This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

BiCapsHate: Attention to the Linguistic Context of Hate via Bidirectional Capsules and Hatebase

Ashraf Kamal, Tarique Anwar, Vineet Kumar Sejwal, and Mohd Fazil

***Abstract*— Online social media (OSM) communications sometimes turn into hate-filled and offensive comments or arguments. It not just disrupts the social fabric online, but also leads to hate, violence, and crime, in the real physical world in worst scenarios. The existing content moderation practices of OSM platforms often fail to control the online hate. In this article, we develop a deep learning model called `BiCapsHate` to detect hate speech (HS) in OSM posts. The model consists of five layers of deep neural networks. It starts with an *input* layer to process the input text and follows on to an *embedding* layer to embed the text into a numeric representation. A *BiCaps* layer then learns the sequential and linguistic contextual representations, a *dense* layer prepares the model for final classification, and lastly the *output* layer produces the resulting class as either hate or non-HS (NHS). The *BiCaps* layer, being the most important component, effectively learns the contextual information with respect to different orientations in both forward and backward directions of the input text via capsule networks. It is further aided by our rich set of hand-crafted shallow and deep auxiliary features including the `Hatebase` lexicon, making the model *well-informed*. We conduct extensive experiments on five benchmark datasets to demonstrate the efficacy of the proposed `BiCapsHate` model. In the overall results, we outperform the existing state-of-the-art methods including fBERT, HateBERT, and ToxicBERT. `BiCapsHate` achieves up to 94% and 92% *f-score* on balanced and imbalanced datasets, respectively. Our complete source code is publicly available at GitHub repository https://github.com/Ashraf-Kamal/BiCapsHate.**

***Index Terms*— Capsule networks, hate speech (HS) detection, Hatebase lexicon, long short-term memory (LSTM) networks, online social media (OSM).**

I. Introduction

ONLINE social media (OSM) platforms, such as Facebook and Twitter, allow users to create, foster, and shape relationships, by exchanging ongoing moods, emotions, thoughts, information, and experiences [1], [2], [3], [4]. The exchange of thoughts of users on critical issues sometimes leads to heated arguments [5]. Tweets posted by high-profile politicians and celebrities get disseminated to large masses in a short span of time through their followers, and then receive responses back from the followers in the form of retweets, comments, likes, and shares. It has become a problem of growing concern these days that often the responses are filled with negative, hateful, and abusive comments, especially on crucial and controversial issues. The hate sometimes targets a specific community, race, gender, or socio-political group. This nature of communications further propagates hate that may lead to adverse consequences, such as physical violence, in our real-world societies. For example, in February 2020, lots of hate speech (HS) contents were posted on social media platforms related to Delhi's (India) Shaheen Bagh protest against the national register of citizens (NRC), citizenship amendment act (CAA), and national population register (NPR).[1] The hate particularly targeted one religious community and those supporting the protest. Soon after this, a violent riot took place in Delhi.[2] Similar HS instances keep appearing in different parts of the world in small (individual level) and large (community level) scales. The social media platforms usually have teams for content moderation, following a standard policy to deal with such offensive contents. Sometimes the contents are deleted and the violating users are suspended. However, the moderation in its current form is ineffective in controlling the amount of online hate propagation [6], [7].

The problem of HS detection on social media platforms is attracting attention of researchers from different fields, including data science, artificial intelligence, natural language processing, social science, psychology, and law. The existing literature has explored several supervised learning methods for solving this problem [8], [9], [10], [11]. However, there is still a demand for more effective methods. There is a significant role of context in HS detection [8], [12]. Our communicative language is often ambiguous. A word can be hateful in one context and normal in another. For example, the word "*bitch*," existing in `HateBase` (discussed later in detail), is a fairly normal word when used to mean a "*female dog*," but can be hateful when used in the context of a "*human female*." It is the context that distinguishes between the hateful and nonhateful forms. One major challenge in HS detection is to effectively capture the context of hate. Some HS detection methods rely on handcrafted features to a great extent. It requires a lot of manual engineering and design,

Manuscript received 30 January 2022; revised 4 July 2022 and 23 October 2022; accepted 5 January 2023. *(Corresponding author: Tarique Anwar.)*

Ashraf Kamal is with ACL Digital, Bengaluru 560029, India (e-mail: ashrafkamal.mca@gmail.com).

Tarique Anwar is with the Department of Computer Science, University of York, YO10 5DD York, U.K. (e-mail: tarique.anwar@york.ac.uk).

Vineet Kumar Sejwal is with the Delhi Directorate of Education, Delhi 110054, India (e-mail: vineetsejwal.jmi@gmail.com).

Mohd Fazil is with the Centre for Transformative Learning, University of Limerick, Limerick, V94 T9PX Ireland (e-mail: mohd.fazil@ul.ie).

Digital Object Identifier 10.1109/TCSS.2023.3236527

[1] https://www.aljazeera.com/news/2020/2/10/indias-bjp-slammed-for-offensive-tweet-on-anti-caa-protesters [accessed on 23-Oct-2022].

[2] https://thewire.in/communalism/delhi-riots-misinformation-radicalisation-social-media [accessed on 23-Oct-2022].

2329-924X © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

which makes it time-consuming, cost-intensive, and ad hoc. Most of the existing deep learning methods are based on recurrent neural networks (RNNs) [13], [14] such as long short-term memory (LSTM) and gated recurrent unit (GRU), and convolutional neural networks (CNNs) [11], [15]. While RNNs are able to capture the sequential context, they perform marginal due to the short length of the nonliteral social posts [16]. CNNs capture the semantics of a sentence by performing convolution operations over the tweets, but suffer from being translation invariant [17]. The contextual language models such as BERT [18] and its variants perform much better than the others, as shown in [19].

We formulate the problem of HS detection as a binary classification problem. As discussed earlier, the role of context is important in this problem. Context is multifaceted. In this article, we particularly mean the linguistic *context of utterance*. It is defined by the surrounding language and conversations. The *social context* [20], [21], on the other hand, is defined by the culture and societal hierarchies. We do not explicitly study this form of context, but it may be reflected implicitly by the type of hateful language used in the datasets, given the way these are collected. The linguistic *context of utterance* exists in different orientations and different directions of the text [22]. To this end, we develop a novel deep learning model, called `BiCapsHate`, for HS detection. The model consists of several advanced layers of deep neural networks (DNNs), each one dedicated to capture different specific properties of HS. It starts with an input layer that processes the input text, and follows on to an embedding layer that embeds the text into a numeric representation, a BiCaps layer that learns the sequential and contextual representation in different orientations, a dense layer that prepares the model for final classification, and lastly the output layer producing the resulting class. The novelty of the proposed model lies mainly in the *BiCaps* layer. It learns the contextual information with respect to different orientations in both forward and backward directions of the input text. This is further aided by our rich hand-crafted auxiliary features including the `Hatebase` lexicon, making the model *well-informed*. Overall, we make the following major contributions in this article.

1) We investigate the problem of HS detection over social media. Deep learning approaches generally perform better than classical machine learning approaches.
2) We develop a novel deep learning model called `BiCapsHate` for HS detection. Our model is based on five advanced layers of DNNs and well-informed by a rich set of auxiliary features.
3) As part of our `BiCapsHate` model, the developed *BiCaps* layer effectively learns the contextual information in different orientations of the input text. This is achieved with the help of stacked bidirectional LSTM (BiLSTM) networks integrated with forward and backward capsules, attention and auxiliary features including the `Hatebase` lexicon.
4) We conduct extensive experiments to demonstrate the efficacy of the proposed model in comparison to the existing state-of-the-art methods on five benchmark datasets.

TABLE I
SUMMARY OF EXISTING LITERATURE ON HS DETECTION

Literature	Features / Models	Language	Classfication
Warner and Hirschberg [8]	unigram, part of speech, template-based	English	Binary
Burnap and Williams [23]	*n*-grams, dependencies, and hateful *n*-grams	English	Binary
Alfina et al. [24]	*n*-grams, negative sentiments	Indonesian	Binary
Waseem and Hovy [9]	*n*-grams	English	Binary
Djuric et al. [25]	embeddings, paragraph2vec	English	Binary
Kumar et al. [26]	*n*-grams, tf-idf	Tamil, Malayalam	Binary
Davidson et al. [12]	*n*-grams, part-of-speech, tf-idf	English	Ternary
Malmasi and Zampieri [27]	*n*-grams, word skip-gram	English	Ternary
Badjatiya et al.[15]	CNN, LSTM	English	Binary
Park and Fung [28]	Hybrid CNN	English	Ternary
Zhang et al. [14]	CNN, GRU	English	Multiclass
Gamback and Sikdar [29]	CNN	English	Multiclass
Founta et al. [30]	RNN	English	Multiclass
Roy et al. [11]	CNN	English	Binary
Roy et al. [31]	Transformers, BERT	Tamil Malayalam	Multiclass
Pamungkas et al. [10]	Transformers, BERT	English	Binary
Wang and Ding [32]	BiGRU-capsule, LSTM	English	Binary
Ding et al. [22]	BiGRU, Capsule	English	Binary
Mossie and Wang [33]	GRU, RNN	Amharic	Binary
Fortuna et al. [19]	BERT, ALBERT	English	Multiclass
Ghosh et al. [7]	CNN, GRU, Stacked ensemble	English, Multi-domain	Binary
Proposed - `BiCapsHate`	BiLSTM, Capsule, BiCaps, Attention, Hatebase, Auxiliary features	English	Binary

The rest of this article is organized as follows. We present the existing literature in Section II and an overview in Section III. Section IV presents the architectural details of the proposed model and Section V demonstrates the efficacy of the proposed method. Finally, this article is concluded in Section VI.

II. RELATED WORK

HS lacks a universally accepted definition. Social networks have their own definitions for content moderation. According to Twitter, HS is a tweet "*that promotes violence against or directly attack or threaten other people on the basis of race, ethnicity, nationality, sexual orientation, gender, religious affiliation, age, disability, or serious disease.*"[3] The problem of HS detection is better addressed by supervised learning approaches. Table I presents a summary of the existing literature. Some of the early works were based on classical machine learning, whereas recent works have found deep learning more effective.

A. Classical Machine Learning Approach

This group of works defines a set of relevant features and train a machine learning model on them for classification. Warner and Hirschberg [8] used unigram, part of speech, and other template-based features. They trained a support vector machine (SVM) model with the linear kernel to classify hate and nonhate contents. Some works have also used logistic regression (LR) [9], [23], [25]. Other similar works [23], [24] use *n*-grams, hateful *n*-grams, and typed

[3]https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy [accessed on 23-Oct-2022].

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

dependencies to extract features and train classifiers such as Bayesian LR, random forest, SVM, and voted ensemble, over Twitter data. Some studies [9] have found character n-gram features more discriminatory than word n-grams in segregating the hateful and nonhateful content. The paragraph2vec neural model used in [25] is another effective way for vector representation. Some works highlight a subtle difference between hateful and offensive language [12], [27]. They treat the task as a multilabel classification of hate, offensive, and normal contents. Davidson et al. [12] developed an LR model with L2 regularization using a large set of shallow and deep features. Malmasi and Zampieri [27] used character n-gram, word n-gram, and word skip-gram features to train a linear SVM classifier for the multilabel classification. This approach also been used for script-mixed and code-mixed contents [26]. However, classical machine learning requires feature engineering, which is a manual and time-consuming task involving human bias.

B. Deep Learning Approach

Deep learning has become a preferred choice of researchers for HS detection, especially because it does not depend on the tedious feature engineering. Badjatiya et al. [15] evaluated random, *Global Vectors* (`GloVe`) [34], and *fastText* [35] word embeddings with various neural network architectures such as CNN and LSTM for HS detection in tweets. In [28], a hybrid model based on LR and CNN is used to classify abusive and nonabusive tweets. Zhang et al. [14] developed a deep learning model with a combination of convolution and GRU layers. Most of these models use only word embedding representations [14], [15], [29], [30]. Cao et al. [36] incorporated multifaceted information such as semantic, sentiment, and topical information, by using a combination of three different word embeddings, sentiment embedding, and topic representation. A deep learning model based on CNN, LSTM, and attention is trained on the representations. Roy et al. [11] developed a CNN-based framework for binary classification of hate. Sometimes, social media users post cross-lingual textual contents. For example, English used together with Hindi (aka *Hinglish*). Some works [10], [31] developed deep learning models based on CNN, LSTM, BiLSTM, and more advanced neural network architectures for HS detection in such cross-lingual texts. Capsule networks [37] have recently been found effective in capturing the context of textual contents in different orientations. These networks have been found promising also for HS detection [22], [32]. Transformer-based language models pretrained on hate-specific contents such as fBERT [38], HateBERT [39], and ToxicBERT [40] have been found quite effective for this task. A recent study created a multidomain HS corpus and developed a simple deep learning baseline [7]. Some analogous research directions are identification of hate-targeted vulnerable communities [33] and generalization of HS detection models [19].

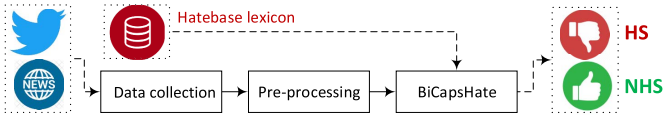

Fig. 1. Components of the proposed HS detection method.

C. Current Status and Limitations

Existing research has found deep learning approaches fairly effective for HS detection. Several works have highlighted the importance of context in disambiguating textual contents and identifying hate. The existing works, particularly those based on capsule networks, have started investigating and exploiting the context. However, the existing works still have some limitations. First, the approach in which they use capsule networks is ineffective in capturing the bidirectional context. Second, they lack the useful neural networks for *attention* to emphasize on the hate context. Third, they do not use any HS-related lexicon to design a domain-relevant model. The `BiCapsHate` model, proposed in this article, addresses all these existing limitations.

III. Problem and Solution Overview

A. Problem Statement

Let us suppose, we have a set of textual posts, published by users on an OSM platform. The problem of *HS detection* considered in this article aims to classify each post into the classes of *HS* and *non-HS* (NHS). Thus, it is a binary classification problem.

B. Proposed Method Overview

Fig. 1 shows the overall components of the proposed HS detection method. We begin with the preliminary steps of *data collection* and *preprocessing*, and then follow on to our novel bidirectional capsule-based deep learning model called `BiCapsHate`. The preprocessing step rigorously cleans and case-normalizes the raw tweets using the NLTK toolbox.[4] The resulting fine-grained dataset is passed into the input layer of our multilayered `BiCapsHate` model. It also exploits the `HateBase` lexicon, as shown in the figure, and a rich set of hand-crafted shallow and deep features. The architectural details of the model are presented in Section IV.

IV. The `BiCapsHate` Model

We develop our novel deep learning model, called `BiCapsHate`, by addressing the existing limitations. Fig. 2 shows the detailed architecture of the proposed model. It consists of five advanced layers of DNNs. As shown in the figure, the social media text is first collected using their Tweet IDs provided in the datasets and preprocessed (shown in the top-left corner) and then passed into the *input* layer of the model. The input text (tweet) is converted into its corresponding input vector, and it is forwarded to the *embedding* layer. The output 2-D matrix produced by the

[4]http://www.nltk.org/howto/sentiment.html [accessed on 23-Oct-2022].

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

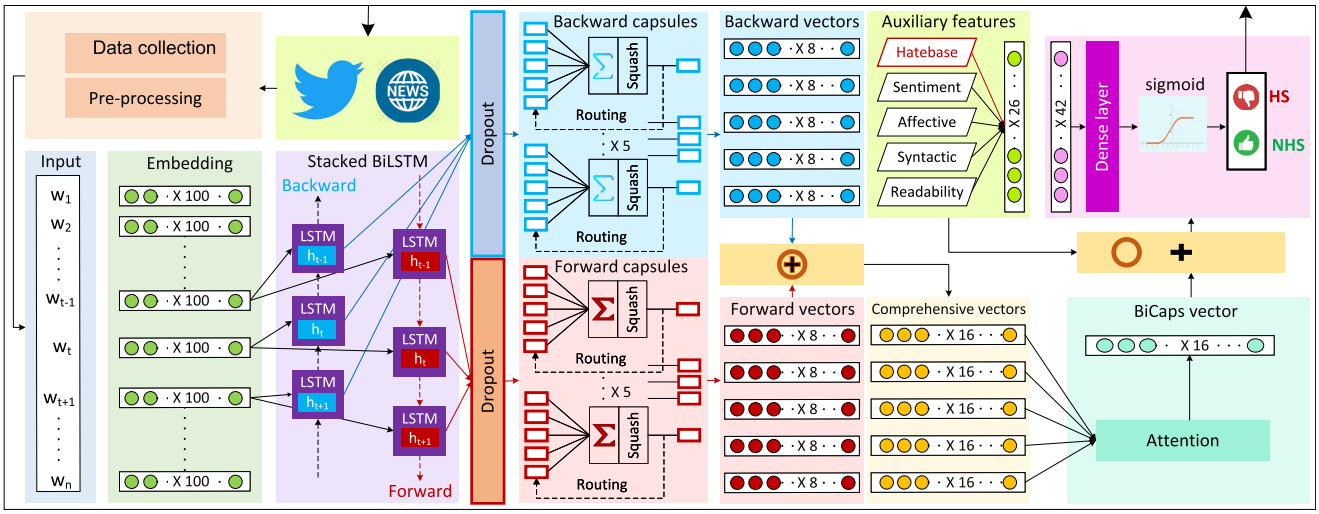

Fig. 2. Architecture of the proposed BiCapsHate model.

embedding layer is passed into the *BiCaps* layer. It consists of a stack of BiLSTM networks, bidirectional capsules, an attention network and auxiliary features. The forward and backward directions of the stacked BiLSTM together with the two opposite directions of the capsules generate contextual representations from different orientations. The dynamic routing algorithm employed inside the capsules increases the weight values of the important HS-related latent features present in the input text. Two 2-D matrices corresponding to the forward and backward capsules are generated as output. These matrices are concatenated together to produce a comprehensive 2-D matrix. This matrix is then passed into the attention network to emphasize maximum attention on the HS-related words. It distills the most significant information and generates an input vector. Moreover, several shallow and deep auxiliary features, including the Hatebase lexicon (detailed later in Section IV-C), are also extracted from five different categories. We generate a feature vector with all these additional information. The feature vector is concatenated with the BiCaps vector, which further enriches the representation. One part of the vector is learned from a series of neural networks and the other part is captured from inherent characteristics. The concatenated vector is passed into the *dense* layer followed by a *sigmoid* activation function in the *output* layer to classify the input text as either HS or NHS. We present the details of each layer in the following sections.

A. Input Layer

Given the preprocessed input text of a tweet t of w_t words, the input layer first tokenizes its textual contents. The tokens are indexed in a dictionary before further processing, if they are not already indexed by the previous tweets. The tokens are then converted into a numeric vector u, by replacing each token with its index in the dictionary. The length of u may vary for different tweets, depending on the tweet length. To maintain a fixed-length p of input vectors of all the tweets, we transform u to a fixed-length padded-vector v, such that $|v| = p \geq |u|$. The first $|u|$ entries of v are filled by u and the remaining $(p - |u|)$ entries are filled with padding values of zero. The fixed-length resulting vector $v \in R^{1\times p}$ is then forwarded to the embedding layer.

B. Embedding Layer

Text embedding provides learned word representations in the form of low-dimensional dense vectors in a continuous embedding space. We use GloVe in the proposed model. It is a popular pretrained word embedding model based on distributed word representations. It encodes the semantic information between word pairs in a co-occurrence matrix and applies an unsupervised algorithm to generate the vector representations. It has been found to be highly effective. As our model is to be applied on social data, we use a Twitter-specific GloVe embedding that consists of 27 billion tokens. It is able to generate embeddings of almost all the tokens in our datasets. Each token of the input vector is converted into its distributional vector representation of dimension d. As a result, the input vector is converted to a matrix $I \in R^{p\times d}$, where each row of the matrix is a vector representation of the corresponding token.

C. BiCaps Layer

The embedding matrix I, representing the input text, is processed through a neural network layer called BiCaps layer. It consists of three different types of neural networks. It starts with a stacked LSTM, treating the forward and backward textual representations separately, which is followed by two capsule networks corresponding to each direction. The resulting forward and backward vectors are concatenated and processed through an attention network to generate a BiCaps vector. Overall, the BiCaps layer aims to achieve the following three objectives, in order to detect HS in the input text: 1) extract the HS-related contextual sequences using stacked BiLSTM networks; 2) capture the different orientations and spatial relationships using two capsule networks; and

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

3) emphasize toward important contextual words using attention.

1) Stacked BiLSTM: LSTM [41] is as a special type of RNN. It is capable of handling long-term dependencies and avoiding the vanishing gradient descent problem. Although LSTM has an ability to extract HS-based contextual sequences from textual data, it is operational in only forward direction. A stacked LSTM architecture [42] consists of multiple LSTM layers, where each preceding LSTM layer provides a sequence output rather than a single-value output to the following LSTM layer. It has been found to be effective for sequence modeling [42].

The proposed model employs a stacked BiLSTM network. It consists of two stacks of 128 LSTM units, one each for the forward and backward directions. Thus, it handles long-term dependencies and extracts HS-based contextual sequences more effectively from the input text in both forward and backward directions. The semantically rich low-dimensional vectors produced by the embedding layer are passed separately to the forward and backward stacked LSTMs. It generates the forward and backward contextual sequences. The two hidden states $\overrightarrow{h_f}$ and $\overleftarrow{h_b}$ are produced as outputs of the forward and backward stacked LSTMs, respectively. In order to minimize the overfitting and enhance the generalization error, the previous outputs are processed through respective dropouts with a rate of 0.5. The resulting representations are passed separately to two different capsule networks.

2) Capsule Network: A CNN or ConvNet is a class of DNN, used in many image and text processing applications. Traditional CNNs suffer from limitations in recognizing features of different orientations. The pooling technique used in them leads to a loss of some important features. CNNs are also unable to retain spatial relationships between the learned features in deeper layers. Capsule networks [43] overcome these limitations by recognizing the part-whole spatial relationship across features in the textual data.

Capsule networks were introduced for computer vision applications. A capsule is a group of neurons that perform computations on the inputs and encapsulate the results as an activity vector of highly informative outputs. The vector represents the instantiation parameters of a specific type of entity such as an object (e.g., eye or nose) or an object part [44]. Its length is used to represent the probability that the entity exists and its orientation represents the instantiation parameters. After the features are detected by lower level capsules, these are sent to higher level capsules through the dynamic routing algorithm. In this way, capsules learn the spatial relationships between the different entities present in an image. Capsule networks work in a similar way for text processing [45]. While an image is represented by a matrix of pixels, a piece of text is represented by a matrix of its embedding, where each row is an embedding vector of a word. The lower level capsules encode the spatial information from the input embedding matrix as activity vectors containing important semantic information of the text. The connection strengths between lower- and upper-level capsules are determined by repetitive routing as part of the dynamic routing algorithm with the help of a coupling coefficient. The algorithm is responsible for computing the weight values of latent features, encoding of spatial information, and mining rich contextual information in different orientations. The computed connection strengths are used to send the information to upper levels with a good fit. In this way, capsules learn the spatial relationships between textual entities or words through the dynamic routing algorithm and represent the inherent properties. Thus, it makes sense of words even if they do not appear adjacently. Considering the textual semantics and context, they enrich the information by encoding the local ordering of words in the input text [46], [47]. Thus, capsule networks are able to extract salient features in different orientations of the input text. They capture the textual expression ability and help in retrieving the latent contextual information.

Considering the above-mentioned characteristics, we use two capsule networks in the proposed model, corresponding to the forward and backward directions. Each network consists of five capsules of dimension eight. The capsule networks capture the different orientations and spatial relationships of HS-related contextual information present in the input text. To this end, the two stacks $\overrightarrow{h_f}$ and $\overleftarrow{h_b}$ of hidden state vectors generated from the stacked LSTMs, after processing through the dropouts, are passed separately as inputs to the forward and backward capsule networks, respectively. Equation (1) shows the prediction vector $\overrightarrow{\hat{u}_{j|i}}$ of capsule j from capsule i of lower layer, for the forward contextual representation. It is calculated from a learned transformation weight matrix $\overrightarrow{w_{ij}}$ and the hidden state vector $\overrightarrow{h_f}$ given as input to the forward capsule network. Coupling coefficients determine the degree to which lower capsules are promoted to upper capsules. The vector of coupling coefficients $\overrightarrow{c_{ij}}$ in between the input and output layers is calculated via *softmax* as shown in (2), wherein b_{ij} is the log prior probabilities that capsule i should be coupled to capsule j. b_{ij} is initialized to 0. The higher weight values of important features directly depend on larger values of coupling coefficients $\overrightarrow{c_{ij}}$ [48]. The output of higher layer capsule $\overrightarrow{s_j}$ is calculated by summing up of all the prediction vectors with the coupling coefficient vectors $\overrightarrow{c_{ij}}$, as shown in (3). The coupling coefficient vectors $\overrightarrow{c_{ij}}$ are updated iteratively via the dynamic routing algorithm. As a result, unimportant and unrelated NHS words are ignored in the forward direction of the input text. The total value of the coupling coefficients between the capsules of input and output layers is 1. Equation (4) shows a nonactivation function called *squash* used for normalizing and generating the output vector $\overrightarrow{v_j}$ in the forward direction of the input text

$$\overrightarrow{\hat{u}_{j|i}} = \overrightarrow{w_{ij}}\overrightarrow{h_f} \tag{1}$$

$$\overrightarrow{c_{ij}} = \frac{\exp(b_{ij})}{\sum_t \exp(b_{it})} \tag{2}$$

$$\overrightarrow{s_j} = \sum_{i=1}^{n} \overrightarrow{c_{ij}}\overrightarrow{\hat{u}_{j|i}} \tag{3}$$

$$\overrightarrow{v_j} = \frac{\|\overrightarrow{s_j}\|^2}{1+\|\overrightarrow{s_j}\|^2}\frac{\overrightarrow{s_j}}{\|\overrightarrow{s_j}\|}. \tag{4}$$

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

The same process is separately applied on the backward hidden state vector $\overleftarrow{h_b}$ using the backward capsule network. After computing the backward prediction vector $\overleftarrow{\hat{u}_{j|i}}$, the backward capsule $\overleftarrow{s_j}$ is used to generate the backward output vector, as shown in (5). It captures the different orientations in backward direction of the input text. Finally, (6) generates the comprehensive output vectors f_c as a 5×16 matrix by concatenating the 4×8 matrices corresponding to the forward and backward output vectors

$$\overleftarrow{v_j} = \frac{\|\overleftarrow{s_j}\|^2}{1 + \|\overleftarrow{s_j}\|^2} \frac{\overleftarrow{s_j}}{\|\overleftarrow{s_j}\|} \tag{5}$$

$$f_c = \overrightarrow{v_j} \oplus \overleftarrow{v_j}. \tag{6}$$

3) Attention Network: Attention is a DNN-based technique that aims to capture the most important information according to overall semantics of the input data [49]. It is achieved by focusing on relevant contextual words appearing in the input text. In the proposed model, we consider a word-level attention mechanism. Our attention network receives the 5×16 matrix of comprehensive vectors f_c as input and captures the relative importance by giving focus on HS-related tokens. Equation (7) shows the hidden representation u_t generated by a $\tanh(\cdot)$ activation function. The input to tanh is the product of a learned trainable weight matrix W_w and f_c added by a bias b_w vector. Equation 8 computes the normalized similarity α_t between the hidden representation u_t using the `softmax`. Lastly, (9) calculates the BiCaps vector s_i of the input text based on α_t and f_c

$$u_t = \tanh(W_w f_c + b_w) \tag{7}$$

$$\alpha_t = \frac{\exp(u_t)}{\sum_t(\exp(u_t))} \tag{8}$$

$$s_i = \sum_{i=1}^{t} \alpha_t f_c. \tag{9}$$

D. Enhancement With Lexicon and Other Auxiliary Features

HS characterizes specific properties, some of which are obvious whereas others are obscure. One of the most prominent properties is the usage of *hate lexicon* terms in the input text. There exists a well-established benchmark lexicon for HS called *Hatebase*.[5] A high match of the input text with Hatebase terms indicates a high possibility of being an HS. We consider the Hatebase lexicon as a category of auxiliary features. Similarly, there also exist some other important features, such as sentiment, that can further aid our deep learning model. Therefore, we hand-craft a total of 26 auxiliary features grouped into five categories. These features aim to enhance the performance of our `BiCapsHate` model. A feature vector $l_v \in R^d$ of $d = 26$ dimensions is generated using these auxiliary features.

1) Hatebase Lexicon: Hatebase lexicon features are effective in the identification of offensive terms [12]. We consider ten independent lexical features based on this lexicon: *hate score, unambiguous score, offensiveness score, nationality score, ethnicity score, religion score, gender score, sexual orientation score, disability score,* and *class score*. The score of each feature is computed as the normalized summation of individual word scores in the input text.

2) Sentiment: Generally, there is a significant presence of sentiment in the context of hate. HS-related texts exhibit a high degree of negative polarity in comparison to NHS [50]. We consider four sentiment-related independent features: *polarity score, subjectivity score, count of positive words,* and *count of negative words*. These are computed using `TextBlob`.[6]

3) Affective: Affective contents related to moods and feelings are important constituents in HS-related textual data. *Affective Norms for English Words* (`ANEW`) [51] is a collection of 1034 words characterized in affective dimensions. Based on `ANEW`, we consider three independent affective features: *valence score, arousal score,* and *dominance score*. Each of these scores is computed as the normalized summation of individual word scores in the input text.

4) Syntactic: Syntactic features are important in the identification of targets and intensity of HS [12]. We consider 4 syntactic features based on the presence of POS tags of *verb (VB), noun (NN), adverb (RB),* and *adjective (JJ)*. These features are extracted using `NLTK`.[7] To this end, each input text is tokenized and the respective POS tags are assigned using *Penn Treebank*.[8] All the four considered features are independent and binary in nature. A score of 1 is assigned if the corresponding POS tag appears at least three times in the input text, otherwise it is set to 0.

5) Readability: Readability features capture the language quality of a text [12]. `Textstat`[9] is a Python library that can calculate some statistical measures related to readability and complexity of a text. Based on Textstat, we consider five independent readability features: *Flesch reading ease score, Gunning fog index score, automated readability index score, Coleman Liau index score,* and *syllable count score*. The score of each feature is computed as the normalized summation of individual words scores in the input text.

E. Dense and Output Layers

The BiCaps vector s_i generated by the BiCaps layer in Section IV-C and the auxiliary feature vector l_v generated in Section IV-D are concatenated as $c_v = l_v \oplus s_i$. The concatenated vector c_v is then passed on to a fully connected dense layer. We use a binary cross-entropy loss function for training the model. Thereafter, the classification is finally done by a *sigmoid* function into the classes of HS and NHS.

V. Experimental Evaluation

We conduct extensive experiments on five real benchmark datasets. Our complete source code and detailed results are publicly available at GitHub repository.[10]

[5]https://hatebase.org/ [accessed on 23-Oct-2022].

[6]https://textblob.readthedocs.io/en/dev/ [accessed on 23-Oct-2022].

[7]https://www.nltk.org/ [accessed on 23-Oct-2022].

[8]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html [accessed on 23-Oct-2022].

[9]https://pypi.org/project/textstat/ [accessed on 23-Oct-2022].

[10]https://github.com/Ashraf-Kamal/BiCapsHate

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
DATASET STATISTICS

	Raw dataset		Pre-processed			
			Imbalanced		Balanced	
Datasets	**#HS**	**#NHS**	**#HS**	**#NHS**	**#HS**	**#NHS**
DS-1	1430	4163	1155	3475	1155	1155
DS-2	2740	40275	2615	37077	2615	2615
DS-3	435	1093	406	927	406	406
DS-4	2013	27517	1421	16975	1421	1421
DS-5	5935	14212	5647	7383	5647	5647

TABLE III
EXPERIMENTAL AND PARAMETER SETTINGS

Experimental settings		Parameter settings	
Environment	**Configuration**	**Parameters**	**Value**
Operating system	Windows 10 64-bit	Embedding dimension	100
Processing machine	Intel i-3 6006U CPU	Padding sequences	20 (65 in DS-3)
RAM	8 GB	#Neurons (BiLSTM)	128
Programming language	Python 3.7	Dropout	0.5
Deep learning library	Keras 2.2.4	Routing	3
Twitter-API	Tweepy 3.7.0	#Capsules	5
-	-	Capsule dimensions	8
-	-	Optimizer	Adam

A. Datasets

We consider five real benchmark datasets: 1) *DS-1* [12]: It consists of 24 802 tweets, each labeled as either *hate*, *offensive*, or *neither*. Since we focus on the binary classification of hate, we consider only the tweets labeled as *hate* and *neither* for our HS and NHS classes. It results into a total of 5593 tweets; 2) *DS-2* [50]: It consists of 80 000 tweets, each labeled as either *abusive*, *hateful*, *normal*, or *spam*. Similar to the previous dataset, we consider only the tweets labeled as *hateful* and *normal* for our HS and NHS classes. It results into 43 015 tweets; 3) *DS-3* [52]: It consists of 1528 `Fox` news user comments that belong to either HS or NHS classes. We consider this dataset, as it is; 4) *DS-4*: This dataset is obtained from `kaggle`,[11] consisting of 31 962 tweets. Each of them is labeled as either HS or NHS. After some filtering, we consider a total of 29 530 tweets; and 5) *DS-5* [53]: It consists of 20 148 tweets, each labeled as either *hateful*, *offensive*, *normal*, or *undecided*. Like others, we consider only the tweets labeled as *hateful* and *normal* for our HS and NHS classes. Table II shows our dataset statistics. The second column of the table shows the numbers of HS and NHS instances in the raw datasets. There exist some redundant instances in these datasets, which are removed through preprocessing. The third column shows the number of instances after preprocessing. These datasets are imbalanced in terms of the number of instances in each class. For a detailed analysis, we further create another balanced version having equal number of HS and NHS instances.

B. Experimental and Model Parameter Settings

The proposed method is implemented in Python, using Keras[12] library. Table III shows our experimental and model parameter settings. The datasets are split into training and validation sets in the ratio of 80:20. The batch size, verbose, and epoch are set to 128, 2, and 50, respectively.

C. Compared Methods and Evaluation Metrics

1) State-of-the-Art Methods: We consider the following six existing state-of-the-art methods.

1) *Ding et al. [22]:* This method consists of a stack of bidirectional GRU (BiGRU) and capsule network layers in its deep learning model for HS detection on Twitter datasets.
2) *Mossie and Wang [33]:* This method consists of two RNN (GRU/LSTM) layers for HS detection and identification of vulnerable communities on Facebook and Twitter.
3) *Roy et al. [11]:* This method consists of a CNN-based deep learning model for HS detection on Twitter dataset.
4) *fBERT [38]:* It is a BERT model, pretrained on a large English offensive language corpus (SOLID), containing more than 1.4 million offensive instances.
5) *HateBERT [39]:* It is another BERT model, pretrained on a Reddit dataset of communities banned for being offensive, abusive, or hateful.
6) *ToxicBERT [40]:* ToxicBERT is another BERT-based pretrained model for toxic content detection.

2) Baseline Methods: We consider ten baseline methods, consisting of three classical machine learning methods (naive Bayes, decision tree and random forest), six DNN models (DNN, CNN, LSTM, BiLSTM, GRU, and BiGRU), and one transformer-based pretrained language model (BERT [18]—bidirectional encoder representations from transformers). Our DNN has two hidden layers with 128 neurons each, CNN has 128 filters with a width of 3 each, ad LSTM, BiLSTM, GRU, and BiGRU have 128 neurons. We use BERT-base-uncased version with 768-dimensional embedding having 12 hidden layers and 12 attention heads.

3) Evaluation Metrics: We evaluate the performance in terms of four standard metrics—*Precision*, *Recall*, *F-score*, and *Accuracy*.

D. Experimental Results

This section presents our experimental results and evaluation of the proposed `BiCapsHate` model in comparison to the six state-of-the-art methods and ten baseline models. Tables IV and V show the results on the balanced and imbalanced datasets, respectively, in terms of *precision*, *recall* and *f-score*. Observe from Table IV that our `BiCapsHate` consistently outperforms all the compared methods over balanced datasets. The proposed model also completely outperforms the compared methods over imbalanced datasets over DS-1, DS-3, and DS-5 (3 out of 5). ToxicBERT

[11]https://www.kaggle.com/vkrahul/twitter-hate-speech?select = train_E6oV3lV.csv [accessed on 23-Oct-2022].

[12]https://keras.io/ [accessed on 23-Oct-2022].

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IV

PERFORMANCE RESULTS ON BALANCED DATASETS. THE PROPOSED BiCapsHate MODEL SHOWS THE BEST PERFORMANCE (BOLD)

Datasets →	DS-1			DS-2			DS-3			DS-4			DS-5		
Methods ↓	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
BiCapsHate model	**0.9413**	**0.9512**	**0.9423**	**0.8329**	**0.8212**	**0.8222**	**0.8412**	**0.8357**	**0.8333**	**0.9187**	**0.9311**	**0.9214**	**0.9401**	**0.9377**	**0.9381**
Ding et al. [22]	0.4621	0.5226	0.4923	0.1143	0.2223	0.1535	0.2424	0.4549	0.3132	0.3409	0.4001	0.3705	0.8105	0.7612	0.7799
Mossie and Wang [33]	0.9223	0.9143	0.9132	0.8126	0.7942	0.8036	0.4925	0.8128	0.6126	0.8804	0.8924	0.8812	0.8130	0.8186	0.8127
Roy et al. [11]	0.8821	0.8844	0.8832	0.5243	0.5023	0.5132	0.8222	0.7741	0.7937	0.8921	0.7849	0.8335	0.7627	0.5948	0.6073
fBERT	0.9104	0.8638	0.8858	0.4104	0.3258	0.3575	0.7232	0.5359	0.5940	0.8416	0.7657	0.7996	0.8848	0.8359	0.8591
HateBERT	0.9058	0.8670	0.8853	0.3988	0.2958	0.3318	0.5819	0.4975	0.5269	0.8255	0.7209	0.7658	0.8997	0.8314	0.8633
ToxicBERT	0.8932	0.8916	0.8924	0.4825	0.5046	0.4910	0.7157	0.7048	0.7099	0.7930	0.7940	0.7935	0.9043	0.9125	0.9082
Naïve Bayes	0.8404	0.8408	0.8406	0.4848	0.4871	0.4859	0.6603	0.6511	0.6552	0.7008	0.6958	0.6984	0.8201	0.8203	0.8202
Decision Tree	0.9001	0.8945	0.8975	0.5124	0.4968	0.4942	0.6314	0.6318	0.6316	0.6814	0.6824	0.6819	0.8511	0.8498	0.8505
Random Forest	0.8809	0.8815	0.8812	0.4935	0.4951	0.4942	0.6636	0.6638	0.6637	0.7249	0.7001	0.7124	0.8525	0.8575	0.8551
DNN	0.8917	0.8521	0.8719	0.6123	0.5827	0.5925	0.7232	0.6627	0.6931	0.8338	0.8232	0.8236	0.8494	0.7691	0.8055
CNN	0.8533	0.8444	0.8441	0.6331	0.6135	0.6231	0.7143	0.6632	0.6837	0.8006	0.7915	0.7912	0.8411	0.7745	0.8048
LSTM	0.9221	0.9114	0.9116	0.7132	0.7121	0.7135	0.7747	0.7432	0.7538	0.8824	0.8832	0.8827	0.8266	0.7901	0.8062
BiLSTM	0.9103	0.9021	0.9015	0.8044	0.7936	0.7941	0.7748	0.7532	0.7638	0.9003	0.9101	0.9002	0.8233	0.7904	0.8049
GRU	0.9042	0.9202	0.9121	0.7432	0.7333	0.7331	0.7626	0.7421	0.7523	0.9016	0.8911	0.8913	0.8264	0.7915	0.8068
BiGRU	0.9247	0.9143	0.9145	0.8126	0.8041	0.8035	0.7632	0.7424	0.7528	0.8914	0.9018	0.8916	0.8143	0.8073	0.8091
BERT	0.9060	0.8148	0.8509	0.4179	0.3104	0.3491	0.7445	0.6595	0.6966	0.8424	0.7139	0.7612	0.8860	0.8418	0.8630

TABLE V

PERFORMANCE RESULTS ON IMBALANCED DATASETS. THE BEST RESULTS OVER EACH DATASET ARE SHOWN IN BOLD

Datasets →	DS-1			DS-2			DS-3			DS-4			DS-5		
Methods ↓	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
BiCapsHate model	**0.9457**	**0.9181**	**0.9251**	0.6325	0.5113	0.5606	**0.9085**	**0.8111**	**0.8502**	0.8992	0.7144	0.7962	**0.9305**	**0.9125**	**0.9231**
Ding et al. [22]	0.3315	0.2729	0.3005	0.3543	0.3916	0.3738	0.3023	0.4001	0.3445	0.4011	0.4324	0.4142	0.8169	0.7202	0.7570
Mossie and Wang [33]	0.9112	0.8907	0.9045	0.4149	0.2723	0.3336	0.8009	0.7312	0.7449	0.8711	0.6528	0.7414	0.7753	0.7857	0.7536
Roy et al. [11]	0.8914	0.6636	0.7625	0.2609	0.1218	0.1613	0.8827	0.6731	0.7629	0.8344	0.5201	0.6437	0.5722	0.3324	0.3185
fBERT	0.9368	0.9177	0.9249	0.9281	0.9144	0.9211	0.7716	0.6343	0.6822	0.9475	0.9208	**0.9338**	0.8989	0.8396	0.8673
HateBERT	0.9234	0.9023	0.9125	0.9270	0.9125	0.9195	0.8144	0.6226	0.6809	0.9495	0.9075	0.9275	0.8878	0.8000	0.8398
ToxicBERT	0.9146	0.9106	0.9111	**0.9470**	**0.9235**	**0.9240**	0.7730	0.7666	0.7695	0.9290	**0.9329**	0.9315	0.8620	0.8689	0.8645
Naïve Bayes	0.9017	0.9031	0.9025	0.6625	0.6925	0.6823	0.7185	0.7283	0.7215	0.8375	0.8389	0.8381	0.8305	0.8306	0.8305
Decision Tree	0.9101	0.9109	0.9105	0.6124	0.5968	0.6005	0.7124	0.7348	0.7236	0.7299	0.7855	0.7452	0.8666	0.8655	0.8605
Random Forest	0.9204	0.9002	0.9103	0.6011	0.6111	0.6055	0.6555	0.6882	0.6721	0.6211	0.6664	0.6456	0.8625	0.8675	0.8662
DNN	0.8931	0.7529	0.8128	0.1817	0.1131	0.1425	0.7443	0.3649	0.4848	0.7711	0.5131	0.6176	0.8470	0.7355	0.7843
CNN	0.8521	0.7123	0.7719	0.3427	0.3818	0.3622	0.7334	0.3732	0.4931	0.7419	0.4224	0.5437	0.8451	0.7350	0.7840
LSTM	0.9332	0.8887	0.9035	0.4012	0.1621	0.2332	0.7516	0.5843	0.6511	0.8415	0.6813	0.7532	0.8030	0.7733	0.7849
BiLSTM	0.9206	0.8923	0.9017	0.5943	0.4028	0.4822	0.7934	0.6601	0.7232	0.8732	0.6912	0.7716	0.8233	0.7904	0.8049
GRU	0.9321	0.9012	0.9111	0.4843	0.3333	0.3936	0.7612	0.6530	0.7024	0.8648	0.7024	0.7736	0.8264	0.7915	0.8068
BiGRU	0.9106	0.9043	0.9024	0.5932	0.4933	0.5443	0.7707	0.6332	0.6915	0.8732	0.6825	0.7627	0.8143	0.8073	0.8091
BERT	0.9290	0.9118	0.9243	0.9306	0.8971	0.9128	0.7939	0.5156	0.5871	**0.9543**	0.9003	0.9257	0.8847	0.8147	0.8474

performs the best in DS-2 and fBERT performs the best in DS-4 (in terms of *f-score*). We achieve *f-scores* of 94%, 82%, 83%, 92%, and 93% in the balanced datasets, respectively, and 92%, 56%, 85%, 79%, and 92% in the imbalanced datasets, respectively. Among the other methods, Mossie and Wang [33] and transformer-based language models mostly perform the best, followed by some baseline models. Table VI shows the training and validation accuracy (VA) of BiCapsHate, in comparison to the other methods. Observe that the proposed model completely outperforms the compared methods over DS-5. We also outperform others over balanced DS-1, balanced DS-2, and imbalanced DS-3 in terms of VA, TA, and TA, respectively. Overall, we outperform in seven instances, whereas fBERT, HateBERT, ToxicBERT, and BERT outperform in 6, 2, 1, and 4 instances respectively. We are able to achieve up to 95% training accuracy (TA) and 92% VA. The results in the table show that there is a gap between TA and VA of our model as well as all the compared methods, over DS-2. It indicates that all the models overfit in this dataset. We believe that this behavior is due to the nature of the dataset.

Another interesting observation is that RNNs and their variants such as LSTM and GRU also show good performance because of their sequence learning properties. Note that the datasets DS-1, DS-2, DS-4, and DS-5 contain short length tweets, whereas DS-3 contains long texts of news comments. The overall best performance of BiCapsHate

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VI

COMPARISON IN TERMS OF TA AND VA ON BALANCED AND IMBALANCED DATASETS. THE BEST RESULTS OVER EACH DATASET ARE SHOWN IN BOLD

Datasets →	Balanced datasets										Imbalanced datasets									
	DS-1		DS-2		DS-3		DS-4		DS-5		DS-1		DS-2		DS-3		DS-4		DS-5	
Methods ↓	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA	TA	VA
BiCapsHate model	0.9492	**0.9045**	**0.8545**	0.5045	0.8332	***0.6949***	0.9249	***0.8045***	**0.9681**	**0.9189**	0.9523	***0.9285***	0.8943	0.8003	**0.9212**	0.7449	0.9226	0.8408	**0.9591**	**0.8852**
Ding et al. [22]	0.6122	0.6449	0.4943	0.4747	0.4912	0.5222	0.5739	0.5933	0.7884	0.7878	0.8009	0.7803	0.5721	0.5923	0.6933	0.6832	0.6035	0.5611	0.7923	0.8069
Mossie and Wang [33]	0.9215	0.8630	0.8232	0.4444	0.4943	0.5711	0.8946	0.7232	0.8712	0.8221	0.9309	0.8802	0.7921	0.7228	0.8504	0.6807	0.9007	0.7943	0.8787	0.7881
Roy et al. [11]	0.8831	0.6432	0.5008	0.4726	0.8122	0.6449	0.8432	0.6225	0.7212	0.6660	0.8931	0.7737	0.8321	0.7843	0.8809	0.6626	0.8943	0.7831	0.6103	0.5972
fBERT	**0.9715**	0.8935	0.8485	0.5111	**0.9004**	0.6773	**0.9463**	**0.8179**	0.9576	0.8675	0.9796	0.9281	0.9430	0.9232	0.9188	0.7449	**0.9803**	**0.9366**	0.9552	0.8778
HateBERT	0.9675	0.8887	0.8447	0.5024	0.8955	0.6901	0.9437	0.7884	0.9513	0.8749	**0.9807**	0.9142	0.9401	0.9210	0.9064	**0.7737**	0.9770	0.9353	0.9478	0.8591
ToxicBERT	0.9674	0.8938	0.7079	0.4843	0.8661	0.7036	0.9101	0.7936	0.9677	0.9143	0.9734	0.9132	**0.9626**	0.9235	0.9115	0.7666	0.9721	0.9337	0.9490	0.8696
Naïve Bayes	0.8712	0.8309	0.6464	0.4764	0.7698	0.6536	0.7781	0.6913	0.8757	0.8169	0.9223	0.8975	0.8464	0.7785	0.7658	0.7155	0.9111	0.8379	0.8675	0.8248
Decision Tree	0.9214	0.8901	0.7512	0.4923	0.8152	0.6324	0.8254	0.6712	0.8945	0.8424	0.9412	0.9101	0.7435	0.7588	0.8889	0.7387	0.9012	0.8345	0.8914	0.8545
Random Forest	0.9001	0.8725	0.8432	0.4998	0.8123	0.6545	0.9176	0.7015	0.9348	0.8458	0.9211	0.9101	0.8888	0.7997	0.9123	0.6725	0.8947	0.8201	0.9425	0.8545
DNN	0.8943	0.8543	0.6132	0.4846	0.7033	0.6531	0.8343	0.7607	0.8365	0.8146	0.9203	0.9104	0.8302	0.7543	0.7741	0.7037	0.9021	0.8137	0.8308	0.8291
CNN	0.8421	0.8725	0.6328	0.4731	0.6909	0.6312	0.8043	0.7547	0.8144	0.8126	0.8934	0.8831	0.8327	0.7621	0.7737	0.7005	0.8832	0.8147	0.8122	0.8279
LSTM	0.9139	0.8733	0.7112	0.4832	0.7533	0.6642	0.8809	0.7604	0.9152	0.8107	0.9321	0.8926	0.8527	0.7723	0.8343	0.7138	0.8922	0.8243	0.9132	0.8194
BiLSTM	0.9031	0.8834	0.8121	0.4706	0.7747	0.6746	0.8932	0.7643	0.9434	0.8085	0.9246	0.8843	0.8708	0.7832	0.8544	0.7025	0.9041	0.8329	0.9432	0.8180
GRU	0.9043	0.8635	0.7441	0.4849	0.7540	0.6743	0.8842	0.7626	0.9282	0.8107	0.9103	0.8744	0.8625	0.7527	0.8443	0.7144	0.9125	0.8243	0.9257	0.8217
BiGRU	0.9123	0.8822	0.8005	0.4813	0.7547	0.6549	0.9035	0.7644	0.9492	0.8093	0.9335	0.8824	0.8622	0.7743	0.8536	0.7123	0.9121	0.8124	0.9475	0.8241
BERT	0.9565	0.8852	0.8137	**0.5092**	0.8890	**0.7638**	0.9249	0.8031	0.9464	0.8702	0.9653	**0.9414**	0.9415	**0.9257**	0.8867	0.7112	0.9750	0.9361	0.9484	0.8581

TABLE VII

PERFORMANCE RESULTS FOR MULTICLASS CLASSIFICATION. BiCapsHate SHOWS THE BEST PERFORMANCE (BOLD)

Datasets ↓	TA	VA	P	R	F
BiCapsHate					
DS-1	**0.9644**	**0.9067**	**0.9671**	**0.9614**	**0.9642**
DS-2	**0.9128**	0.7887	**0.9171**	0.9067	**0.9116**
DS-5	**0.8753**	**0.8132**	**0.8931**	**0.8481**	**0.8679**
BERT					
DS-1	0.9555	0.9000	0.8033	0.9206	0.8571
DS-2	0.8463	**0.8058**	0.6516	**0.9286**	0.7655
DS-5	0.8172	0.6603	0.6362	0.6969	0.6636

across all these datasets in both balanced and imbalanced forms demonstrates its efficacy on both short and long texts for HS detection.

The transformer-based language models pretrained on offensive contents—fBERT, HateBERT, ToxicBERT—are recently developed highly effective state-of-the-art models, specific to HS detection. They require high-end hardware resources for computation. On the contrary, in addition to performing the best, BiCapsHate does not have such high requirements. It can be run on a machine without GPU.

E. Experimental Results for Multiclass Classification

We also evaluate the performance of BiCapsHate for multiclass classification of hate and related speeches over DS-1, DS-2, and DS-5 (other datasets are binary labeled). Table VII presents our multiclass classification results. In this set up, BiCapsHate outperforms BERT by significant margins.

F. Ablation Study

The main purpose of our ablation study is to examine the componentwise performance of BiCapsHate.

TABLE VIII

ABLATION STUDY OF BiCapsHate MODEL. BiCapsHate SHOWS ITS BEST PERFORMANCE (BOLD) WITH ALL THE LAYERS

Datasets →	Balanced datasets					Imbalanced datasets				
Evaluation metrics →	F-score					F-score				
BiCapsHate ↓	DS-1	DS-2	DS-3	DS-4	DS-5	DS-1	DS-2	DS-3	DS-4	DS-5
All layers	**0.9423**	**0.8222**	**0.8333**	**0.9214**	**0.9381**	**0.9251**	**0.5606**	**0.8502**	**0.7962**	**0.9231**
w/o capsule network	0.9312	0.8005	0.8023	0.9112	0.9271	0.9001	0.5449	0.7516	0.7742	0.9102
w/o auxiliary features	0.9212	0.8132	0.8143	0.9141	0.9008	0.9034	0.5324	0.7421	0.7414	0.8881
w/o BiLSTM	0.8938	0.7109	0.7603	0.8512	0.8714	0.8342	0.2124	0.5124	0.6426	0.8824
w/o attention	0.9103	0.7925	0.8009	0.9115	0.9002	0.8915	0.5307	0.8212	0.7543	0.7527

Table VIII shows the overall findings of this study on all balanced and imbalanced datasets in terms of *f-score*. Observe that BiCapsHate, in its complete form, consistently outperforms all the other alternatives with reduced components. Excluding the capsule networks degrades the *f-score* significantly. It demonstrates the critical role of capsule networks in learning the contextual information in different orientations of the input text. Excluding the auxiliary features degrades the *f-score*. It demonstrates the important role of our *well-informed* model by external knowledge including the Hatebase lexicon. Excluding the stacked BiLSTM network degrades the *f-score*. In line with the current state-of-the-art methods using sequence modeling networks such as RNN, LSTM, and GRU, stacked BiLSTM is obviously one of the most important components learning the contextual sequences in both forward and backward directions. Excluding the attention network degrades the *f-score*. It shows that attention is also another important component, highlighting the important HS-related words.

G. Effects of Parameters

This section presents the effects of parameters.

1) Number of BiLSTM Hidden Units: Choosing the total number of hidden units is important in terms of analyzing the classification performance of a deep learning-based model. While presenting the overall performance results earlier in

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

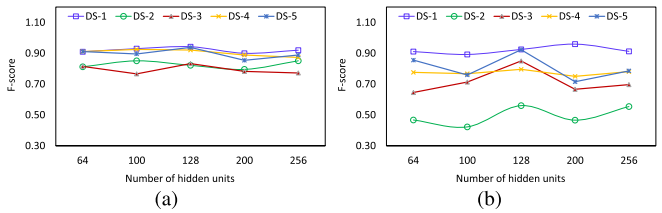

Fig. 3. Effects of number of BiLSTM hidden units. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

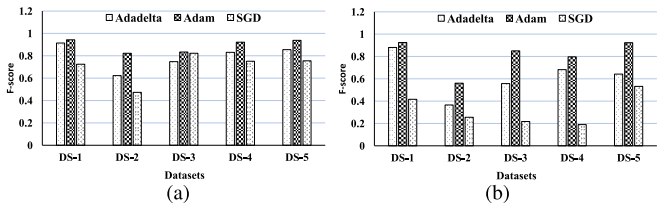

Fig. 4. Effects of optimization algorithms. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

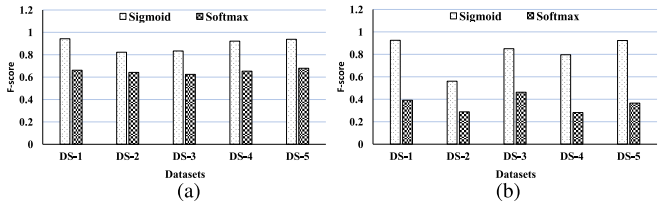

Fig. 5. Effects of activation functions. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

Section V-D, we considered a total of 128 BiLSTM hidden units. Fig. 3 shows the effects of varying the number of BiLSTM hidden units from 64 to 256 on our classification results on balanced and imbalanced datasets, respectively. Observe that 128 hidden units in BiLSTM perform remarkably better across all datasets.

2) Optimization Algorithms: The main function of optimization algorithm is to minimize the loss function in order to produce good results. Fig. 4 shows the effects of three different optimization algorithms (*Adadelta*, *Adam*, and *SGD*) on our classification results on balanced and imbalanced datasets, respectively. Observe that *Adam* is comparatively better than *Adadelta* and *SGD* on both balanced and imbalanced datasets. This is because of the bias-correction in *Adam* toward the end of optimization as gradients becomes sparser.

3) Activation Functions: The role of an activation function is to introduce nonlinearity into the output of a neuron and check whether a neuron should be activated or not. Fig. 5 shows the effect of two different activation functions (*sigmoid* and *softmax*) on our classification results on balanced and imbalanced datasets, respectively. Observe that *sigmoid* performs better as compared with *softmax* across all the datasets. This is because the proposed model works on a binary class problem.

4) Number of Routing Iterations in Capsule Networks: In capsule networks, the routing algorithm convergence is used to connect capsules in consecutive levels, where the

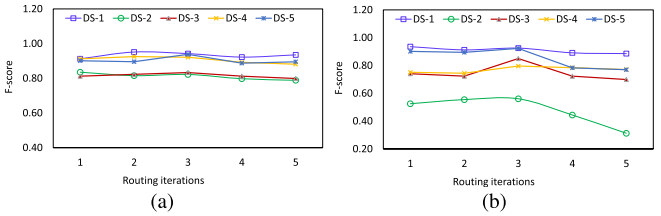

Fig. 6. Effects of number of routing iterations. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

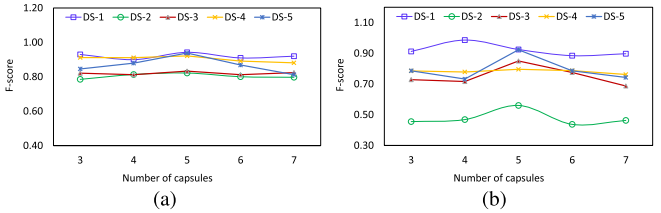

Fig. 7. Effects of number of capsules. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

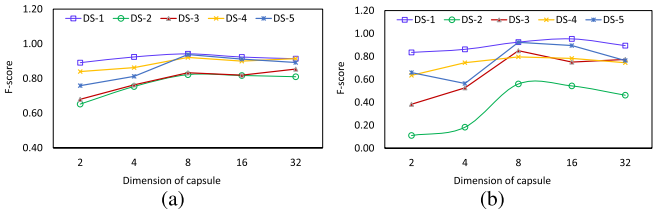

Fig. 8. Effects of dimension of capsules. (a) F-score on balanced datasets. (b) F-score on imbalanced datasets.

information is passed from the lower level capsule to the higher level capsule, if it agrees with its input. Fig. 6 shows the effects of the number of routing iterations from 1 to 5 on our classification results on balanced and imbalanced datasets, respectively. Observe that the most suitable number of routing iterations is 3.

5) Number of Capsules: A capsule network is formed by a number of capsules that help in encoding the spatial information. Fig. 7 shows the effects of varying the number of capsules from 3 to 7 on our classification results on the balanced and imbalanced datasets, respectively. Observe that with five capsules, the proposed model shows the optimal performance over the datasets.

6) Dimension of Capsules: The dimension of a capsule indicates the length of the output vector of a capsule. Fig. 8 shows the effects of varying the dimension of capsules from 2 to 32 on our classification results on the balanced and imbalanced datasets, respectively. Observe that 8 is the most suitable value for capsule dimension. It implies that both undersized and oversized dimensions of capsules may adversely affect the performance.

H. Qualitative and Error Analysis

We conduct a qualitative analysis of the proposed model by analyzing the predictions on some sample instances, few shown in Table IX. The first sample given in the table shows that the hateful content is correctly predicted as hateful by our model because it has explicit offensive words such as

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE IX
SAMPLE INSTANCES FOR ANALYSIS

Text	Actual label	Predicted label
Dear Utah Polynesians: Again, stop acting black and speaking ghetto. Speak FOB, you ain't no ghetto niggaz.	Hateful	Hateful
And now Jihadis bragging about a massive bombing in Sadr city in Baghdad	Normal	Normal
Man I swear today's like nation hit tony while he's driving day.... This chink almost smashed into me...	Hateful	Normal
@user @user lax gun laws? Ur retarded. There's murder there bcuz no one can defend themselves. Only crooks have guns there	Normal	Hateful

ghetto and niggaz, used in negative context. Also observe from the second sample that normal text may also contain some offensive words but used to present information rather than to demean others. The correct predictions of such context-dependent samples show that our model effectively captures the contextual usage of hateful and offensive words.

We also analyze some sample instances that are misclassified by the proposed model. The third row of Table IX has a sample *hateful* labeled tweet that our model classifies as *normal* by underestimating its hateful nature. This might be because it contains "chink," which is an offensive word but generally used in nonoffensive context. The last row has a sample *normal* tweet that the model misclassifies as *hateful*. It is probably because the user expresses strong emotional outburst containing hateful (often) words such as "gun" and "retarded."

I. Findings and Limitations

In this research, we developed `BiCapsHate`, which is an advanced deep learning model that generally outperforms the existing methods. It does not require high-end hardware resources for its computation. It can be executed on a CPU machine with even 8-GB RAM. On the contrary, state-of-the-art pretrained language models such as BERT, fBERT, HateBERT, and ToxicBERT are highly complicated and require atleast a GPU machine. Another interesting finding is that `BiCapsHate` shows good performance over both short and long texts. Moreover, it is effective for both binary and multiclass datasets.

Though `BiCapsHate` is highly effective for HS detection in English texts, it has some limitations as well. First, it has been evaluated only on English datasets. Its adaptation over multilingual or code-mixed data is a promising future research. Second, it is not suitable for HS detection from multimodal data. Lastly, as discussed in Section V-H, the proposed model does not always capture the offensive words with mild intensity of hate or low hateful tone.

VI. CONCLUSION

In this article, we proposed a novel deep learning model, called `BiCapsHate`, for HS detection on OSM platforms. The model consists of five advanced layers of DNNs, including the *input*, *embedding*, *BiCaps*, *dense*, and *output* layers. Each layer is dedicated to capture specific properties of HS. With the help of bidirectional capsule networks, the *BiCaps* layer learns the linguistic contextual information with respect to different orientations in both forward and backward directions of the input text. The rich set of hand-crafted shallow and deep auxiliary features including the `Hatebase` lexicon further enriches the layer. In our extensive experiments, the proposed model generally outperformed the existing state-of-the-art methods. It can be very useful to aid the content moderation systems of OSM platforms. Furthermore, unlike its competitors, it does not have a requirement of high-end hardware resources for computation. This research leads to the future research directions of generalization of BiCapsHate for other text classification tasks, particularly the ones that cannot afford high-end hardware resources. It is also worth investigating its adaptation for HS detection in multilingual social posts.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their valuable comments and suggestions.

REFERENCES

[1] L.-L. Shi, L. Liu, Y. Wu, L. Jiang, J. Panneerselvam, and R. Crole, "A social sensing model for event detection and user influence discovering in social media data streams," *IEEE Trans. Computat. Social Syst.*, vol. 7, no. 1, pp. 141–150, Feb. 2020.

[2] S. Ghosh and T. Anwar, "Depression intensity estimation via social media: A deep learning approach," *IEEE Trans. Computat. Social Syst.*, vol. 8, no. 6, pp. 1465–1474, Dec. 2021.

[3] M. Abulaish, A. Kamal, and M. J. Zaki, "A survey of figurative language and its computational detection in online social networks," *ACM Trans. Web*, vol. 14, no. 1, pp. 1–52, Feb. 2020.

[4] V. K. Sejwal, M. Abulaish, and Jahiruddin, "CRecSys: A context-based recommender system using collaborative filtering and LOD," *IEEE Access*, vol. 8, pp. 158432–158448, 2020.

[5] D. Borrelli, L. Iandoli, J. E. Ramirez-Marquez, and C. Lipizzi, "A quantitative and content-based approach for evaluating the impact of counter narratives on affective polarization in online discussions," *IEEE Trans. Computat. Social Syst.*, vol. 9, no. 3, pp. 1–12, Jun. 2021.

[6] S. Ullmann and M. Tomalin, "Quarantining online hate speech: Technical and ethical perspectives," *Ethics Inf. Technol.*, vol. 22, no. 1, pp. 69–80, Mar. 2020.

[7] S. Ghosh, A. Ekbal, P. Bhattacharyya, T. Saha, A. Kumar, and S. Srivastava, "SEHC: A benchmark setup to identify online hate speech in English," *IEEE Trans. Computat. Social Syst.*, early access, Mar. 21, 2022, doi: 10.1109/TCSS.2022.3157474.

[8] W. Warner and J. Hirschberg, "Detecting hate speech on the world wide web," in *Proc. 2nd Workshop Lang. Social Media*, 2012, pp. 19–26.

[9] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter," in *Proc. NAACL Student Res. Workshop*, 2016, pp. 88–93.

[10] E. W. Pamungkas, V. Basile, and V. Patti, "A joint learning approach with knowledge injection for zero-shot cross-lingual hate speech detection," *Inf. Process. Manag.*, vol. 58, no. 4, 2021, Art. no. 102544.

[11] P. K. Roy, A. K. Tripathy, T. K. Das, and X.-Z. Gao, "A framework for hate speech detection using deep convolutional neural network," *IEEE Access*, vol. 8, pp. 204951–204962, 2020.

[12] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. ICWSM*, 2017, pp. 512–515.

[13] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," *Appl. Intell.*, vol. 48, no. 12, pp. 4730–4742, 2018.

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[14] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on Twitter using a convolution-GRU based deep neural network," in *Proc. ESWC*, 2018, pp. 745–760.

[15] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. 26th Int. Conf. World Wide Web Companion (WWW Companion)*, 2017, pp. 759–760.

[16] G. Liu and J. Guo, "Bidirectional LSTM with attention mechanism and convolutional layer for text classification," *Neurocomputing*, vol. 337, pp. 325–338, 2019.

[17] J. Yang, K. Yu, and T. Huang, "Supervised translation-invariant sparse coding," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3517–3524.

[18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019, pp. 4171–4186.

[19] P. Fortuna, J. Soler-Company, and L. Wanner, "How well do hate speech, toxicity, abusive and offensive language classification models generalize across datasets?" *Inf. Process. Manag.*, vol. 58, no. 3, 2021, Art. no. 102524.

[20] J. Hoover et al., "Investigating the role of group-based morality in extreme behavioral expressions of prejudice," *Nature Commun.*, vol. 12, no. 1, pp. 1–13, 2021.

[21] J. Uyheng and K. M. Carley, "Bots and online hate during the COVID-19 pandemic: Case studies in the United States and the Philippines," *J. Comput. Social Sci.*, vol. 3, no. 2, pp. 445–468, Nov. 2020.

[22] Y. Ding, X. Zhou, and X. Zhang, "YNU_DYX at SemEval-2019 task 5: A stacked BiGRU model based on capsule network in detection of hate," in *Proc. 13th Int. Workshop Semantic Eval.*, 2019, pp. 535–539.

[23] P. Burnap and M. L. Williams, "Cyber hate speech on Twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy Internet*, vol. 7, no. 2, pp. 223–242, 2015.

[24] I. Alfina, R. Mulia, M. I. Fanany, and Y. Ekanata, "Hate speech detection in the Indonesian language: A dataset and preliminary study," in *Proc. Int. Conf. Adv. Comput. Sci. Inf. Syst. (ICACSIS)*, Oct. 2017, pp. 233–238.

[25] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate speech detection with comment embeddings," in *Proc. 24th Int. Conf. World Wide Web*, 2015, pp. 29–30.

[26] A. Kumar, S. Saumya, and J. P. Singh, "NITP-AI-NLP@HASOC-dravidian-CodeMix-FIRE2020: A machine learning approach to identify offensive languages from dravidian code-mixed text," in *Proc. Forum Inf. Retr. Eval.*, 2020, pp. 384–390.

[27] S. Malmasi and M. Zampieri, "Detecting hate speech in social media," in *Proc. RANLP*, 2017, pp. 467–472.

[28] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on Twitter," in *Proc. 1st Workshop Abusive Lang. Online*, 2017, pp. 41–45.

[29] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proc. 1st Workshop Abusive Lang. Online*, 2017, pp. 85–90.

[30] A.-M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis, "A unified deep learning architecture for abuse detection," in *Proc. 10th ACM Conf. Web Science*, 2019, pp. 105–114.

[31] P. K. Roy, S. Bhawal, and C. N. Subalalitha, "Hate speech and offensive language detection in dravidian languages using deep ensemble framework," *Comput. Speech Lang.*, vol. 75, Sep. 2022, Art. no. 101386.

[32] B. Wang and H. Ding, "YNU NLP at SemEval-2019 task 5: Attention and capsule ensemble for identifying hate speech," in *Proc. 13th Int. Workshop Semantic Eval.*, 2019, pp. 529–534.

[33] Z. Mossie and J.-H. Wang, "Vulnerable community identification using hate speech detection on social media," *Inf. Process. Manag.*, vol. 57, no. 3, 2020, Art. no. 102087.

[34] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[35] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[36] R. Cao, R. K.-W. Lee, and T.-A. Hoang, "Deephate: Hate speech detection via multi-faceted text representations," in *Proc. 12th ACM Conf. Web Sci.*, 2020, pp. 11–20.

[37] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with EM routing," in *Proc. ICLR*, 2018, pp. 44–51.

[38] D. Sarkar, M. Zampieri, T. Ranasinghe, and A. Ororbia, "FBERT: A neural transformer for identifying offensive content," in *Proc. EMNLP*. Punta Cana, Dominican Republic: ACL, 2021, pp. 1792–1798.

[39] T. Caselli, V. Basile, J. Mitrovic, and M. Granitzer, "HateBERT: Retraining BERT for abusive language detection in English," in *Proc. WOAH*. Punta Cana, Dominican Republic: ACL, 2021, pp. 17–25.

[40] L. Hanu and Unitary Team. (2020). *Detoxify*. Github. [Online]. Available: https://github.com/unitaryai/detoxify

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[43] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *Proc. Int. Conf. Artif. Neural Netw.*, 2011, pp. 44–51.

[44] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. NIPS*, 2017, pp. 3856–3866.

[45] J. Kim, S. Jang, and S. Choi, "Text classification using capsules," *Neurocomputing*, vol. 376, no. 1, pp. 214–221, Feb. 2020.

[46] D. K. Jain, R. Jain, Y. Upadhyay, A. Kathuria, and X. Lan, "Deep refinement: Capsule network with attention mechanism-based system for text classification," *Neural Comput. Appl.*, vol. 32, no. 7, pp. 1839–1856, Apr. 2020.

[47] M. Abulaish, M. Fazil, and M. J. Zaki, "Domain-specific keyword extraction using joint modeling of local and global contextual semantics," *ACM Trans. Knowl. Discovery from Data*, vol. 16, no. 4, pp. 1–30, Aug. 2022.

[48] Y. Du, X. Zhao, M. He, and W. Guo, "A novel capsule based hybrid neural network for sentiment classification," *IEEE Access*, vol. 7, pp. 39321–39328, 2019.

[49] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.

[50] A.-M. Founta et al., "Large scale crowdsourcing and characterization of Twitter abusive behavior," in *Proc. Int. AAAI Conf. Web Social Media*, Jun. 2018, vol. 12, no. 1, pp. 491–500.

[51] M. M. Bradley and P. J. Lang, "Affective norms for English words (ANEW): Stimuli, instruction manual, and affective ratings," The Center Res. Psychophysiology, Univ. Florida, Gainesville, FL, USA, Tech. Rep. C-1, 1999.

[52] L. Gao and R. Huang, "Detecting online hate speech using context aware models," in *Proc. RANLP Recent Adv. Natural Lang. Process. Meet Deep Learn.*, Nov. 2017, pp. 260–266.

[53] B. Mathew, P. Saha, S. M. Yimam, C. Biemann, P. Goyal, and A. Mukherjee, "Hatexplain: A benchmark dataset for explainable hate speech detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 17, 2021, pp. 14867–14875.

Authorized licensed use limited to: JAMIA MILLIA ISLAMIA UNIVERSITY- NEW DELHI. Downloaded on January 20,2023 at 08:28:32 UTC from IEEE Xplore. Restrictions apply.