Received: 29 January 2021 | Revised: 17 July 2021 | Accepted: 9 November 2021

DOI: 10.1002/cpe.6754

SPECIAL ISSUE PAPER

WILEY

TM-HOL: Topic memory model for detection of hate speech and offensive language

Jing Chen[1] | Kun Ma[1,2] | Ke Ji[1,2] | Zhenxiang Chen[1,2]

[1]College of Information Science and Engineering, University of Jinan, Jinan, China

[2]Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan, China

Correspondence
Kun Ma, Shandong Provincial Key Laboratory of Network Based Intelligent Computing, University of Jinan, Jinan 250022, China.
Email: ise_mak@ujn.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Number: 61772231; Natural Science Foundation of Shandong Province, Grant/Award Number: ZR2017MF025; Project of Independent Cultivated Innovation Team of Jinan City, Grant/Award Number: 2018GXRC002; Project of Shandong Provincial Social Science Program, Grant/Award Number: 18CHLJ39; Shandong Provincial Key R&D Program of China, Grant/Award Number: 2021CXGC010103

Abstract

In the era of the explosion of digital content of large-scale self-media, user-friendly social platforms such as Twitter and Facebook, provide opportunities for people to express their ideas and opinions freely. Due to lack of restrictions, hateful speech and its exposure can have profound psychological impacts on society. Current social networking platform is over-reliant on the manual check, and it is labor-intensive and time-consuming. Although there are many machines learning methods for the detection of hate speech, short text with character limit on social platforms is more challenging for the detection of hate speech and offensive language. To address the problem of data sparsity, we have proposed a topic memory model for hate speech and offensive language detection (abbreviated as TM-HOL). Potential topics are generated with our encoder and decoder to enrich short text features. Two memory matrices correspond to the topic words and the text, and the hate feature matrix is used to learn the syntactic features. It is demonstrated that our proposed method is effective on three datasets, performing better weighted-F1.

KEYWORDS

Big data processing, hate speech, offensive language, text classification, topic model

1 | INTRODUCTION

Worldwide accessibility to digital Internet resources has incredibly reshaped the way that people express ideas and opinions of the world.[1] Social network services were created to bring people together with different backgrounds and cultures,[2] such as Twitter, Weibo, Facebook, Telegram, WhatsApp, and WeChat.[2,3] Since no clear standard was applied to government regulation of online speech on the Internet,[4] speeches on the current social networks are extremely subjective and biased. Especially, people often fail in controlling the use of social media when it conflicts with other goals and obligations.[5]

Nowadays, unrestrained speeches are flooding the Internet.[6] Hate speeches seriously affect people's online experience and it may even lead to crime.[7] Many governments in Europe have not only criminalized hate speech, but they are also actively prosecuting artists, curators, and writers accused of stirring anti-Muslim or anti-Christian hatred.[8] Several social platform companies have developed hundreds of rules to draw elaborate distinctions between what should and should not be allowed. For example, Facebook promised to delete illegal speeches manually within 24 h after they are reported.[9,10] However, there was this barrage of criticism that they were not doing enough.[11] The current process to manage the speeches is labor-intensive, time-consuming, and not sustainable or scalable in reality.[12,13] Therefore, it is an urgent need to develop a model and method to detect hate speech automatically.

In general, hate speech is defined as any statements that disparage individuals or groups of people on the basis of characteristics such as race, color, ethnicity, gender, sexual orientation, nationality, religion, or other characteristics.[14] Comparatively speaking, offensive language refers to

degrading speech that does not conform to etiquette, expressly, or implicitly. It is compatible with the distinctive characteristics of the recipient.[2] However, it is difficult to clearly determine which text contains hateful or offensive content. Generally, the speech is hiding behind irony or if no clear words showing hate, ethnicity, or stereotyping exist.[2] Furthermore, social media are full of irony and joking content that might sound racist, segregative, or offensive, which in reality is not. Some examples are given in the following:

- These muslim refugees are trouble-makers and parasites, they should be deported from my country.
- You might not get your bitch back.
- Our class prom night just got ruined because u showed up. Who invited u anyway?

The first sentence sounds hateful and demeaning to the group of Muslims, it is a typical example of hate speech. The second sentence is an example of offensive language, which is typically characterized by the use of swearing or curse words. The third sentence is another example of offensive language, which has the purpose to harass, threaten, or intimidate typically individuals rather than groups. If these three sentences are classified as hate speech, it can lead to a high rate of false positives. Therefore, the key to detecting accurate hate speech and in cyberspace is to distinguish the two concepts.[10]

There are three major challenges to detect hate speech and offensive language. First, it is a lack of a sufficient dataset in a broader context due to the confidentiality agreement.[15] Second, hate is a relatively subjective term with no single definition.[1,13] It is difficult to distinguish hate speech from freedom of speech. Third, most content is in the form of short text due to character limitation of social platforms.[16] The sparsity of data and the semantic sensitivity to context often hinders the classification performance of short texts.

Topic model is a popular method for learning representations of text.[17] Topic model is a frequently used text-mining tool for discovering the abstract "topic words" that occur in a collection of texts. Topic memory network (abbreviated as TMN) was proposed for short text classification.[18] TMN solves the problem of data sparsity by extracting latent topic words. Topic-Aware Neural Keyphrase Generation was proposed for Social Media Language.[19] This model is based on TMN. This model allows joint modeling of corpus-level latent topic representations, which helps alleviate the problem of data sparsity. In our previous work, CLSTM-TMN is an improved TMN model that was proposed to identify marketing intent.[20] This model can be mined not only time information but also exploitation of spatial information. But CLSTM-TMN has the problem of loss of overall features.

Motivated by the CLSTM-TMN, this paper has proposed a **T**opic **M**emory model for detection of **H**ate speech and **O**ffensive **L**anguage (abbreviated as TM-HOL). TM-HOL mainly includes three parts: neural topic model for hate speech and offensive language detection (abbreviated as NTM-HOL), topic memory mechanism for hate speech and offensive language detection (abbreviated as TMM-HOL), and classifier. The main contributions of our work are TMM-HOL and NTM-HOL:

- NTM-HOL: Potential topics are generated with our encoder and decoder to enrich short text features. The linear unit SELU layer in the model extracts topics. Neuron activations of NTM-HOL automatically converge toward zero mean and unit variance. This mechanism avoided the loss of essential information easily.
- TMM-HOL: Two memory matrices and a hate feature matrix are proposed in TMM-HOL. Two memory matrices correspond to the topic words and the text, and the hate feature matrix in the modified computing layer is used to learn the syntactic features. It can make sentences and features merge better. This mechanism can solve the problem of losing the overall characteristics.

The rest of this article is organized as follows. Section 2 provides a brief description of the existing works done in the area of hate speech and offensive language detection. Section 3 introduces our TM-HOL for hate speech and offensive language detection. Firstly, a brief description of data preprocessing and feature extraction is provided. Secondly, the overall architecture of TM-HOL is introduced in detail. Section 4provides details about the dataset used for the experiments and evaluation metrics. Experiments show that our method is effective. Section 5 provides a summary of our work and briefly described the future work.

2 | RELATED WORK

From the perspective of natural language processing (abbreviated as NLP), detection of hate speech and offensive language can be considered as a classification task: given a sentence, determine whether or not it contains hate speech and offensive language.[21] In the earliest work, some researchers confuse the concepts of hate speech and offensive language.[10] They labeled each of the texts with "hate" and "non-hate". Recently, some researchers have made a more fine-grained division.[22] The detection of hate speech and offensive language can be also seen as multiclass classification rather than binary classification.

2.1 | Binary classification

Several works have been done for the binary classification of hate speech detection.

HaterNet system was designed to identify and classify hate speech.[23] The model enriches token embeddings with TF-IDF and classifies tweets using LSTM and MLP. Hierarchical LSTM with attention achieved an accuracy of 66.6% while subword level LSTM model scored 69.8%. It achieved the highest AUC of 0.828. However, the model cannot implement multiclass classification of tweets, for example, racist, homophobic, and xenophobic.[23,24]

A machine learning model is used to detect hate speech in Hindi–English code-mixed social media text.[22] After preprocessing data with Facebook's pretrained word embedding library, this method used many machines learning classifiers, such as Support Vector Machines (SVM) and Random Forest (RF). This method cannot make finer classification of hate tweets to offensive, abusive, and profane likewise.

However, the differences between hate speech and offensive language are often with subtle linguistic distinctions.[10,25] The presence of hate speech should not be considered a binary (hate or nonhate) classification, and raters need more detailed instructions for the annotation.[21]

2.2 | Multiclass classification

The model in this section usually classified the text in a more fine-grained manner.

An approach was proposed to detect hate expressions on Twitter using emotional features, semantic features, Unigram, and other patterns that are collected from the training set.[2] Depending on replacing hateful content with a specific pattern and unigram features, this method might lead to the possibility of overlooking implicit hateful content with no clearly hateful pattern.[26]

Another work used a multiclass classifier to distinguish hate/offensive/neither from a crowd-sourced labeled dataset.[10] It extracted N-gram features weighted with its TF-IDF. They tested a large number of classifiers: logistic regression, Naive Bayes (NB), decision trees (DTs), RFs, and linear SVMs. However, almost 40% of hate speeches are misconceived.[10]

Another work used NB, SVM, and logistic regression as the classifiers to detect hate speech.[27] They designed a self-identified hateful community as training data for hate speech classifiers, and the dataset is annotated as Black, Plus, and Female. They used a sparse representation of unigrams with TF-IDF weight as the feature set. However, some noise disturbance of the dataset by a hateful community might generate incorrect results.[27]

A hate speech detection approach was proposed to identify hatred against vulnerable minority groups on social media[28] for vulnerable community identification. They extracted features by word *n*-grams and word embedding techniques and used RNN-GRU classifiers for hate speech detection. They were more robust to detect hate speech for different language datasets. However, there is a lack of development and testing of models using data from multiple social media platforms.[10] The study tended to focus on one platform. This mono-platform focus is problematic because there are no guarantees the models that researchers develop generalize well across platforms.[29]

Character *n*-grams have been shown to be appropriate for hate speech and offensive language tasks due to their ability to capture variations of words associated with hate.[13] They provided a data set of 16k sentences marked for hate speech, 3383 of the dataset is annotated as sexist, 1972 of the dataset is annotated as racist, and 11,559 is annotated as neither. But the various classes of data in the dataset are very unbalanced.

Another work proposed several methods such as convolutional neural network (CNN), LSTM, and FastText to detect sexism and racist speech.[30] They explored various sentence semantic embeddings like char *n*-grams, TF-IDF values, BoW over Global Vectors for Word Representation (GloVe). But they experimented with a dataset annotated by Reference 13. The various types of data in the dataset are very unbalanced, it is hard to prove the effectiveness of the result when the experiment uses only one dataset.

Ensemble stacking classifier was proposed to detect offensive language and hate speech in the Arabic language.[31] They built a stacking classifier that is an ensemble of a NB classifier, a Logistic Regression model, a SVM, a Nearest Neighbors classifier, and a RF. The resulting ESOTP-based system ranked sixth out of 35 on the shared task of Offensive Language detection (subtask A) and fifth out of 30 on Hate Speech Detection (subtask B).

3 | OUR PROPOSED METHOD

In this section, we have proposed the topic memory model for the detection of hate speech and offensive language (TM-HOL).

3.1 | Data preprocessing

We examine all the extracted texts, finding that there was a presence of noisy content. Some examples are given in the following:

- RT @mayasolovely: As a woman you should not complain about cleaning up your house. & as a man you should always take the trash out …
- RT @mleew17: boy dats cold … tyga dwn bad for cuffin dat hoe in the 1st place!!

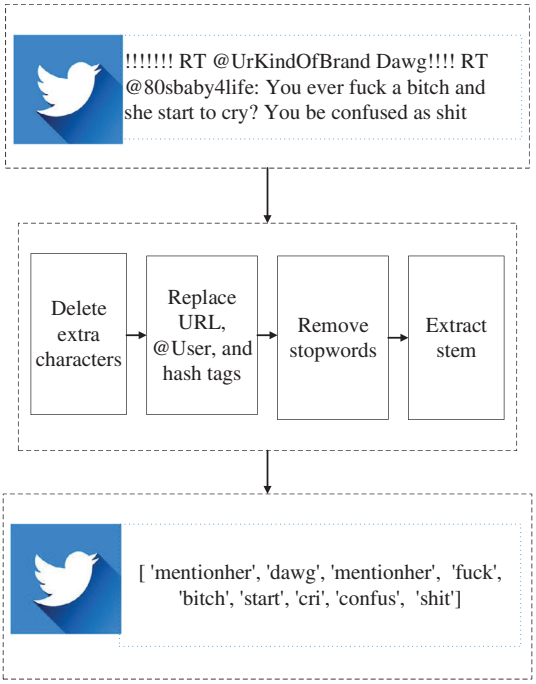

FIGURE 1 An example of data preprocessing

It contains a lot of irrelevant characters and stopwords. For a better effect of classification, we decided to delete these irrelevant content as follows.

Step1: we clean up the texts. This part of processing includes the removal of URLs (which start either with "http://" or "https://") and tags ("@user") and irrelevant expressions (the word is not supported by ANSI coding). We use regular expressions to match strings and replace strings to eliminate irrelevant characters. This part of the work is done because the information is not helpful for the subsequent work, and may even have a negative impact on the classification effect.

Step2: We remove stopwords. This part of the work is mainly to prepare for the extraction of the dictionary. The data contains too many words like "i", "is", "what", etc., these types of words appear frequently in the text, but they have little meaning and may even interfere with the classification effect, we decided to delete the stopwords. The NLTK module[*] contains a list of stopwords. We use this list to complete the work of removing stopwords.

Step3: We extract the stem of the word. First, we lowercase all the words and use the Poster–Stemmer[32] tool for stemming. The main reason for this part is that most of these forms of words express the same meaning. For example: "change," "changing," and "changed." These words all mean to change.

Figure 1 is an example of data preprocessing.

3.2 | Feature extraction

After preprocessing, we consider feature extraction on the data. Three main sets of features are extracted which we qualify as "Sentiment-based feature," "Number-based feature," and "Readability-based feature." By combining these sets, we believe it is possible to detect hate speech and offensive language: "Sentiment-based feature" allows us to extract the polarity of the text, a very essential component of hate speech and offensive language. "Number-based feature" allows us to find any emphasized expression. "Readability-based feature" allows us to understand the quality of each piece of text. In the rest of this subsection, we describe how these features are extracted.

Step 1: Sentiment-based feature. Although the detection of hate speech and offensive language differs from that of sentiment analysis and polarity detection, it still makes a positive meaning to use sentiment-based features as the feature. For hate speech and offensive language, we can determine that it is an emotion and a negative emotion. Therefore, we use the *vaderSentiment*[†] tool to score the sentiment of each piece of text, which is helpful for better classification.

Step 2: Number-based feature. We counted the number of URLs, mentions, and hashtags which are replaced in preprocessing.

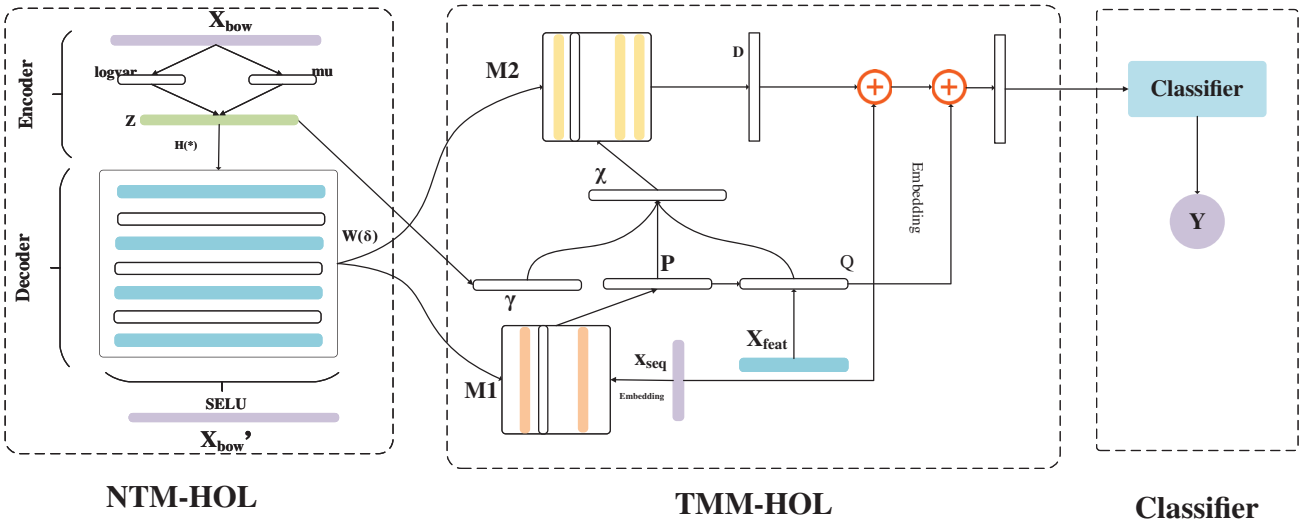

FIGURE 2 TM-HOL (Topic Memory model for detection of Hate speech and Offensive Language): The overall framework of our TM-HOL. The dotted boxes from left to right show the NTM-HOL, TMM-HOL, and the classifier. Here the classifier allows multiple option

Step 3: Readability-based feature. To capture the quality of each piece of data, we improve and calculate the Flesch–Kincaid grade level and the Flesch Reading Ease score[‡]. At the same time, we also count the number of characters, words and syllables in each data.

After this step of processing, we have generated the feature matrix, which we denote as x_{feat}. The matrix is used to feature learning in Section 3.3.2.

3.3 | TM-HOL

In this section, we have proposed the topic memory model for the detection of hate speech and offensive language (TM-HOL). TM-HOL jointly explores topic words extract, detection of hate speech, and memory networks in an end-to-end manner. Our overall architecture of TM-HOL is shown in Figure 2. TM-HOL mainly includes three parts:

(1) Neural topic model for hate speech and offensive language detection (abbreviated as NTM-HOL): It is used to extract potential topics;
(2) Topic memory mechanism for hate speech and offensive language detection (abbreviated as TMM-HOL): It corresponded to the topic words and the text and learned jointly with the extracted features;
(3) Classifier: It is used to distinguish hate speech and offensive language. The classifier allows multiple options, for example, CNN and LSTM.

Given $X = \{x_1, x_2, \ldots, x_n\}$ as the input with N data instances, each instance x is processed into three forms: bag-of-words term vector $x_{\text{bow}} \subset R^{V_b}$, word index sequence vector $x_{\text{seq}} \subset R^{L_{\text{len}}}$, and hate feature vector $x_{\text{feat}} \subset R^{L_{\text{len}}}$. Here, L_{len} is the sequence length and V_b is the size of the vocabulary. First, x_{bow} is used to be input into the neural topic model to induce latent topics z, $z \subset R^P$ where P denotes the number of topics. Then, we use the learned topics to match the topics and texts through two memory matrices M1 and M2 in the TMM-HOL. Finally, it is input into the classifier for predicting the classification label.

3.3.1 | NTM-HOL

The architecture of neural topic model for hate speech and offensive language detection (abbreviated as NTM-HOL) is mainly divided into two parts: encoder and decoder.

NTM-HOL takes x_{bow} as the input of the encoder. The encoder is used to compress the x_{bow}. The decoder takes the output of the encoder and tries to recreate the x_{bow}, what is named $x_{\text{bow}'}$. These compressed data can reflect the important basic characteristics of the data, which is named z, and $z$is the latent topic vector. The detailed workflow is following:

The encoder takes x_{bow} as the input. We assume that each text x has a P-dimensional topic distribution. Topic distribution is named doc-topic. Each topic p is represented by a word distribution over the vocabulary. Word distribution is named topic-word γ_p. Figure 3 shows an example of the

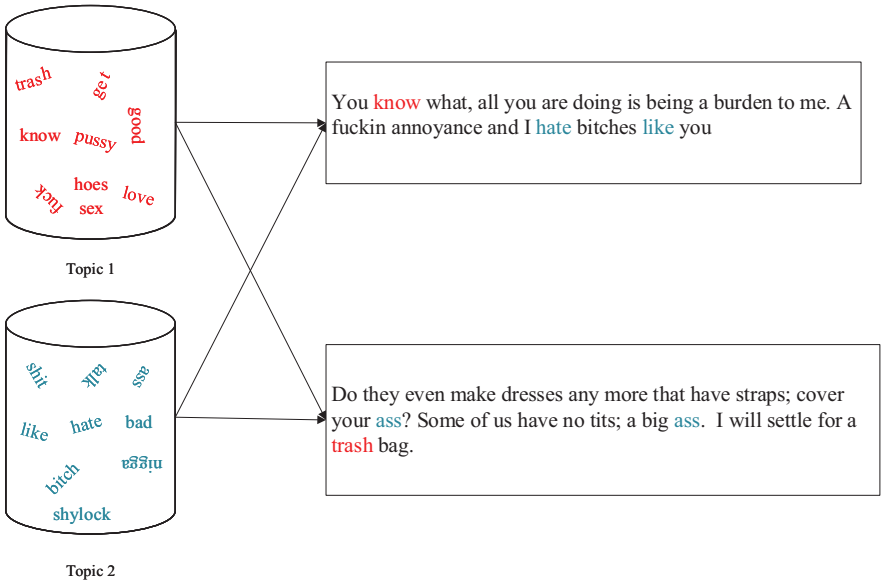

FIGURE 3 NTM-HOL (neural topic model for hate speech and offensive language detection): An example of the contrast relationship between the topic words and the text.

contrast relationship between the topic words and the text. We obtained the random variable x_{bow} by sampling the vector *mu* and vector *logvar* as the mean and variance, where $H(*)$ is a neural perceptron that linearly transforms inputs, activated by a nonlinear transformation. Then we get the p-dimensional latent topic variable z after n sampling, that is, the potential topic z is sampled from Equation (1).

$$z|x_{\text{bow}} \sim N\left(\mu_{z|x_{\text{bov}}}, \sum z|x_{\text{bow}}\right). \tag{1}$$

$$\text{mu} = H\left(x_p\right). \tag{2}$$

$$\log \text{var} = H\left(x_p\right). \tag{3}$$

Then we use this result as the output of the encoder. The probability of the encoder's hidden state is formed in the process of normalization using the SELU function.

3.3.2 | TMM-HOL

Topic memory mechanism for hate speech and offensive language detection (abbreviated as TMM-HOL) is proposed to make the model fit the learned features better. It can map the potential topics learned in the NTM-HOL to classification features. The model learns text features while learning topics.

Two memory matrices, M1 and M2, are produced by two RELU-activated neural perceptions. We assume that γ_p is a p-dimensional topic words distribution of text x_p. We use the formula (4) to calculate the weight matrix $W(\delta)$ of topic words:

$$W(\delta) = \text{RELU}(H(\gamma)). \tag{4}$$

We use $H(\gamma)$ to compute the word distribution given γ. Both memories take the topic words weight matrix $W(\delta)$ as input. We use $\mathfrak{R}$ to represent the embedded, then we use formula (5) to calculate the matching degree between the mth topic and the embedding of the nth word in x_{Seq}:

$$P_{m,n} = \text{RELU}\left(W(\delta)^s[M1\&\mathfrak{R}] + b^s\right), \tag{5}$$

where $[M1\&\mathfrak{R}]$ represents the combination of $M1$ and $\mathfrak{R}$. $W(\delta)$ and b are the parameters to be learned. Here, we use cascading operation. First, we work out cascading operation for x_{seq} and x_{bow}. After this processing, we learn about the features of hate speech and offensive language.

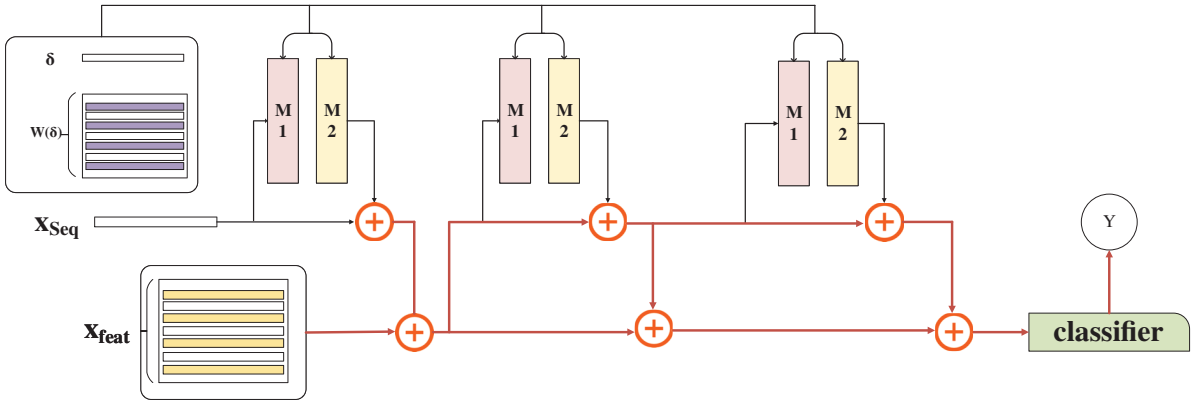

FIGURE 4 TMM-HOL: Topic memory mechanism for hate speech and offensive language detection with three hops

We use x_{feat} to calculate the matching degree between the $P_{m,n}$ and the corresponding feature, where x_{feat} is the feature matrix after feature extraction in the Section 3.2:

$$Q_{P,f} = \text{RELU}\left(P_{m,n} \& \left(x_{\text{feat}} + d^{s}\right)\right). \tag{6}$$

To further combine the text-topic mixture γ, P, and Q, we define the integrated memory weights as:

$$\chi_p = \gamma_p + \tau\left(\sum P + \sum Q\right), \tag{7}$$

where τ is a hyperparameter. Then, by calculating the memory matrix M2 and χ, we obtain the output representation D of the topic memory mechanism:

$$D_p = \chi_p \bullet M2_p. \tag{8}$$

Then it is sent to the classifier for classification prediction. Its overall architecture of TMM-HOL is shown in Figure 4.

3.3.3 | Classifier

Since the classifier allows multiple options, we have proposed two different classifiers (CNN and CLSTM). The models are called TM-HOL-CNN[§] and TM-HOL-CLSTM[¶].

In the CNN model (TM-HOL-CNN), we use two-dimensional convolution as the convolution layer. The pooling method is max pooling where the maximum element in the pooling window is selected. To feed the pooled output from stacked featured maps to the next layer, the maps are flattened into one column. The final layers in a CNN are typically fully connected.

The CLSTM model (TM-HOL-CLSTM) first comes to local perception through the convolutional layer, which is built on top of the pooling layer. The optimal value of the feature matrix is extracted through the maximum pooling layer operation, and the optimal value usually represents the most significant information in the feature set.

4 | EXPERIMENTS

In this section, we firstly introduce the datasets (Section 4.1). Then, we present our experimental results of different methods. We compare our TM-HOL-CNN and TM-HOL-CLSTM methods with 14 current methods to make further analysis (Section 4.2). Besides, we list the evaluation metrics (Section 4.4), construction of environment (Section 4.5), and we discuss the results in detail (Section 4.6). Finally, we list the parameter in our experiment (Section 4.7). The experiments and the objective analysis show the effectiveness of our TM-HOL-CNN and TM-HOL-CLSTM methods to detect hate speech and offensive language (Section 4.7). As shown in Sections 4.6 and 4.7, TM-HOL is effective.

TABLE 1 Statistics of the experimental datasets. Texts to the count of texts in datasets. Class to class labels

Dataset	Texts	Class(%)
Twitter	24,783	Hate (6%); Offensive (77%); Neither (17%)
Supermacist	10,776	noHS (88%); HS (11%); skip (1%)
Misogyny	2285	misogynistic (45%); non-misogynistic (55%)

4.1 | Datasets

The dataset consists of a total of 37,844 textual records which are equally included many aspects of hate speech and offensive language. We selected three datasets with different sizes to test the performance of our models. These datasets also cover different aspects of hate speech, such as color discrimination and gender discrimination. This allows us to test our model more comprehensively. The dataset is created by collecting sentences from three different sources. Table 1 shows the details of the dataset.

- Twitter[2#]: The dataset is provided with 24,783 tweets, randomly sampled from 85.4 M tweets extracted from 33,458 Twitter users, using hate speech lexicon, and it is annotated in a crowdsourcing way by at least three workers using CrowdFlower as "Hate," "Offensive," and "Neither." "Hate" means the text is hate speech. "Offensive" means the text is offensive language, and "Neither" means the text is neither hate speech nor offensive language. The average percentage agreement among annotators on the four labels is 92%.
- Supremacist[33‖]: The dataset is provided 10,568 English sentences that are extracted from the white supremacist Forum. They annotated by three experts as "HS/ noHS"; the labels "skip" and "relation" are also used. "HS" is the abbreviation of "Hate Speech," which means the text is a hate speech. "noHS" is the abbreviation of "not Hate Speech," which means the text is not a hate speech. At the same time, "skip" means the text is not in English, and "relation" means consecutive sentences with this label convey hate speech. The average percentage agreement among annotators on the four labels is 90.97%. Due to each text being isolated on our model, we modified the dataset into three categories: "HS," "noHS," and "skip."
- Misogyny[34**]: The dataset is provided with 2285 sentences posted on the Urban Dictionary platform from 1999 to May 2016. The data were annotated as "misogynistic" and "non-misogynistic" by three independent researchers with domain knowledge. "Misogynistic" means the text expresses obvious hate to women. "non-misogynistic" did not express hate of women.

4.2 | Comparison of methods

We compared TM-HOL with 14 methods to detect hate speech and offensive language. Text categorization can be broadly divided into four categories: machine learning methods; ensemble learning; deep learning and pretrained language methods. We conducted a series of experiments with the three categories. The detail description of the 14 methods is as follows.

4.2.1 | Ensemble learning methods

Ensemble learning is widely used in classification tasks.[35] We choose two traditional and representative models to detect hate speech and offensive language for comparison: Bootstrap aggregating and Boost.

- Bootstrap aggregating: Bootstrap aggregating (abbreviated as Bagging) is a method to improve classification and regression models in terms of stability and accuracy.[36] We used the K-Nearest Neighbor (KNN) as the weak classifier, extracted the TF-IDF feature. Finally, we set the following parameters: max_samples = 5, n_neighbors = 5, weights = 'uniform,' algorithm = 'auto,' and leaf_size = 30.
- Boost: Boost-based approaches achieve good performance in several classification tasks. We use gradient boosting DT as the base classifier, extracted the TF-IDF feature, and set n_estimators = 50. When data is extracted, its weights are updated, and the weight of the dataset that was incorrectly divided last time is increased. Finally, 50 base classifiers are obtained.

4.2.2 | Machine learning methods

Machine learning has achieved outstanding achievements in text classification in recent years.[35] We choose some traditional and representative models to detect hate speech and offensive language for comparison. These models are DT, KNN, Multinomial NB, and RF.

- Decision Tree: DT builds classification or regression models in the form of a tree structure.[37,38] This model is trained using information entropy, extracted the TF-IDF feature and set the following parameters: random_state = 100, max_depth = 3, min_samples_leaf = 5.
- K-Nearest Neighbor: KNN is a nonparametric method widely used for text classification. We used Minkowski distance as the distance measurement and set the following parameters: n_neighbors = 5, weights = "uniform," algorithm = "auto," p = 2, and leaf_size = 30.
- Multinomial NB: Multinomial NB is based on NB. We set the following parameters: alpha = 1.0, class_prior = None, fit_prior = True.
- Random Forest: RF is an extended variant of the bagging model. We set the following parameters: n_estimators = "warn," criterion = "gini," min_samples_split = 2, min_samples_leaf = 1, max_features = "auto," and n_estimators = 100.

4.2.3 | Deep learning methods

Recently deep learning methods have started to be applied to text classification.[39-41] We choose three simple and representative models to detect hate speech and offensive language for comparison: Deep neural networks (DNNs), Recurrent convolutional neural networks (RCNN), and Convolutional neural networks (CNN).

- Deep neural networks: A general DNN was proposed for natural language processing.[40] This method uses the weighted average of the word embeddings and subsequently applies a softmax layer. The weight for each word is its TF-IDF value.
- Recurrent convolutional neural networks: We also select a RCNN[42] for comparison. With the pretrained word embeddings, we use a recurrent structure, which is a bidirectional recurrent neural network, to capture the contexts. Finally, the softmax function is applied on the output layer.
- Convolutional neural networks: We select a CNN[41] for comparison.
- CLSTM-based topic memory network: A CLSTM-based topic memory network (abbreviated as CLSTM-TMN) was proposed for marketing intention detection.[20] We use this model for hate speech and offensive language detection.

4.2.4 | Pretrained language methods

Recently, pretrained language models have achieved remarkable success, presenting new state-of-the-art results in NLP. In this work, we chose the following four models for comparison. This pretrained model uses the bert-base-uncased English pretrained model.

- Bidirectional Encoder Representations from Transformers (abbreviated as BERT):[24] BERT is an NLP model that was designed to pretrained deep bidirectional representations from unlabeled text and, after that, is fine-tuned using labeled text for different NLP tasks. Bert-base model[††] contains an encoder with 12 Transformer blocks, 12 self-attention heads, and the hidden units size of 768.
- BertATT: BertLSTM used BERT and a two-layer Attention. BertATT model can be divided into the following two parts: First obtain the semantic representation of each text through BERT model training; then input the vector representation of each word in the text into the Attention model.
- BertCNN: BertCNN uses the words generated by the BERT pretrained model for embedding, then uses CNN for pooling operation, and finally connects to the dense layer for classification.
- BertLSTM: The BertLSTM fusion model uses the BERT model for word vector training, and combines the LSTM model for text classification, and finally uses the Softmax function for text classification.

4.3 | Parameter setup

In the experiment, we used the pretrained GloVe for word embedding. GloVe Word Embeddings[‡‡] has been trained on a large tweet corpus (2B tweets, 27B tokens, 1.2M vocab, uncased). Other specific parameters are set as follows:

- TOPIC_NUM = 2;
- HIDDEN_NUM = [500,500];

- TOPIC_EMB_ DIM = 150;
- BATCH_SIZE = 32;
- TOP_P = 10.

TOPIC_NUM is the number of extracted topics, HIDDEN_NUM is the size of the hidden layer, TOPIC_EMB_DIM is the memory size of the topic. BATCH_SIZE is the number of data samples captured in one training session. Top_P represents the top K words in the vocabulary distribution. In the learning process, we use the Early Stopping strategy when we train the network. The maximum value of the epoch is 800.

4.4 | Evaluation metrics

To evaluate the performance of classification, we use four different key performances indicators (KPIs):

(1) weighted-F1 score (Weighted-F1);
(2) Accuracy;
(3) macro-average precision (macro-Pre);
(4) macro-average recall (macro-Re).

Due to the imbalance in the amount of data in each category in the dataset, we use weighted-F1 to calculate the F1 score. A macro-average will compute the metric independently for each class and then take the average (hence treating all classes equally), whereas a micro-average will aggregate the contributions of all classes to compute the average metric. Since we pay more attention to categories with a small sample size, we choose to use micro-averaging. The detail descriptions are given below:

- Weighted-F1: The Weighted-F1 score is the harmonic mean of precision and recall at a decision threshold of 0.50. The support of each category is used as the weight to calculate the average value of F1 scores by category. The calculation of Weighted-F1 is divided into two steps:
  (a) Calculate F1 for each category:

$$F1_{c_i} = 2\frac{\text{Recall}_{c_i} * \text{Precision}_{c_i}}{\text{Recall}_{c_i} + \text{Precision}_{c_i}}. \tag{9}$$

  (b) Calculate the value of Weighted-F1:

$$\text{Weighted -}\mathbf{F1} = \sum_{i=1}^{C} F1_{c_i} * W_{c_i}. \tag{10}$$

- macro-Pre: macro-Pre indicates the proportion of samples that are really positive among the samples predicted to be positive. The calculation formula is:

$$\text{macro-}\mathbf{Pre} = \frac{\overline{\text{TP}}}{\overline{\text{TP}} * \overline{\text{FP}}}. \tag{11}$$

- Accuracy: Accuracy is the proportion of correctly classified samples to the total number of samples. The calculation formula for accuracy is:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{12}$$

- macor-Re: macor-Re indicates the proportion of all samples that are truly positive. The formula for calculating the recall is:

$$\text{macro-}\mathbf{Re} = \frac{\overline{\text{TP}}}{\overline{\text{TP}} * \overline{\text{FN}}}. \tag{13}$$

4.5 | Construction of environment

The experiment is based on the Python language. The specific parameters of the experimental environment are as Tables 2 and 3:

TABLE 2 Environment configuration of Topic memory model for detection of hate speech and offensive language

Hardware and software	Configure
CPU	Intel(R)Core(TM) i7-8750H CPU @ 2.20GHz
RAM	16GB
Operating System	Windows 10
Environment	Anaconda 3.6 Tensorflow=1.12.0 Keras =2.2.4 Gensim=3.5.0

TABLE 3 Environment configuration of comparable experiments

Hardware and software	Configure
CPU	Intel(R)Core(TM) i7-8750H CPU @ 2.20GHz
RAM	16GB
Operating System	Windows 10
Environment	Anaconda 3.7 Sklearn=0.21.3

4.6 | Experiment results

This section explains the results of all proposed methods in detail. In our paper, we carried out three sets of experiments.

4.6.1 | Twitter experiment

To evaluate different methods, We first evaluated our preliminary models using the Twitter dataset. Table 4 presents the results obtained using different classification methods.

Regarding the model families, TM-HOL-CLSTM outperforms the other models and achieved a Weighted-F1 of 0.9042 on the Twitter dataset. TM-HOL-CLSTM is closely followed by the TM-HOL-CNN. This shows that the ability of the TM-HOL model is better in the Twitter dataset. Table 4 shows that in general, the majority of models of macro-Pre and macro-Re got low scores. On the contrary, our model is better.

As shown in Table 4, when deep learning methods (CNN, RCNN, and DNN) are compared with machine learning methods (Boost and Bagging), the results show that the neural network approach is superior to the traditional machine approach.

For classical models such as DNN and RF, they were able to achieve performance accuracy for predicting hate speech and offensive language too. The reason is due to the stacking of the Word2Vec word embedding model as a feature vector that keeps the semantics of hate speech and offensive language. To further analyze the experimental effect, we looked at the classification performance for a specific category. We discovered a phenomenon. Because of the extremely uneven distribution of various categories in this dataset, this issue caused some models to fish in troubled waters, Such as Boost and DT. It is more accurate in detecting offensive language, but when it detects hate speech, the F1-score of Boost is only 0.23 and the F1-score of DT is only 0.06.

For the CLSTM-TMN model, the evaluation metrics show that our model successfully solves the loss of the overall characteristics. For the pretrained model, our model performs better than BertLSTM, and our model has achieved similar results with the pre-trained model on this dataset.

4.6.2 | Supremacist experiment

Table 5 presents the results obtained using different classification algorithms what is used Supremacist dataset. TM-HOL-CNN outperforms the other models and achieved a Weighted-F1 of 0.8852 on the Twitter dataset. TM-HOL-CNN is closely followed by the TM-HOL-CLSTM. The effects of DNN and RCNN perform well too when the model used the Supremacist dataset. However, the indexes of macro-Pre and macro-Re are lower. But compared to the Twitter dataset, the scores of macro-Pre and macro-Re have improved. This shows that most models are more suitable for small datasets.

For the traditional classical machine learning method, such as Bagging and DT, the result of it has been improved(the score of Weighted-F1 is improved by 0.1191 and 0.1774, respectively). But it is still not effective when the model is used to detect hate speech(the F1-score of Boost is only 0.19).

TABLE 4 Comparisons of Weighted-F1, Accuracy, macro-Pre and macro-Re on Twitter datasets

	Twitter			
Model	**Weighted-F1**	**Accuracy**	**macro-Pre**	**macro-Re**
Bagging	0.6949	0.7817	0.5320	0.3502
Boost	0.8365	0.8563	0.5648	0.7383
Decision Tree	0.6686	0.7690	0.2563	0.3333
KNN	0.7839	0.8166	0.4915	0.6908
Multinomial NB	0.7045	0.7849	0.3641	0.5820
Random Forest	0.8298	0.8563	0.5366	0.7760
DNN	0.8594	0.8648	0.6475	0.6785
CNN	0.8219	0.8069	0.5955	0.4888
RCNN	0.8057	0.8075	0.5601	0.5803
CLSTM-TMN	0.8806	0.8818	0.6933	0.6975
BERT	0.9016	0.9107	0.7853	0.7089
BertLSTM	0.8835	0.9052	0.7251	0.6438
BertATT	0.9047	0.9097	0.7747	0.7347
BertCNN	0.9034	0.9097	0.7950	0.7283
Our method				
TM-HOL-CNN	**0.9009**	**0.9179**	**0.6694**	**0.8071**
TM-HOL-CLSTM	**0.9042**	**0.9036**	**0.7718**	**0.7597**

Abbreviations: CNN, convolutional neural network; DNN, deep neural networks; KNN, K-Nearest Neighbor; NB, Naive Bayes; RCNN, recurrent convolutional neural networks.

Among all the comparison models, CNN has the most outstanding performance. The Weighted-F1 of CNN is 0.8981, this score is 0.0129 higher than TM-HOL-CNN. But its macro-Pre and macro-Re scores are low (macro-Pre is 0.5404 and macro-Re is 0.6942), our model (macro-Pre is 0.6278 and macro-Re is 0.8435)is better than CNN. Our model is better than CLSTM-TMN. For pretrained models, our model is better than BertLSTM but lower than other pretrained models (The difference between the models is about 0.02 or 0.04). This shows that pretrained on a large text corpus can learn common language representations and help complete subsequent tasks. Pretrained provides better model initialization, which usually leads to better generalization performance and accelerates the convergence of the target task. Therefore, when applied to a smaller dataset, the pretrained model performs better.

4.6.3 | Misogyny experiment

The above two experiments prove the effectiveness of our model on multiclass classification. To further prove the results of our model on the binary classification, we conducted a third experiment. Misogyny is a dataset labeled as two categories.

Table 6 presents the results obtained using different classification methods that are used Misogyny dataset. According to Table 6, we can conclude that TM-HOL-CLSTM outperforms the other models and achieved a Weighted-F1 of 0.8817 on the Twitter dataset. TM-HOL-CLSTM is closely followed by the TM-HOL-CNN. TM-HOL-CLSTM achieved a Weighted-F1 of 0.8817 on the Misogyny set. Most algorithms have achieved good results when they are using the Misogyny dataset. All indicators have been improved. But our model performs well on all evaluation metrics. This shows that the ability of the TM-HOL model is better on binary classification.

To further analyze the experimental effect, we looked at the classification performance for a specific category. We conclude that when these models are used for binary classification, the classification effect is relatively stable and the accuracy, compared to multiclassification. But in terms of overall performance, our model performs better.

Combining the experimental results of the above three datasets, we can conclude. Bagging and Boost are more suitable for binary classification, although they have higher F1 scores in multiclass classification. When this type of model is applied to real life, it will easily lead to a higher false alarm rate.

TABLE 5 Comparisons of Weighted-F1, Accuracy, macro-Pre, and macro-Re on Supremacist datasets

	Supremacist			
Model	**Weighted-F1**	**Accuracy**	**macro-Pre**	**macro-Re**
Bagging	0.8310	0.8845	0.3569	0.4616
Boost	0.8502	0.8835	0.3901	0.5268
Decision Tree	0.8386	0.8807	0.5879	0.3704
KNN	0.8310	0.8845	0.3569	0.4616
Multinomial NB	0.8303	0.8845	0.3333	0.2948
Random Forest	0.8350	0.8854	0.3622	0.6843
DNN	0.8596	0.8650	0.4752	0.5717
CNN	0.8981	0.8844	0.5404	0.6942
RCNN	0.8648	0.8654	0.4592	0.4479
CLSTM-TMN	0.8562	0.8755	0.5061	0.4286
BERT	0.9222	0.9239	0.6775	0.7226
BertLSTM	0.8681	0.8998	0.8969	0.4471
BertATT	0.9024	0.9128	0.7883	0.6578
BertCNN	0.9200	0.9267	0.8977	0.6933
Our method				
TM-HOL-CNN	**0.8852**	**0.8939**	**0.6278**	**0.8435**
TM-HOL-CLSTM	**0.8817**	**0.8908**	**0.6502**	**0.7764**

Abbreviations: CNN, convolutional neural network; DNN, deep neural networks; KNN, K-Nearest Neighbor; NB, Naive Bayes; RCNN, recurrent convolutional neural networks.

When our model is applied to the dataset, the experimental effect is still better than CLSTM-TMN. The effectiveness of our model is proved on these three datasets. Misogyny is a smaller dataset. The experimental effect of our model is still lower than that of the pretrained model. This is because our model lacks a lot of prior knowledge.

4.7 | Parameter sensitivity

To better analyze the results of our experiments, we fixed other parameters to view the influence of a certain parameter on the experiment. The parameters to be optimized are as follows:

- TOPIC_NUM;
- Top_P;
- HIDDEN _ NUM;
- TOPIC _ EMB_ DIM;
- BATCH_SIZE;

To tune these parameters, each time we fix all the parameters except one and look for their optimal value. Therefore, to determine the best value of the parameter TOPIC_NUM and Top_P, we set the values of the TOPIC_NUM as 1, 2, 3, 4, and 5, when Top_P is 10. At the same time, we set the values of the Top_P as 5, 10, 15, and 20, when TOPIC_NUM is 2. To facilitate the comparison of the experiment, we selected the first 50 items in the experimental data record for comparison.

4.7.1 | Effects of the TOPIC−NUM

The first indicator we consider is the TOPIC_NUM. We try different values of the parameter TOPIC_NUM. The results are given in Figures 5–7. Figure 5 shows the classification accuracy changes with TOPIC−NUM when the model used Twitter as the dataset. Figure 6 shows the classification

TABLE 6 Comparisons of Weighted-F1, Accuracy, macro-Pre and macro-Re on Misogyny datasets

	Misogyny			
Model	**Weighted-F1**	**Accuracy**	**macro-Pre**	**macro-Re**
Bagging	0.7877	0.7877	0.7867	0.7869
Boost	0.8465	0.8490	0.8421	0.8655
Decision tree	0.8227	0.8227	0.8221	0.8220
KNN	0.7845	0.7855	0.7819	0.7873
Multinomial NB	0.7951	0.8008	0.7912	0.8278
Random Forest	0.8557	0.8577	0.8514	0.8720
DNN	0.8598	0.8599	0.8585	0.8600
CNN	0.8846	0.8493	0.5878	0.5444
RCNN	0.6696	0.6827	0.6963	0.7338
CLSTM-TMN	0.8485	0.8513	0.8663	0.8390
BERT	0.8938	0.8947	0.9044	0.8925
BertLSTM	0.8681	0.8684	0.8706	0.8674
BertATT	0.8859	0.8860	0.8858	0.8858
BertCNN	0.9034	0.9035	0.9037	0.9032
Our Method				
TM-HOL-CNN	**0.8788**	**0.8792**	**0.8733**	**0.8785**
TM-HOL-CLSTM	**0.8817**	**0.8820**	**0.8760**	**0.8807**

Abbreviations: CNN, convolutional neural network; DNN, deep neural networks; KNN, K-Nearest Neighbor; NB, Naive Bayes; RCNN, recurrent convolutional neural networks.

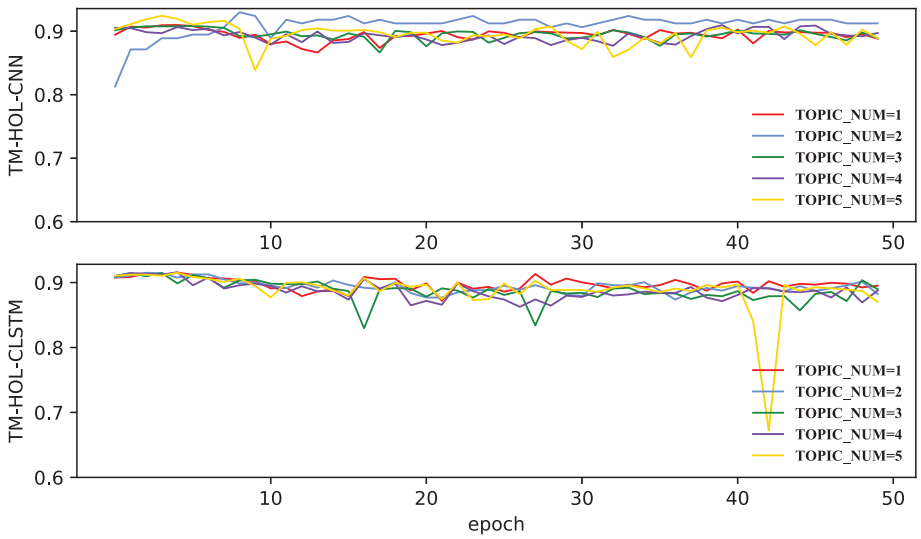

FIGURE 5 Classification accuracy for different values of the parameter TOPIC_NUM when the experiment used Twitter dataset

accuracy changes with TOPIC−NUM when the model used Supremacist as the dataset. Figure 7 shows the classification accuracy changes with TOPIC−NUM when the model used Misogyny as the dataset.

As is shown in the picture, we can conclude, that is TM-HOL-CLSTM is more stable than TM-HOL-CNN when the experiment used the same dataset. In general, the effect is the worst and the experiment results are unstable when the model is TM-HOL-CNN and the TOPIC_NUM is 1. When using model TM-HOL-CLSTM and setting TOPIC_NUM to 5, the same effect.

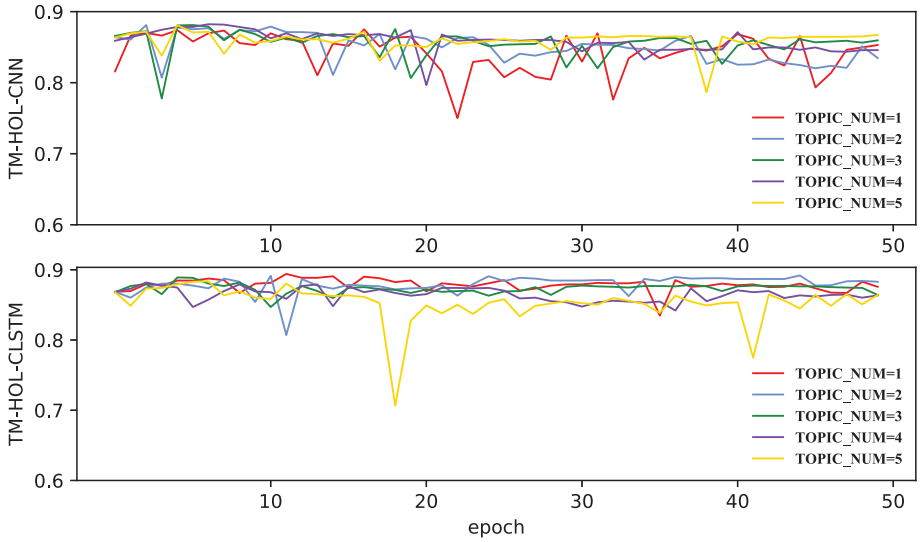

FIGURE 6 Classification accuracy for different values of the parameter TOPIC_NUM when the experiment used Supremacist dataset

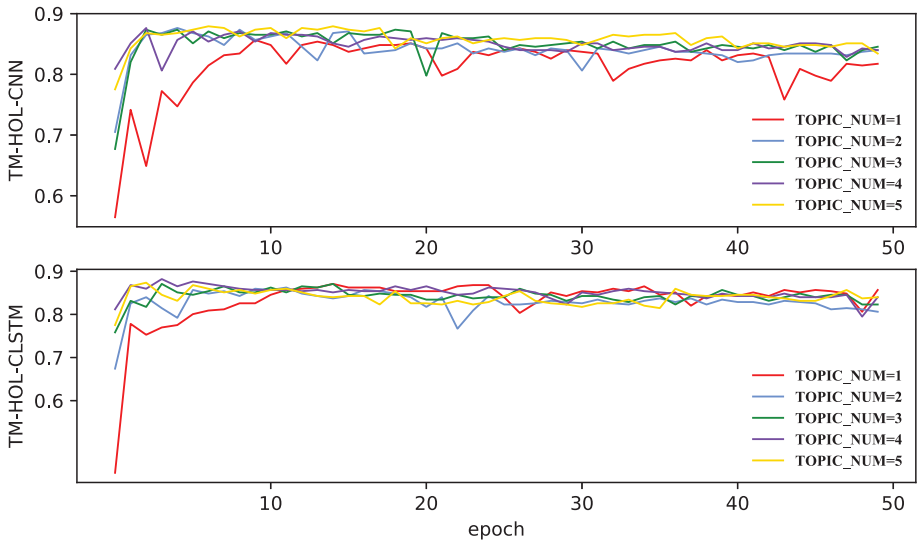

FIGURE 7 Classification accuracy for different values of the parameter TOPIC_NUM when the experiment used Misogyny dataset

TABLE 7 Topic words extracted when the model used Misogyny as the dataset and TOPIC_NUM is 5

know	think	good	get	like	life	go	really	even	time
learned	sluts	costs	kids	owned	gansta	houses	live	jocks	stupid
people	one	make	like	someone	want	ever	get	girl	anyone
person	sex	girl	sexual	female	man	one	like	woman	face
vagina	female	act	male	woman	penis	sex	man	sexual	term

We observe the topic words what is extracted under this indicator. When the TOPIC_NUM is 5, the topic words are shown below:

According to Table 7, we find that when the value of TOPIC_NUM is too large, some of the extracted topic words are not representative, such as "one," "make," and "want." At the same time, the classification effect is also unstable. Therefore, reasonable control of the size of TOPIC_NUM has a huge impact on the classification effect.

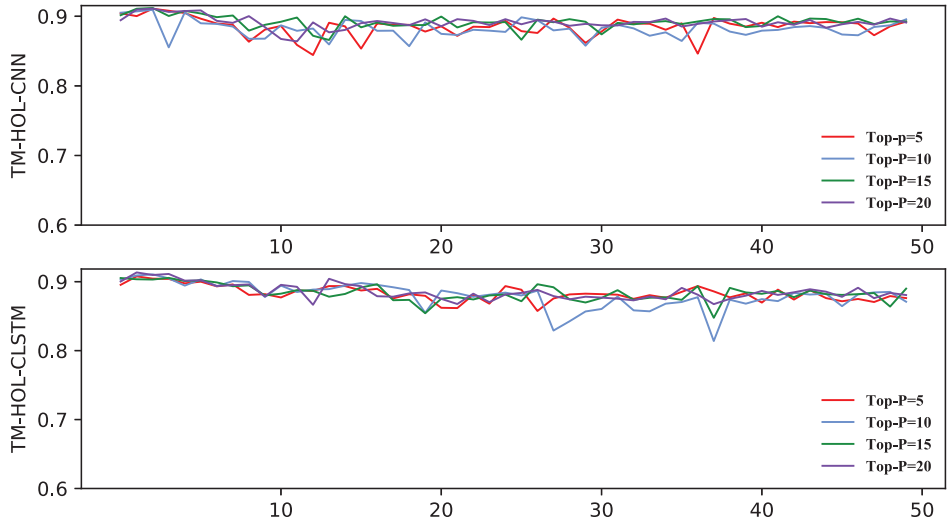

FIGURE 8 Classification accuracy for different values of the parameter Top_P, when the experiment used Twitter dataset and TOPIC_NUM is 2

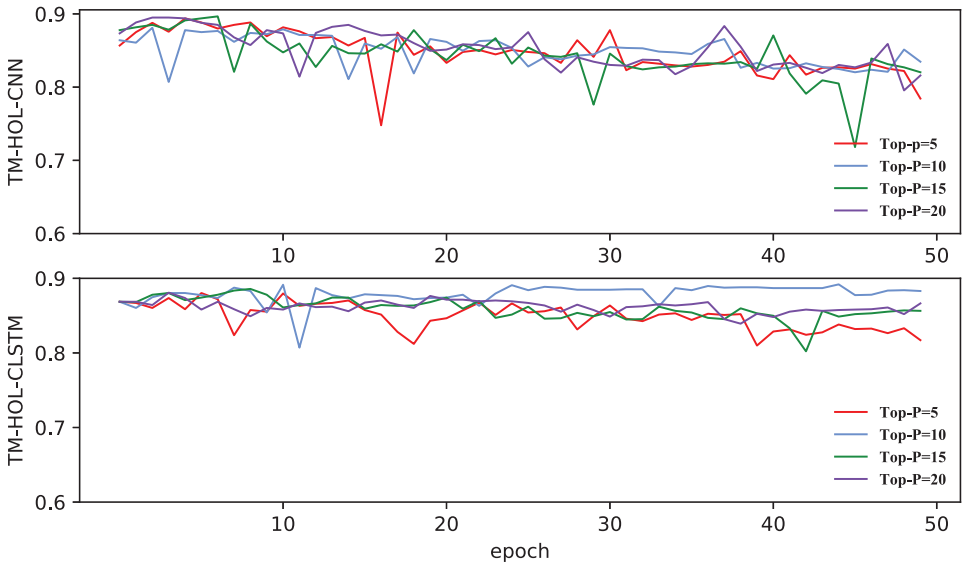

FIGURE 9 Classification accuracy for different values of the parameter Top_P, when the experiment used Supremacist dataset and TOPIC_NUM is 2

4.7.2 | Effects of the Top_P

The results are given in Figures 8–10. Figure 8 shows the classification accuracy changes with Top_P when the model used Twitter as the dataset. Figure 9 shows the classification accuracy changes with Top_P when the model used Supremacist as the dataset. Figure 10 shows the classification accuracy changes with Top_P when the model used Misogyny as the dataset. The result shows that the addition or reduction of the number of Top_P can affect the stability of the model.

When Top_P is 1, the classification effect is the most unstable and the effect is not good. At the same time, When Top_P is 2, the classification effect is the most stable. And TM-HOL-CLSTM is better than TM-HOL-CLSTM using the same dataset. At the same time, we found when the top is 1, TM-HOL-CNN performs the worst, and when the top is 5, TM-HOL-CLSTM performs the worst. This phenomenon is more obvious when using Misogyny as the dataset. The reason for this phenomenon is TM-HOL-CLSTM has the feature of storing historical information. TM-HOL-CLSTM is used to obtain context information semantics.

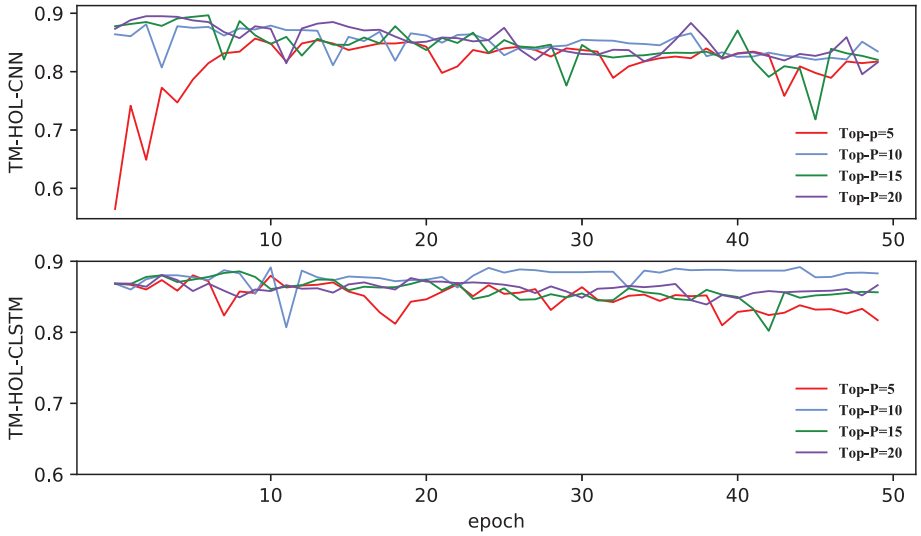

FIGURE 10 Classification accuracy for different values of the parameter Top_P, when the experiment used Misogyny dataset and TOPIC_NUM is 2

5 | CONCLUSIONS AND FUTURE WORK

In this work, we proposed a topic memory model to detect hate speech and offensive language. Our proposed approach automatically detects hate speech and offensive language by extracting topic words and learning features to classify tweets into hate, offensive and clean. TM-HOL reaches an accuracy equal to 90.42% for the multiclass classification using the Twitter dataset, an accuracy equal to 88.52% for the multiclass classification using the Supremacist dataset, and accuracy equal to 88.17% for the binary classification using Misogyny dataset.

Due to people's tend to use code-mixed language, such as Hindi–English, the need for code-mixed hate speech and offensive language detection has risen. Our method has shown good results in English hate speech and offensive language detection. In the future, we hope the model will be applied to code-mixed hate speech and offensive language detection.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (61772231), the Shandong Provincial Natural Science Foundation (ZR2017MF025), the Project of Shandong Provincial Social Science Program (18CHLJ39), the Project of Independent Cultivated Innovation Team of Jinan City (2018GXRC002), and the Shandong Provincial Key R&D Program of China (2021CXGC010103).

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ENDNOTE

*https://pythonspot.com/nltk-stop-words/
†https://pypi.org/project/vaderSentiment/
‡https://readable.com/
§The source codes of TM-HOL-CNN are available at: https://github.com/ChenJing-825/TM-HOL-CNN
¶The source codes of TM-HOL-CLSTM are available at: https://github.com/ChenJing-825/TM-HOL-CLSTM
#https://data.world/crowdflower/hate-speech-identification
‖https://github.com/Vicomtech/hate-speech-dataset
**https://data.mendeley.com/datasets/3jfwsdkryy/3
††https://github.com/huggingface/pytorch-pretrained-BERT
‡‡http://nlp.stanford.edu/projects/glove/

ORCID

Kun Ma https://orcid.org/0000-0002-0135-5423

REFERENCES

1. Fernquist J, Lindholm O, Kaati L & Akrami N A study on the feasibility to detect hate speech in Swedish. Proceedings of the 2019 IEEE International Conference on Big Data (Big Data); 2019.

2. Watanabe H, Bouazizi M, Ohtsuki T. Hate speech on twitter a pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access*. 2018;6:1-1.
3. Urman A, Katz S. What they do in the shadows: examining the far-right networks on Telegram. *Inf Commun Soc*. 2020;0:1-20.
4. Elkin-Koren N, Perel M. Separation of functions for AI: restraining speech regulation by online platforms. *Lewis Clark L. Rev*. 2020;24:857.
5. Du J, van Koningsbruggen GM, Kerkhof P. A brief measure of social media self-control failure. *Comput Human Behav*. 2018;84:68-75.
6. Gaydhani A, Doma V, Kendre S, Bhagwat L. Detecting hate speech and offensive language on twitter using machine learning: an N-gram and TFIDF based approach; 2018. arXiv preprint arXiv:1809.08651.
7. Perry J. Ireland in an international comparative context. *Critical Perspectives on Hate Crime*. Springer; 2017:93-107.
8. Gomez R, Gibert J, Gomez L, Karatzas D. Exploring hate speech detection in multimodal publications. Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision; 2020:1470-1478.
9. Sayer P. Line spacing in LaTeX documents. Accessed May 31, 2016. https://www.pcworld.com/article/3076745/facebook-google-microsoft-and-twitter-crack-down-on-online-hate-speech-in-eu.html/
10. Davidson T, Warmsley D, Macy MW, Weber I. Automated hate speech detection and the problem of offensive language. Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017; 2017:512-515.
11. Gambäck B, Sikdar UK. Using convolutional neural networks to classify hate-speech. Proceedings of the 1st Workshop on Abusive Language Online; 2017:85-90.
12. Chen Y, Zhou Y, Zhu S, Xu H. Detecting offensive language in social media to protect adolescent online safety. Proceedings of the 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing; 2012:71-80.
13. Waseem Z, Hovy D. Hateful symbols or hateful people? predictive features for hate speech detection on twitter. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT); 2016:88-93.
14. Al-Makhadmeh Z, Tolba A. Automatic hate speech detection using killer natural language processing optimizing ensemble deep learning approach. *Computing*. 2020;102(2):501-522.
15. Zhang Z. Hate speech detection using a convolution-LSTM based deep neural network. Proceedings of the 2018 World Wide Web Conference; 2018:1-10.
16. Chen Y, Zhang H, Liu R, Ye Z, Lin J. Experimental explorations on short text topic mining between LDA and NMF based Schemes. *Knowl Based Syst*. 2019;163(January 1):1-13.
17. Srivastava A, Sutton C. Autoencoding variational inference for topic models; 2017. arXiv preprint arXiv:1703.01488.
18. Zeng J, Li J, Song Y, Gao C, Lyu MR, King I. Topic memory networks for short text classification; 2018. arXiv preprint arXiv:1809.03664.
19. Wang Y, Li J, Chan HP, King I, Lyu MR, Shi S. Topic-aware neural keyphrase generation for social media language; 2019. arXiv preprint arXiv:1906.03889.
20. Wang Y, Ma K, Garcia-Hernandez L, et al. A CLSTM-TMN for marketing intention detection. *Eng Appl Artif Intell*. 2020;91:103595.
21. Roß B, Rist M, Carbonell G, Cabrera B, Kurowsky N, Wojatzki MM. Measuring the reliability of hate speech annotations: the case of the European refugee crisis. by Michael Beißwenger, Michael Wojatzki and Torsten Zesch (Eds.), *Proceedings of the Originally published in Bochumer Linguistische Arbeitsberichte*. (Vol 17). NLP4CMC III: 3rd Workshop on Natural Language Processing for Computer-Mediated Communication; 2016:6-9.
22. Sreelakshmi K, Premjith B, Soman K. Detection of hate speech text in Hindi-English code-mixed data. *Proc Comput Sci*. 2020;171:737-744.
23. Pereira-Kohatsu JC, Sánchez L, Liberatore F, Camacho-Collados M. Detecting and monitoring hate speech in twitter. *Sensors (Basel, Switzerland)*. 2019;19:4654–4691.
24. Plaza-del-Arco FM, Molina-González MD, Ureña-López LA, Martín-Valdivia MT. Comparing pre-trained language models for Spanish hate speech detection. *Expert Syst Appl*. 2021;166:114120.
25. Kwok I, Wang Y. Locate the hate: detecting tweets against blacks. Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 27, 2013.
26. Alorainy W, Burnap P, Liu H, Williams ML. 'The enemy among us': Detecting cyber hate speech with threats-based othering language embeddings. *ACM Trans Web*. 2019;13(3):1-26.
27. Saleem HM, Dillon KP, Benesch S, Ruths D. A web of hate: tackling hateful speech in online social spaces; 2017. arXiv preprint arXiv:1709.10159.
28. Mossie Z, Wang JH. Vulnerable community identification using hate speech detection on social media. *Inf Process Manag*. 2020;57(3):102087.
29. Salminen J, Hopf M, Chowdhury SA, Jung GS, Almerekhi H, Jansen BJ. Developing an online hate classifier for multiple social media platforms. *Human-centric Comput Inf Sci*. 2020;10(1):1-34.
30. Badjatiya P, Gupta S, Gupta M, Varma V. Deep learning for hate speech detection in tweets. WWW '17 Companion Proceedings of the 26th International Conference on World Wide Web Companion; 2017:759-760.
31. Saeed HH, Calders T & Kamiran F OSACT4 shared tasks: ensembled stacked classification for offensive and hate speech in Arabic tweets. Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection; 2020:71-75.
32. Bird S. NLTK: the natural language toolkit. Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics; 2006:69-72.
33. de Gibert O, Perez N, García-Pablos A & Cuadros M Hate speech dataset from a white supremacy forum. Proceedings of the 2nd Workshop on Abusive Language Online (ALW2); 2018:11-20.
34. Lynn T, Endo PT, Rosati P, Silva I, Santos GL, Ging D. Data set for automatic detection of online misogynistic speech. *Data Brief*. 2019; 26:104223.
35. Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D. Text classification algorithms: a survey. *Information*. 2019;10:4. doi:10.3390/info10040150
36. Breiman L. Random forests. *Mach Learn Arch*. 2001;45(1):5-32.
37. Magerman DM. Statistical decision-tree models for parsing. Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics; 1995:276-283.
38. Quinlan JR. Induction of decision trees. *Mach Learn*. 1986;1(1):81-106.
39. Liu P, Qiu X, Huang X. Recurrent neural network for text classification with multi-task learning; 2016. ArXiv:abs/1605.05101.
40. Collobert R, Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning. *Proceedings of the 25th international conference on Machine learning*; 2008;160–167.

41. Kim Y. Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014:1746-1751.
42. Lai S, Xu L, Liu K, Zhao J. Recurrent convolutional neural networks for text classification. Proceedings of the 29th AAAI Conference on Artificial Intelligence. AAAI'15; 2015:2267-2273.

How to cite this article: Chen J, Ma K, Ji K, Chen Z. TM-HOL: Topic memory model for detection of hate speech and offensive language. *Concurrency Computat Pract Exper*. 2021;e6754. doi: 10.1002/cpe.6754