AR 72701, USA
{depengxu,aunwude,lz006,azajicek,xintaowu}@uark.edu
[2] Utah State University, Logan, UT 84322, USA
shuhan.yuan@usu.edu
[3] Paycom Software, Inc., Oklahoma City, OK 73142, USA
Yueyang.wang@paycomonline.com

Abstract. Hate speech on online social media seriously affects the experience of common users. Many online social media platforms deploy automatic hate speech detection programs to filter out hateful content. To evade detection, coded words have been used to represent the targeted groups in hate speech. For example, on Twitter, "Google" is used to indicate African-Americans, and "Skittles" is used to indicate Muslim. As a result, it would be difficult to determine whether a hateful text including "Google" targets African-Americans or the search engine. In this paper, we develop a coded hate speech detection framework, called CODE, to detect hate speech by judging whether coded words like Google or Skittles are used in the coded meaning or not. Based on a proposed two-layer structure, CODE is able to detect the hateful texts with observed coded words as well as newly emerged coded words. Experimental results on a Twitter dataset show the effectiveness of our approach.

Keywords: Coded hate speech · Few-shot learning

1 Introduction

Online social media bring people together and encourage people to share their thoughts freely. However, due to the openness of social media, some users misuse the platforms to promote hateful language. As a result, hate speech, which "expresses hate or encourages violence towards a person or group based on characteristics such as race, religion, sex, or sexual orientation"[1], unfortunately becomes a common phenomenon on online social media. Hate speech on online social media not only seriously affects the experience of regular users, but also could lead to a real-world consequence. Currently, many online social media, such as Facebook and Twitter, have a policy prohibiting hate speech. Users who violate the policy could result in permanent account suspension.

[1] https://dictionary.cambridge.org/dictionary/english/hate-speech.

These companies also deploy programs to automatically filter out hateful content. In early 2016, Google unveiled a tech incubator Jigsaw to reduce online hate and harassment. In response to these programs, trolls started the "Operation Google Campaign", which replaces racial slurs with names of technology brands and products. For example, when racists publish hate speech to attack African-American, instead of using the n-word that can be easily detected, they use the word "Google" to represent African-American, such as "worthless google, kill yourself"[2]. Since then, more coded words are proposed to represent different targeted groups to avoid the censorship of hate speech. Table 1 shows some coded words and their corresponding targeted groups. We call such hate speech using coded words the *coded hate speech*.

Table 1. Commonly-used coded words

Coded word	Targeted group	Coded word	Targeted group
Google	African-American	Butterfly	Gay
Skittles	Muslim	Car Salesman	Liberal
Pepe	Alt-right	Bing	Asian
M&M	Mexican & Muslim	Yahoo	Mexican

By replacing the targeted groups with specific coded words, it becomes an uneasy task to detect whether a text is hateful or not because it is difficult to determine whether these words are in the coded meanings. In this paper, we detect the hateful texts by distinguishing the meanings of coded words based on their contexts. Specifically, we consider coded words as a special case of polysemy, i.e., a single word is associated with two or more different meanings. In our case, each coded word is associated with a regular meaning, e.g., Google is a search engine, and Skittles is a brand of candies, and also a coded meaning, e.g., African-American or Muslin. We consider that if a coded word is in its coded meaning, the text containing the coded word expresses hate.

Given a training dataset $\mathcal{T}$ consisting of regular and hateful texts and collected with a set of coded words $\mathcal{C}$, we aim to build a classifier that is able to: 1) detect the hateful texts with some observed coded words in $\mathcal{C}$; and 2) detect the hateful texts with new coded words. To achieve these goals, there are three challenges: 1) how to represent the coded word with two different meanings; 2) how to detect whether a coded word is used in its coded or regular meaning; 3) how to detect new coded words without collecting new labeled texts.

In this work, we propose a **CO**ded hate speech **DE**tection (CODE) framework, which is able to detect the observed coded words $\mathcal{C}$ as well as newly emerged coded words. CODE makes use of Embeddings from Language Models (ELMo) [10] as the pre-trained word embedding model to derive the contextual embeddings of coded words. ELMo is capable of handling the polysemy by

[2] https://knowyourmeme.com/memes/events/operation-google.

representing a word in different texts with different word embeddings. CODE then applies transformations on the derived embeddings via a two-layer structure (general and specific layers) to detect the meaning of a coded word. In particular, the general layer derives a generalized coded meaning anchor based on all the observed coded hate speech and makes the coded words in the coded meanings close to the anchor. We then expect a new coded word in the coded meaning also close to this generalized coded meaning anchor. The specific layer then derives the specialized coded meaning anchor for each observed coded word to detect whether a coded word is used in its coded meaning. Experimental results on a collected Twitter dataset indicate that CODE can detect hate speech by distinguishing the coded and regular meanings of coded words, even the new coded words without collecting new training samples.

2 Related Work

Hate speech detection has attracted increasing attention in recent years [2,14]. Research in [3] develops a semantic dictionary of hate domain and uses a rule-based classifier to detect hate speech. Research in [1,16] uses a bag-of-word model to derive the features of hate speech and then adopt classical machine learning models such as SVM, logistic regression, random forests to detect the hate speech. Recently, deep learning models, such as recurrent neural networks or deep auto-encoder, are also proposed for hate speech detection [9,11–13,15].

There are few studies focusing on coded hate speech detection. Research in [6] composes a balanced dataset and adopt SVM to predict the coded hate tweets. In this work, we consider the problem of coded hate speech detection given both regular and hateful tweets while only a small number of coded hate tweets is available for training, which is closer to the real-world scenario. Research in [7] aims at identifying coded words from hateful tweets with a known coded word. The proposed approach can only detect the coded words that co-occur with the known coded word within the same corpus. Meanwhile, it cannot detect whether a coded word is used in the coded or regular meaning given a new tweet.

3 Method

Given a set of coded words $\mathcal{C}$ and related hateful texts $\mathcal{T}^h$ and regular texts $\mathcal{T}^r$, for each coded word $c \in \mathcal{C}$, there are associated hateful texts $\mathcal{T}_c^h \subset \mathcal{T}^h$ and regular texts $\mathcal{T}_c^r \subset \mathcal{T}^r$. Note that in most cases, the coded word is still used as its regular meaning, i.e., $|\mathcal{T}_c^r| > |\mathcal{T}_c^h|$. In this work, we aim to detect a hateful text by determining whether an observed coded word $c \in \mathcal{C}$ or a new coded word $c \notin \mathcal{C}$ in the text is in the coded meaning based on the training corpus $\{\mathcal{T}^r, \mathcal{T}^h\}$.

We propose a **CO**ded hate speech **DE**tection (CODE) framework as shown in Fig. 1. CODE uses ELMo to derive contextual embeddings for coded words and then applies transformations on the derived embeddings via a two-layer structure. CODE derives generalized coded and regular meaning anchors and makes the coded words with coded meanings close to the coded meaning anchor and away from the regular meaning anchor by training on $\mathcal{T}^r$ and $\mathcal{T}^h$. We expect if a new coded word $c \notin \mathcal{C}$ is in its coded meaning, it should be also close to the generalized coded meaning anchor. For the observed coded word $c \in \mathcal{C}$, we have the specific texts $\mathcal{T}_c^r$ and $\mathcal{T}_c^h$. CODE further derives the specialized coded and regular meaning anchors for each observed coded word c based on $\mathcal{T}_c^r$ and $\mathcal{T}_c^h$ so that CODE can leverage the specialized anchors to detect the hate speech.

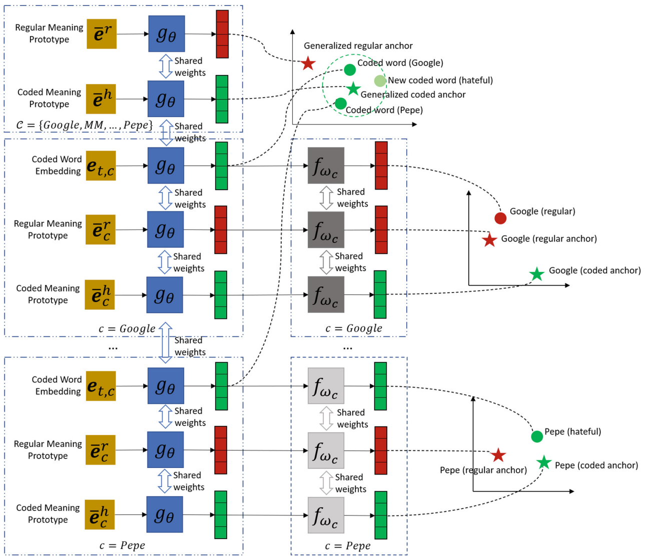


Fig. 1. The proposed CODE framework

3.1 Coded Word Representation

We use word embeddings to represent coded words. However, the traditional word embedding approaches, such as word2vec, usually represent one word by one embedding, which is unable to capture different meanings of coded words [8]. In this work, we use ELMo to derive the coded word embeddings, which addresses the issue of polysemy by representing each word based on its context [10]. As a result, one word has different representations in different contexts. We represent the coded word c in a text t as $\mathbf{e}_{t,c} = \mathbf{ELMo}_t(c)$.

3.2 Transformation Layers

After obtaining $\mathbf{e}_{t,c}$, we aim to detect whether the coded word c is in its coded meaning. CODE consists of one general layer and one specific layer, both of which make transformations to separate the regular and coded meanings of coded words in the embedding space.

General Layer. The general layer of CODE is to identify the generalized coded meanings. In general, for different coded words, although the regular meanings could be different, the coded meanings should be all alike since they all express the social hate towards specific groups. Based on this hypothesis, we first derive prototype representations to represent the regular and coded meanings by using ELMo on $\mathcal{T}^r$ and $\mathcal{T}^h$. Then, we adopt a neural network as the general layer to apply a transformation on the original embedding as well as the prototypes. After we apply the transformation, we consider the transformed prototype representations of regular and coded meanings as the generalized regular and coded meaning anchors, respectively. The objective is to make the transformed coded word embedding in different meanings close to their corresponding anchors. For example, if "Google" is used to indicate the African-American in a text, we aim to make it close to the coded meaning anchor. Because the generalized coded meaning anchor is derived from several coded words $\mathcal{C}$, the anchor should represent the generalized coded meaning. Hence, after training, given a new coded word $c \notin \mathcal{C}$, if the coded word is used in its coded meaning, we expect the transformed embedding has a smaller distance to the coded meaning anchor compared with the distance to the regular meaning anchor.

More specifically, given a set of regular and hateful texts $\mathcal{T}^r$ and $\mathcal{T}^h$, we first adopt ELMo to obtain the word embeddings $\mathbf{e}_{t,c}$ of the coded word c in text t. Note that for ELMo, the coded word has a unique word embedding for each text t. Based on the obtained coded word embedding, we consider the centroid of coded word embeddings in regular or coded meaning as the corresponding prototype representation. Symbolically, the prototype representations of regular meaning and coded meaning of a set of coded words $\mathcal{C}$ are computed by a mean operation, respectively:

$$\bar{\mathbf{e}}^r = \frac{1}{\sum\limits_{c \in \mathcal{C}} |\mathcal{T}_c^r|} \sum_{c \in \mathcal{C}, t \in \mathcal{T}_c^r} \mathbf{e}_{t,c} \quad \bar{\mathbf{e}}^h = \frac{1}{\sum\limits_{c \in \mathcal{C}} |\mathcal{T}_c^h|} \sum_{c \in \mathcal{C}, t \in \mathcal{T}_c^h} \mathbf{e}_{t,c}, \tag{1}$$

where $\bar{\mathbf{e}}^r$ and $\bar{\mathbf{e}}^h$ indicate the prototype representations of regular and coded meanings, respectively.

Then, we aim to map the coded word embedding derived from each text close to the prototype representation based on its meaning via a neural network. To this end, we adopt a neural network g to make a transformation on the coded word embedding in each text t as well as the prototype representations:

$$\mathbf{g}_{t,c} = g_\theta(\mathbf{e}_{t,c}) \quad \bar{\mathbf{g}}^r = g_\theta(\bar{\mathbf{e}}^r) \quad \bar{\mathbf{g}}^h = g_\theta(\bar{\mathbf{e}}^h), \tag{2}$$

where θ indicates the parameters of the neural network g. We consider $\bar{\mathbf{g}}^r$ and $\bar{\mathbf{g}}^h$ as the generalized regular and coded meaning anchors, respectively.

In order to separate $\bar{\mathbf{g}}^r$ and $\bar{\mathbf{g}}^h$ and to make the coded word embedding $\mathbf{g}_{t,c}$ close to the corresponding anchor, the objective of the neural network g_θ is to make the word embedding in the coded meaning (regular meaning) close to the generalized coded meaning anchor (regular meaning anchor) and far from the regular meaning anchor (coded meaning anchor). To this end, we propose the following triplet loss function to train the neural network g_θ:

$$\begin{aligned}\mathcal{L}_{tri} = & \frac{1}{\sum\limits_{c\in\mathcal{C}} |\mathcal{T}_c^h|} \sum_{c\in\mathcal{C}, t\in\mathcal{T}_c^h} \max\{d(\bar{\mathbf{g}}^h, \mathbf{g}_{t,c}) - d(\bar{\mathbf{g}}^r, \mathbf{g}_{t,c}) + \alpha, 0\} \\ & + \frac{1}{\sum\limits_{c\in\mathcal{C}} |\mathcal{T}_c^r|} \sum_{c\in\mathcal{C}, t\in\mathcal{T}_c^r} \max\{d(\bar{\mathbf{g}}^r, \mathbf{g}_{t,c}) - d(\bar{\mathbf{g}}^h, \mathbf{g}_{t,c}) + \alpha, 0\},\end{aligned} \tag{3}$$

where α is the margin and $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$.

Besides the triplet loss, as discussed above, for the coded hate speech, if the coded words are in the coded meanings, they should group together in the embedding space because the contexts of different coded words are similar, i.e., expressing the social hate. To this end, we further aim to minimize the distance between the word embedding in coded meaning $\mathbf{g}_{t,c}$ and the generalized coded meaning anchor $\bar{\mathbf{g}}^h$ based on the mean squared loss:

$$\mathcal{L}_{mse} = \frac{1}{\sum\limits_{c\in\mathcal{C}} |\mathcal{T}_c^h|} \sum_{c\in\mathcal{C}, t\in\mathcal{T}_c^h} (\mathbf{g}_{t,c} - \bar{\mathbf{g}}^h)^2. \tag{4}$$

Finally, the objective function for the general layer is defined as:

$$\mathcal{L}_g = \mathcal{L}_{tri} + \lambda \mathcal{L}_{mse}, \tag{5}$$

where λ is a hyper-parameter.

Specific Layer. The general layer mainly derives the generalized coded meaning anchor to detect the coded hate speech. Because we have the labeled corpus for each coded word $c \in \mathcal{C}$, we can leverage the regular and hateful texts related with each coded word, $\mathcal{T}_c^r$ and $\mathcal{T}_c^h$, to further improve the performance of specific hate speech detection. To this end, we propose the specific layer to detect whether a coded word $c \in \mathcal{C}$ is in its coded meaning. In the specific layer, we derive the prototype representations of regular and coded meanings for each coded word c, respectively, and then adopt another neural network to finetune the coded word representation to make it close to the corresponding prototype.

Given a coded word $c \in \mathcal{C}$, we adopt the mean operation to derive prototype representations of regular and coded meanings, $\bar{\mathbf{e}}_c^r$ and $\bar{\mathbf{e}}_c^h$, respectively.

$$\bar{\mathbf{e}}_c^r = \frac{1}{|\mathcal{T}_c^r|} \sum_{t\in\mathcal{T}_c^r} \mathbf{e}_{t,c} \quad \bar{\mathbf{e}}_c^h = \frac{1}{|\mathcal{T}_c^h|} \sum_{t\in\mathcal{T}_c^h} \mathbf{e}_{t,c}. \tag{6}$$

Then, given $\mathbf{e}_{t,c}$, $\bar{\mathbf{e}}_c^r$ and $\bar{\mathbf{e}}_c^h$, we first use general layer to derive $\mathbf{g}_{t,c}$ by Eq. (2). Similarly, we have $\bar{\mathbf{g}}_c^r = g_\theta(\bar{\mathbf{e}}_c^r)$, $\bar{\mathbf{g}}_c^h = g_\theta(\bar{\mathbf{e}}_c^h)$. After that, we adopt a neural network f_{ω_c} to further derive the specialized representation for $c \in \mathcal{C}$.

$$\mathbf{s}_{t,c} = f_{\omega_c}(\mathbf{g}_{t,c}) \quad \bar{\mathbf{s}}_c^r = f_{\omega_c}(\bar{\mathbf{g}}_c^r) \quad \bar{\mathbf{s}}_c^h = f_{\omega_c}(\bar{\mathbf{g}}_c^h). \tag{7}$$

It is worth noting that to achieve the specific hate speech detection, we train a unique neural network f_{ω_c} for each coded word c. We consider $\bar{\mathbf{s}}_c^r$ and $\bar{\mathbf{s}}_c^h$ as the specialized regular and coded meaning anchors for the coded word c.

To train the neural network, for each coded word c, we adopt the triplet loss as the objective function to make $\mathbf{s}_{t,c}$ close to the corresponding anchors.

$$\begin{aligned}\mathcal{L}_{s,c} = & \frac{1}{|\mathcal{T}_c^h|} \sum_{t \in \mathcal{T}_c^h} \max\{d(\bar{\mathbf{s}}_c^h, \mathbf{s}_{t,c}) - d(\bar{\mathbf{s}}_c^r, \mathbf{s}_{t,c}) + \alpha, 0\} \\ & + \frac{1}{|\mathcal{T}_c^r|} \sum_{t \in \mathcal{T}_c^r} \max\{d(\bar{\mathbf{s}}_c^r, \mathbf{s}_{t,c}) - d(\bar{\mathbf{s}}_c^h, \mathbf{s}_{t,c}) + \alpha, 0\}.\end{aligned} \tag{8}$$

Overall, CODE is trained in an end-to-end manner. The complete objective function is defined as below:

$$\mathcal{L} = \mathcal{L}_g + \lambda' \sum_{c \in \mathcal{C}} \mathcal{L}_{s,c}, \tag{9}$$

where λ' is a hyperparameter to balance two layers.

3.3 Coded Hate Speech Detection

CODE can detect the hateful texts with observed and new coded words.

Coded Hate Speech Detection on Observed Coded Words. Given a new text t with an observed coded word $c \in \mathcal{C}$, CODE derives the coded word representation $\mathbf{s}_{t,c}$ from the specific layer and then compares the distances from $\mathbf{s}_{t,c}$ to the specialized regular and coded meaning anchors ($\bar{\mathbf{s}}_c^r$ and $\bar{\mathbf{s}}_c^h$), respectively. If the coded word representation is closer to the specialized coded meaning anchors, i.e., $d(\bar{\mathbf{s}}_c^h, \mathbf{s}_{t,c}) < d(\bar{\mathbf{s}}_c^r, \mathbf{s}_{t,c})$, the text t will be predicted as hate speech.

Coded Hate Speech Detection on New Coded Words. Given a new text t with a new coded word $c \notin \mathcal{C}$, CODE leverages the general layer for coded meaning detection because there is no specialized network built for the new coded word. First, CODE derives the coded word representation $\mathbf{g}_{t,c}$ from the general layer. It then compares the distances from $\mathbf{g}_{t,c}$ to the generalized regular and coded meaning anchors ($\bar{\mathbf{g}}^r$ and $\bar{\mathbf{g}}^h$), respectively. If the coded word representation is closer to the generalized coded meaning anchor, i.e., $d(\bar{\mathbf{g}}^h, \mathbf{g}_{t,c}) < d(\bar{\mathbf{g}}^r, \mathbf{g}_{t,c})$, CODE will predict the text t as hate speech.

4 Experiments

4.1 Experimental Setup

Coded Hate Speech Corpus. In this work, we evaluate our approach by using the coded hate speech on Twitter. We first collect two sets of tweets, i.e., the benign and hateful tweets, for several selected coded words shown in Table 2. In benign tweets, the coded word is used in its regular meaning, while in hateful tweets, the coded word is used in coded meaning targeting a specific group. In order to collect benign and hateful tweets, we combine different sets of keywords with the coded word as query words to search tweets on Twitter, where the query words are chosen based on the regular and coded meanings of the coded word. For example, given the coded word Skittles, for crawling benign tweets, we compose a set of query words, including pack, eat, and kid, while for crawling hateful tweets, we use query words, such as Muslim, refugee, and terrorist combining with Skittles. By using this strategy, we collected 10000 benign tweets and 6342 hateful tweets. The detailed information of collected benign and hateful tweets for each coded word is shown in Table 2.

Table 2. The information of crawled benign and hateful tweets in the collection and the adopted training corpus size for each coded word

c	Benign	Hateful	Time period	$\|\mathcal{T}_c^r\|$	$\|\mathcal{T}_c^h\|$
Google	10000	69	09/20/16-04/09/18	300	20
M&M	10000	520	09/20/16-09/26/19	300	50
Pepe	10000	6499	08/16/17-09/26/19	300	50
Butterfly	10000	618	11/12/09-09/21/19	300	50
Skittles	10000	6342	09/20/16-10/01/19	0	0

Evaluation Tasks. We evaluate our model on two tasks. (1) *Coded hate speech detection on observed coded words.* We build a training corpus consisting of the coded words "Google", "M&M", "Pepe", and "Butterfly". We train a model with this corpus and evaluate its performance on new tweets containing these observed words. (2) *Coded hate speech detection on a new coded word.* Using the model built on observed coded words, we detect coded hate speech on new tweets containing an unobserved coded word, "Skittles".

As shown in Table 2, in order to simulate the unbalanced nature of coded hate speech, for the coded word, Google, we adopt 300 regular and 20 hateful tweets as labeled texts in training. For coded words, M&M, Pepe, and Butterfly, we adopt 300 regular and 50 hateful tweets in training. The remaining tweets in our collection for each coded word all serve as the testing set.

Hyperparameters. We adopt the pre-trained ELMo model[3] to derive coded word contextual embeddings. The dimension of coded word embeddings $\mathbf{e}_{t,c}$ is 2048. The dimension of the transformed general embeddings $\mathbf{g}_{t,c}$ is 1024. The dimension of the transformed word specific embeddings $\mathbf{s}_{t,c}$ is 512. The hyperparameters λ and λ' are set to 1 in default unless specified otherwise. Both neural networks g_θ and f_θ are fully connected networks with one hidden layer.

Baselines. We compare CODE with following baselines. (1) **LSTM** [4], which is a deep learning based binary classifier; (2) **ELMo**, which is based on the raw embeddings derived from ELMo and uses the same distance-based approach

[3] https://github.com/allenai/allennlp/blob/v0.9.0/tutorials/how_to/elmo.md.

proposed in this work to detect coded hate speech. The purpose is to show the advantage of our two-layer framework; (3) **S model**, which removes the general layer and train the neural network for each coded word separately to detect coded hate speech as a comparison to show the advantage of the general layer.

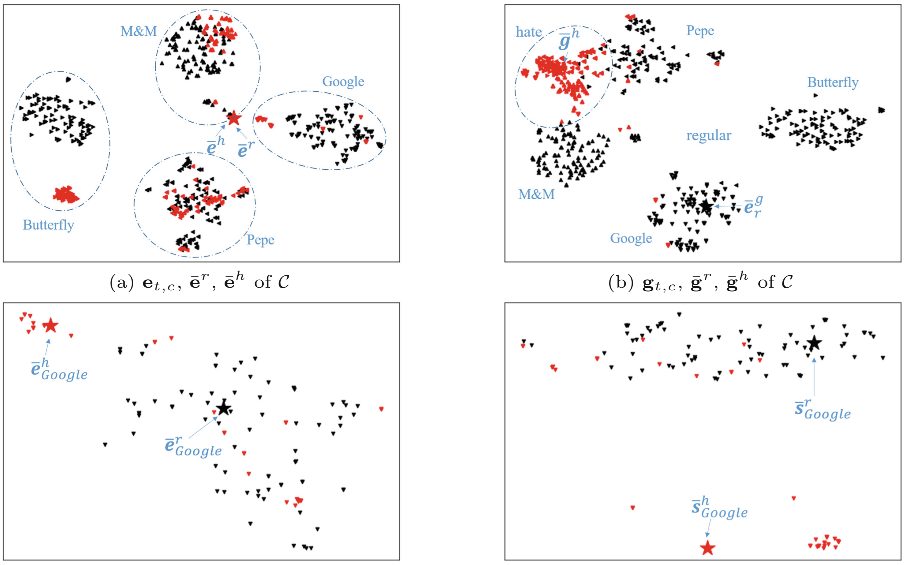
Table 3. Experimental results on coded hate speech detection for the observed (Google, M&M, Pepe, and Butterfly) and unobserved (Skittles) coded words in terms of precision (P), recall (R) and F1-score (F1)

Model	Google			M&M			Pepe			Butterfly			*Skittles*		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
LSTM	0.589	0.524	0.555	0.579	0.392	0.468	0.937	0.489	0.643	0.933	0.997	0.963	0.674	0.548	0.605
ELMo	1.000	0.490	0.658	0.437	0.871	0.582	0.854	0.592	0.699	0.957	1.000	0.978	0.573	0.863	0.689
S model	0.821	0.653	0.727	0.600	0.758	0.670	0.919	0.558	0.694	0.957	1.000	0.978	-	-	-
CODE	0.889	0.653	**0.753**	0.630	0.715	**0.670**	0.871	0.664	**0.754**	0.959	1.000	**0.979**	0.689	0.855	**0.763**

4.2 Experimental Results

Coded Hate Speech Detection on Observed Coded Words. Table 3 shows experimental results on coded hate speech detection for each observed coded word, Google, M&M, Pepe, and Butterfly. For most coded words, our approach significantly outperforms the LSTM model. For LSTM, it usually requires a large amount of training data and a relatively balanced dataset between positive and negative labels. However, it is mostly not the case when we are dealing with coded hate speech. In comparison to ELMo, the post-transformation models (S model and CODE) have improvement on the F1-score. It indicates that training a coded hate speech detection transformation could further separate the coded word embeddings in regular and coded meanings than the original coded word embeddings by the pre-trained ELMo. For the coded word "Butterfly", all models have a nearly perfect performance on coded hate speech detection, which means "Butterfly" in regular meanings and coded meanings are already separated in the original embedding space. However, for other coded words in our experiments, making the transformation on original coded word embeddings can boost the performance on coded hate speech detection with a large margin. In CODE, the transformation on the general layer leverages more training data from multiple coded words, which creates a better context of hateful tweets. The specific layer embedding further combines contexts specifically related to each coded word to further improve the model performance. Hence, CODE has the best performance in terms of the F1-score for all words.

Coded Hate Speech Detection on a New Coded Word. The last column of Table 3 shows experimental results on coded hate speech detection for the unobserved coded word, "Skittles". Tweets containing "Skittles" are not in the training set. The predictions are made by models trained on other words. Similar to the first task, LSTM does not have the best performance due to the same issues. The embeddings by ELMo before any transformation already can achieve 0.689 F1-score. It indicates that there is a similarity in the context of the hateful tweets across different coded words. The general layer transformation in our approach further extracts the hateful context into a general layer embedding, which has the best performance of 0.763 on the new coded hate detection. The S model is not applicable here because it can only detect observed coded words. However, if we compare CODE with the S model trained only on tweets containing Skittles (F1-score: 0.702), our approach, which benefits from more training tweets, even outperforms it.



(a) $\mathbf{e}_{t,c}$, $\bar{\mathbf{e}}^r$, $\bar{\mathbf{e}}^h$ of $\mathcal{C}$

(b) $\mathbf{g}_{t,c}$, $\bar{\mathbf{g}}^r$, $\bar{\mathbf{g}}^h$ of $\mathcal{C}$

(c) $\mathbf{e}_{t,c}$, $\bar{\mathbf{e}}^r_c$, $\bar{\mathbf{e}}^h_c$ of “Google”

(d) $\mathbf{s}_{t,c}$, $\bar{\mathbf{s}}^r_c$, $\bar{\mathbf{s}}^h_c$ of “Google”

Fig. 2. The visualizations of coded word embeddings. The red and black colors indicate the coded and regular meanings, respectively. The triangles pointing down, up, left, and right indicate Google, M&M, Pepe, and Butterfly, respectively. The stars indicate the respected prototypes of coded/regular meanings. (Color figure online)

Visualization. We randomly select 300 regular tweets and 50 coded hateful tweets from the testing set for each coded word. We adopt TSNE [5] to project coded word embeddings to a two-dimensional space and visualize regular and coded meanings of each word in the corpus. Figures 2 (a) and (b) show the visualization results of regular and coded meanings as well as two prototype representations before and after the transformations. We can observe that before conducting the general layer transformation on the contextual embeddings, the embeddings of regular and coded meanings are mixed within the same words, and the prototypes of regular and coded meanings are also close to each other.

The embedding space is divided by words. There is no obvious general trend to detect hateful meanings for all words. After conducting the general layer transformation, the coded word embeddings in regular and coded meanings are clearly separated, and the prototypes of the two meanings are also separated. Meanwhile, most of the coded word embeddings in coded meanings are close to the prototype of coded meaning regardless of the regular meaning of the coded words, which meets our assumption that all hateful tweets have similar hateful contexts. The embedding space is divided by hateful or regular meanings. It indicates that our model exploits the common space for all coded words in the direction of hatefulness. We use the coded word "Google" as an example for the coded word specific layer embedding visualization. Figures 2 (c) and (d) show the visualization results of regular and coded meanings before and after two layers of transformations. After the transformations, the regular and coded meanings of "Google" are better separated in the new embedding space.

Table 4. Parameter sensitivity of λ and λ' on observed (average of all observed words) and unobserved (Skittles) coded words

(λ, λ')	Observed			*Skittles*		
	P	R	F1	P	R	F1
(1, 1)	0.718	0.819	0.763	0.689	0.855	0.763
(0.1, 1)	0.807	0.734	0.768	0.703	0.803	0.750
(10, 1)	0.608	0.874	0.718	0.660	0.911	0.765
(1, 0.1)	0.683	0.834	0.751	0.705	0.874	0.781
(1, 10)	0.783	0.779	0.781	0.685	0.819	0.746

Parameter Sensitivity. We also evaluate the parameter sensitivity of λ and λ' defined in Eqs. (5) and (9), respectively. We evaluate on observed (average of all observed words) and unobserved (Skittles) coded words. The results are shown in Table 4. λ controls the relative weight of MSE loss in comparison to triplet loss in the general layer. We change λ while keeping $\lambda' = 1$. When $\lambda = 0.1$, F1-score of the unobserved word decreases. When $\lambda = 10$, the average F1-score of the observed words decreases.

λ' controls the relative weight of the loss for specific layer in comparison to general layer in the overall objective function. We change λ' while keeping $\lambda = 1$. As λ' controls the trade-off of generalization and specialization of the whole model, the increase of $\lambda' = (0.1, 1, 10)$ increases the average F1-score of the observed words but decreases F1-score of the unobserved word.

5 Conclusion

In this paper, we have developed CODE to achieve coded hate speech detection by determining whether the coded word is in its coded meaning or not. We treat each coded word as a polysemy and use ELMo to map the coded word in each text to an embedding space. Based on that, we have developed a two-layer (general layer and specific layer) transformation approach. The general layer derives a generalized coded meaning anchor and makes the coded word in its coded meaning close to the anchor, while the specific layer derives a specialized coded meaning anchor for each observed coded word and also makes the specific coded word in its coded meaning close to the specialized coded meaning anchor. CODE can detect the hate speech with either observed or newly emerged coded words by comparing the distance to the specialized or generalized anchor, separately. Experimental results on a Twitter dataset show that our approach can effectively detect coded hate speech and significantly outperform the baseline methods. In the future, we plan to study how to identify newly emerged coded words based on the existing coded hate speech corpus.

Acknowledgement. This work was supported in part by NSF grants 1564250, 1946391, and 2103829.

References

1. Davidson, T., Warmsley, D., Macy, M., Weber, I.: Automated hate speech detection and the problem of offensive language. In: ICWSM (2017)
2. Fortuna, P., Nunes, S.: A survey on automatic detection of hate speech in text. ACM Comput. Surv. **51**(4) (2018)
3. Gitari, N.D., Zuping, Z., Damien, H., Long, J.: A lexicon-based approach for hate speech detection. Int. J. Multimed. Ubiquit. Eng. **10**(4), 215–230 (2015)
4. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
5. van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
6. Magu, R., Joshi, K., Luo, J.: Detecting the hate code on social media. In: Eleventh International AAAI Conference on Web and Social Media (2017)
7. Magu, R., Luo, J.: Determining code words in euphemistic hate speech using word embedding networks. In: ALW (2018)
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
9. Mou, G., Ye, P., Lee, K.: SWE2: SubWord enriched and significant word emphasized framework for hate speech detection. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1145–1154. Association for Computing Machinery, New York, October 2020
10. Peters, M., et al.: Deep contextualized word representations. In: NAACL (2018)
11. Qian, J., ElSherief, M., Belding, E., Wang, W.Y.: Hierarchical CVAE for fine-grained hate speech classification. In: EMNLP (2018)
12. Qian, J., ElSherief, M., Belding, E., Wang, W.Y.: Leveraging intra-user and inter-user representation learning for automated hate speech detection. In: NAACL (2018)
13. Rajamanickam, S., Mishra, P., Yannakoudakis, H., Shutova, E.: Joint modelling of emotion and abusive language detection. In: ACL, May 2020

14. Schmidt, A., Wiegand, M.: A survey on hate speech detection using natural language processing. In: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media (2017)
15. Tran, T., et al.: HABERTOR: an efficient and effective deep hatespeech detector. In: EMNLP, October 2020
16. Waseem, Z., Hovy, D.: Hateful symbols or hateful people? Predictive features for hate speech detection on twitter. In: Proceedings of the NAACL Student Research Workshop (2016)