

# Predicting Hate Intensity of Twitter Conversation Threads

Qing Meng<sup>1a</sup>, Tharun Suresh<sup>1b</sup>, Roy Ka-Wei Lee<sup>d</sup>, Tanmoy Chakraborty<sup>c</sup>

<sup>a</sup>*Hohai University, Nanjing, China*

<sup>b</sup>*IIT Delhi, Delhi, India*

<sup>c</sup>*Indian Institute of Technology Delhi, New Delhi, India*

<sup>d</sup>*Singapore University of Technology and Design, Singapore*

---

## Abstract

Tweets are the most concise form of communication in online social media. Wherein a single tweet has the potential to make or break the discourse of the conversation. Online hate speech is more accessible than ever, and stifling its propagation is of utmost importance for social media companies and users for congenial communication. Most of the research has focused on classifying an individual tweet regardless of the tweet thread/context leading up to that point. One of the classical approaches to curb hate speech is to adopt a *reactive strategy* after the hateful content has been published. This strategy results in neglecting subtle posts that do not show the potential to instigate hate speech on their own but may portend in the subsequent discussion ensuing in the post's replies. In this paper, we propose DRAGNET++, which aims to predict the intensity of hatred that a tweet can bring in through its reply chain in the future. Our model uses the semantic and propagating structure of the tweet threads to maximize the contextual information leading up to and the fall of hate intensity at each subsequent tweet. We explore three publicly available Twitter datasets – *Anti-Racism* contains the reply tweets of a collection of social media discourse on racist remarks during US political and COVID-19 background; *Anti-Social* presents a dataset of 40 million tweets amidst the COVID-19 pandemic on anti-social behaviours with custom annotations; and *Anti-Asian* presents Twitter datasets collated based on anti-Asian behaviours during COVID-19 pandemic. All the curated datasets consist of structural graph information of the Tweet threads.

---

<sup>1</sup>Both authors contributed equally to this research.

We show that DRAGNET++ outperforms all the state-of-the-art baselines significantly. It beats the best baseline by an 11% margin on the Person correlation coefficient and a decrease of 25% on RMSE for the Anti-Racism dataset with a similar performance on the other two datasets.

*Keywords:* Hate intensity prediction, Twitter reply chain, online social media, hate speech, graph neural network

*PACS:* 0000, 1111

*2000 MSC:* 0000, 1111

---

## 1. Introduction

**Motivation.** The proliferation of social media has enabled users to share and spread ideas at a prodigious rate. While the information exchanges in social media platforms may improve an individual’s social connectedness with online and offline communities, these platforms are increasingly plagued with the rampant onslaught of provocative and toxic content. One such highly toxic content is ‘hate speech’, defined by the Cambridge dictionary as “*public speech that expresses hate or encourages violence towards a person or group based on race, religion, sex, or sexual orientation .*” Hate speech on social media has created dissension among online communities and culminated in offline violent hate crimes [1]. Therefore, addressing the spread of hate speech on social media is critical.

Major social media platforms such as Facebook and Twitter have made significant efforts to combat the spread of hate speech on their platforms [2, 3]. For example, the platforms have established clear policies regarding hateful conduct [4, 5], implemented mechanisms for users to report hate speech, and employed content moderators to detect hate speech. However, such approaches are labor-intensive, time-consuming, and thus not scalable or sustainable in the long run [6, 7]. Traditional machine learning and deep learning methods have also been proposed to automatically detect hate speech in online social media [8, 9, 10]. However, most of the existing methods are limited to classifying hate speech at the individual post level, ignoring the network and propagation effects of hateful content on social media [11].

Ideally, social media content moderators would want to identify hateful posts and monitor posts and threads more likely to incite hatred. Consider the example of tweet propagation shown in Figure 1. The initial source tweet is a benign tweet that reports the Russian invasion of Ukraine. How-

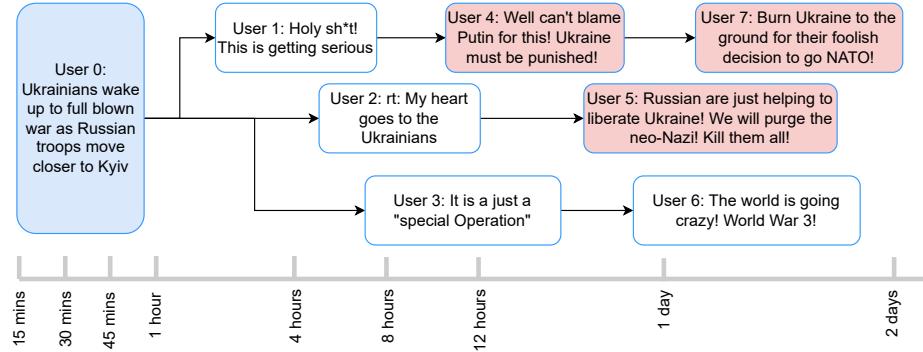


Figure 1: An example of a tweet propagation. The hateful tweets are highlighted in red.

ever, as time evolves, we observe hateful content justifying the Russian invasion and promoting violence against the Ukrainians. Existing automated hate speech detection methods could help content moderators detect hateful tweets. Nonetheless, content moderators could significantly improve the effectiveness and efficiency of moderation by prioritizing posts likely to generate a large number of hateful tweets in their retweet or reply threads. This a challenging problem as the post that induces hateful content may not be hateful itself (e.g., source tweet in Figure 1). Early detection of tweets likely to generate a large amount of hateful content becomes a critical real-world issue.

Very few studies examined the hate intensity of Twitter conversation threads for early hate speech detection. Lin et al. [12] manually categorized Twitter conversation threads into different levels of hatred and proposed a deep learning model to forecast and classify tweets into the pre-defined hatred levels. Recently, Sahnan et al. [13] introduced the hate intensity prediction task and proposed DRAGNET, a deep stratified learning framework that predicted the intensity of hatred that a root tweet can fetch through its subsequent replies. Nevertheless, DRAGNET assumes a linear sequence for the conversation thread in its model, in which tweets in conversation threads are arranged in a chronological sequence. This neglects the tree structural information inherently present in a conversation thread; there could be multiple branches in a conversation thread similar to the example shown in Fig. 1. The structural information may be an important feature that could better inform and improve hate intensity prediction. For instance, we may notice a heated debate along the branches of the conversation threads; therefore, modeling the dynamics of the tree structure of the conversation thread may

aid us in better predicting the eventual hate intensity of the conversation thread.

**Research Objectives.** In this paper, we aim to fill this research gap and extend our earlier work [13] by proposing DRAGNET++ that models multifaceted information to forecast the hate intensity of a Twitter conversation thread. Specifically, we define our problem statement as follows – *given a root tweet and a few of its initial replies, can we predict the hate intensity of the subsequent replies of the tweet?* To model this problem, we first quantify the hate intensities of tweets in the conversation thread and express them as a series of hate intensities, transforming the task into a time-series problem. At a high level, DRAGNET++ adopts a similar stratified learning framework as suggested in [13] to model the hate intensity profiles of Twitter conversation threads and categorize them into clusters of varying hate intensity. In addition to the content sentiment and temporal information, DRAGNET++ also captures structural information of the conversation thread. Specifically, DRAGNET++ adopts Graph Neural Networks (GNN) to learn the semantic and propagation structure of conversation threads.

As the prediction of hate intensity in conversation threads is a relatively new research problem, we conduct thorough experiments and analyses to evaluate DRAGNET++. We collate three publicly available real-world datasets – *Anti-Racism* with 3.5k root tweets, *Anti-Social* with 668k root tweets, and *Anti-Asian* with 218k root tweets. We extensively analyze these datasets to examine the hate intensities of real-world Twitter conversation threads. We benchmark DRAGNET++ against DRAGNET and six other baselines on the hate intensity task. Finally, we examine case studies and conduct ablation studies to explain the advantages and limitations of DRAGNET++.

**Contributions.** We summarize our contributions below:

- We analyze the hate intensity in three large-scale real-world Twitter datasets. This is the first large-scale study that examines hate intensity in Twitter conversation threads.
- We propose DRAGNET++, a stratified learning framework with structural graph augmentation to perform hate intensity prediction.
- We conduct extensive experiments and show that DRAGNET++ consistently outperforms state-of-the-art methods in the early prediction

of hateful conversation threads. Specifically, DRAGNET++ outperforms the best baseline by at least 34% on Pearson Correlation Coefficient (PCC), reduction of at least 200% on Root Mean Square Error (RMSE), and 7% lower on Mean Forecast Error (MFE) across the datasets. Comparing the performance of DRAGNET++ with DRAGNET shows an average improvement of 8.33% on PCC and an average reduction of 16.07%, and 18.20% on RMSE and MFE, respectively.

- Our case studies demonstrate the viability of predicting hate speech early to prevent the propagation of hateful content.

**Organization of the paper:** We start by presenting the developments until recently related to hate speech detection and time series forecasting in Section 2. Following this, we define the formulation of hate intensity profiles appropriate to our problem statement in Section 3. We then move on to detailing our model in Section 4. Further, we analyze the various datasets and present their statistics and observations in Section 5. We discuss a brief overview of various baselines in Section 6.1. Section 6.2 presents a detailed comparison, analysis of the model performance, and the effect of various parameters used to fine-tune the model. We conclude the paper in Section 7.

**Reproducibility.** The source codes of DRAGNET++ and all the baselines and the dataset are available at the following link: <https://github.com/LCS2-IIITD/Predicting-Hate-Intensity>.

## 2. Related Work

### 2.1. Causes of Hate Speech

There are several reasons for individuals to harbor hatred towards each other or a particular group of people. According to Navarro’s [14] discussion, hate can arise when someone perpetrates harm or discrimination against others. Hate is often characterized by the devaluation of the victim, which can escalate to the point of elimination in extreme cases. This notion demonstrates the snowball effect commonly associated with hatred as arguments progress. In the context of the internet, John et al. [15] highlight that people tend to self-disclose more frequently or intensely in online media, a phenomenon known as the online disinhibition effect. Furthermore, Lindsay et al. [16] argue that social media, a highly interconnected platform,

is driven by algorithms and primarily profits from user engagement. As a result, it unintentionally propagates extreme content under the guise of serving user interests. The causes of hate speech are multifaceted, and some of them are introduced due to technological advancements. While we do not explicitly study the cause of hate speech in individual tweets, our proposed model possesses the capability to identify "high-risk" tweets and reply chains, alerting content moderators to conduct further analysis and gain a deeper understanding of the causes of hate speech.

## 2.2. Hate Speech Detection in Social Media

Industry and academics have paid close attention to the work on hate speech detection. With the advancement of deep learning, neural language processing systems that can automatically extract features from the text have seen a lot of success [17, 18, 19]. This opens up new possibilities for hate speech detection [20, 21, 22, 23, 24, 25]. Gamback et al. [7], for example, used Convolutional Neural Networks (CNN) to classify hate speech by extracting word similarities. Similar research was done by Park et al. [26], who presented the HybridCNN model to investigate word and character combination patterns to detect hate speech. On the other hand, Del et al. [27] used Long Short Term Memory (LSTM) to record the long-term dependencies of words in phrases to differentiate hatred remarks. Bajjatiya et al. [21] evaluated the use of the LSTM model in conjunction with the Gradient-Boost Decision Tree (GBDT) to perform hate speech classification and found that it greatly enhanced performance. Zhang et al. [22] presented a CNN+GRU network architecture to investigate word dependency for recognizing hate speech tweets, combining the benefits of CNN-based models and Gate Recurrent Unit (GRU)-based models. However, most research focuses on learning a particular textual property while ignoring other valuable data. Cao et al. [28] introduced DeepHate, a deep learning-based hate speech detection algorithm for mining multi-faceted textual representations. Lee et al. [29] later developed DisMultiHate, a hateful meme classification model that can learn both textual and visual information. Recently, Awal et al. [25] proposed a meta-learning-based framework (HateMAML) that can effectively detect hate speech in eight different low-resource languages.

Pre-trained language models such as BERT [30] and GPT [31] can extract external information from massive volumes of text data. These models have also been used in hate speech detection models with promising results [32, 33]. For example, Awal et al. [33] used the pre-trained BERT model as a

shared layer and created AngryBERT, a multi-task learning model that can jointly detect hate speech and classify sentiment. Nonetheless, most previous research works have focused on categorizing hate speech. A few research looked into how hate speech spreads through social media. In a recent discussion, Dahiya et al. [34] discussed the necessity of anticipating the hatred intensity of tweets. Lin et al. [12] classified the tweets into different levels of hatred and suggested that the HEAR model tracks posts likely to cause hate speech. Closer to our work is DRAGNET [13], which is a deep stratified learning framework that predicts the hate intensity of a conversation thread based on what a root tweet can fetch through its subsequent replies. DRAGNET models the linear sequence for the conversation thread chain, where the tweets are arranged chronologically. Such a modeling approach ignores the conversation thread’s inherent structural information propagation information. We address this limitation and propose DRAGNET++which considers the Twitter conversation thread’s structural information to improve hate intensity prediction.

On the other hand, some comprehensive datasets are proposed in hate speech shared tasks. They focus on niche aspects of hate speech. For example, Basile et al. [35] released a task focusing on hate speech against immigrants and women in a multilingual setup. Sanguinetti et al. [36] formulated testing out of domain dataset between tweets and news headlines. They explored the prevalence of verbless fragments in most hate speech texts. Furthermore, potential hate speech spreaders can be inferred from an individual’s Twitter feed [37]. Narrowing a step deeper, Pavlopoulos et al. [38] discussed toxic span detection from previously annotated hateful comments by re-annotating them at the span level. They gave a more fine-grained understanding of spans contributing to toxicity.

### 2.3. Time Series Forecasting Modeling

The hatred intensity prediction problem can be reduced to a time series prediction task, where we forecast the hate intensity of the conversation thread in the future. Time series models are designed to predict values or trends over time in the future and have been extensively studied in many fields such as transportation[39], finance[40], event forecasting, [41] and disease transmission [42]. Therefore, we also introduce the existing studies of time series forecasting (TSF) models.

Traditional TSF methods, such as ARMA [43], exponential smoothing [44], and linear space models, have been the basis of much work and achieved

better performance. With the rapid development of deep learning technology in recent years, more and more end-to-end TSF models have been proposed. Compared to traditional methods, deep learning-based models such as CNN, RNN, and LSTM can automatically extract features from the input without domain expertise, widely accepted in many fields. For example, Oord et al. [45] adopted CNN in raw audio generation and proposed the WaveNet model. Based on WaveNet, Borovskykh et al. [46] utilized CNN to perform conditional time series forecasting tasks. The model shared filter weights assuming the hidden patterns are time-invariant at each step. In addition, the RNN contains an internal memory state, which can also retain information about previous time steps [47]. However, due to vanishing gradients, the performance of classical RNN-based models degrades as the sequence length increases. Later, the LSTM model overcomes the long-term dependency problem to some extent. Elsworth et al. [48] used LSTM for TSF tasks, improving performance and robustness.

However, most of the above models only focus on one-step predictions. Some recent studies [49, 50] generalize sequence-to-sequence (Seq2Seq) models and propose them for multi-step time series forecasting methods. Fan et al. [50] exploited attention mechanisms to capture the temporal context information extracted by the RNN encoder. Combined with a bidirectional LSTM decoder, the model can generate multiple future horizons simultaneously. Later, researchers proposed new architectures [51, 52] to reduce error accumulation in multi-step prediction. For instance, Sen et al. [51] combined global matrix factorization models and local temporal networks to capture latent patterns in time series. In recent years, the Transformer architecture has been proposed for natural language processing tasks and has also achieved success on time series data [52, 53]. Moreover, some studies [54, 55] proposed models to estimate the probability distribution of future time series. For example, Yuan et al. [54] and Koonchli et al. [56] employed Generative Adversarial Networks (GANs) to predict future values. Salinas et al. [55] proposed a DeepAR model for probabilistic prediction based on auto-regressive recurrent networks.

### 3. Preliminaries

**Hate Intensity Definition.** Table 1 summarizes the denotations of the important notations. Let an ordered sequence of first  $t$  number of replies to a root tweet  $\varphi$  be  $\mathcal{T}_{1,t}^{\varphi} = \langle c_1^{\varphi}, c_2^{\varphi}, \dots, c_t^{\varphi} \rangle$ , where  $c_i^{\varphi}$  refers to the  $i^{th}$  reply.

Table 1: Important notations and denotations.

Symbol	Definition
$\varphi$	Root tweet
$c_i^\varphi$	$i^{th}$ reply to root tweet $\varphi$
$R_{p,q}$	Hate value set of $q$ reply tweets of the $p^{th}$ root tweet
$R_{p,q}^*$	The set of hate values for the $q$ reply tweets of the $p^{th}$ root tweet predicted by the autoencoder
$S_{s(1,n)}$	Cosine similarity set between root tweet and $n$ reply tweets
$R_{s(1,n)}$	Set of hate intensity profiles
$j$	Number of clusters
$\delta$	Window size
$t_h$	Number of replies in history
$t_f$	Index of the last reply tweet in the conversation thread
$n$	Maximum length of the conversation thread
$s$	Total number of conversation threads
$N_{X_h}$	Dimension of encoded history latent vector
$N_{X_f}$	Dimension of encoded future latent vector
$\mathcal{X}$	Latent vector of $R_{p,q}$
$\mathcal{X}^*$	Latent vector of $R_{p,q}^*$
$C_c$	List of cluster centers
$\mathcal{P}(C_{ci})$	Likelihood of belongingness to $i^{th}$ cluster identified with $C_{ci}$
$\mathcal{P}^*(C_{ci})$	Predicted weight for $i^{th}$ cluster centre $C_{ci}$ to calculate $\mathcal{X}_c$
$\mathcal{X}_d$	Pre-processed prior vector
$\mathcal{X}_h, \mathcal{X}_f$	The representation of historical and future conversation threads
$\mathcal{X}_h^c, \mathcal{X}_f^c$	The representation of pseudo historical and pseudo future conversation threads
$\mathcal{E}_h(\cdot)$	The encoder model of historical conversation threads
$\mathcal{E}_f(\cdot)$	The encoder model of future conversation threads
$\mathcal{GM}(\cdot)$	Fuzzy Clustering
$\mathcal{PR}(\cdot)$	Prior model
$\mathcal{FP}(\cdot)$	Future Predictor
$\mathcal{FP}_d(\cdot)$	1 <sup>st</sup> segment of Future Predictor
$\mathcal{FP}_p(\cdot)$	2 <sup>nd</sup> Future Predictor
$\mathcal{D}(\cdot)$	Decoder
$\mathcal{TE}(\cdot)$	Tree Encoder

As the replies in the original dataset have no ground-truth hate intensity, we quantify the hate intensity of each reply using a weighted sum of two measures as suggested in [34]:

$$\mathcal{H} = w\mathcal{H}_c(c) + (1 - w)\mathcal{H}_l(c), \quad (1)$$

where  $\mathcal{H}_c$  indicates the probability that the reply is hateful, and it is calculated by a state-of-the-art hate speech detection model (we will elaborate it in Section 6.2).  $\mathcal{H}_l$  is the average score for all words in a reply from a model-independent hate lexicon that comprises 2,895 words as proposed in [57].  $w \in [0, 1]$  is a hyper-parameter that balances two quantities of the hate intensity score. As  $\mathcal{H}_c \in [0, 1]$  and  $\mathcal{H}_l \in [0, 1]$ , the final hate intensity score of a reply  $\mathcal{H}$  is still between 0 and 1. We do not filter any tweet threads based on their hate intensity. Therefore, each conversation thread  $\mathcal{T}$  can be mapped to a sequence of hate intensity scores,

$$\mathcal{H}(\mathcal{T}_{1,t}^\varphi) = <\mathcal{H}(c_1^\varphi), \mathcal{H}(c_2^\varphi), \dots, \mathcal{H}(c_t^\varphi)> \quad (2)$$

To avoid noise and drastic fluctuations that are not in line with reality, we further smooth the hate intensity sequence by utilizing a *rolling average operation* with window size  $\delta$  [34]. A window consists of  $\delta$  consecutive replies, and the window that starts from the  $k^{th}$  reply is denoted by  $\mathcal{T}_{k,k+\delta}^\varphi$ . Finally, the hate intensity of a window for the tweet  $\varphi$  is measured as,

$$\mathcal{H}(\mathcal{T}_{k,k+\delta}^\varphi) = \sum_{c \in \mathcal{T}_{k,k+\delta}^\varphi} \mathcal{H}(c) = w \sum_{c \in \mathcal{T}_{k,k+\delta}^\varphi} \mathcal{H}_c(c) + (1 - w) \sum_{c \in \mathcal{T}_{k,k+\delta}^\varphi} \mathcal{H}_l(c), \quad (3)$$

where  $\mathcal{H}(\mathcal{T}_{k,k+\delta}^\varphi) \in [0, \delta]$ , and  $\delta$  is the window size.

**Sentiment Features.** The emotional feedback of users is reflected in the sentiment features of reply posts, which provide both for and against arguments of the original tweet. Therefore, we use the cosine similarity between the sentiment embedding of the root tweet ( $\varphi$ ) and its accompanying replies ( $c_1^\varphi, c_2^\varphi, \dots$ ) to capture the sentiment context of a conversation thread, denoted as  $\mathcal{CS}(c_i) = \text{CosineSim}(\text{Embed}(c_i^\varphi), \text{Embed}(\varphi))$ . The sentiment embedding is the second last fully-connected layer from the pre-trained *XLNet* model [58] for sentiment classification. To smooth the value and eliminate the effect of noise, we also apply the *rolling average operation* to the sentiment context sequences  $\mathcal{CS}(\mathcal{T}_{1,t}^\varphi)$  with the same window size  $\delta$  as performed on  $\mathcal{H}(\mathcal{T}_{1,t}^\varphi)$ .

**Problem Definition.** Given (i) a root retweet  $\varphi$ , (ii) its last  $t_h$  historical replies  $\mathcal{T}_{1,t_h}^\varphi = \{c_1^\varphi, c_2^\varphi, \dots, c_{t_h}^\varphi\}$ , (iii) the corresponding hate intensity sequence  $\{\mathcal{H}(\mathcal{T}_{k,k+\delta}^\varphi) | k \in [1, 2, \dots, t_h - \delta]\}$ , and (iv)  $G_\varphi = (V, E)$  referring to the adjacency matrix of the propagation tree, where  $V = \{\varphi, c_1^\varphi, c_2^\varphi, \dots, c_t^\varphi\}$ , and  $e_{ij} \in \{0, 1\}$  denote the retweeting/reply relationships between tweets

(i.e., root tweet or replies), we aim to predict the hate intensity of the upcoming replies  $c_{t'}^\varphi$  in the propagation tree of the root tweet  $\varphi$ . However, in corresponding to the historical hate intensity sequence, we consider predicting the hate intensity for each window of  $c_{t'}^\varphi$ , denoted by  $\mathcal{H}(\mathcal{T}_{t', t'+\delta}^\varphi)$ , instead of directly predicting the hate intensity of each reply.

#### 4. Methodology

This section presents our proposed hate intensity prediction method, DRAGNET++. It is a deep stratified learning [59] method that splits heterogeneous data points (in this case, Twitter conversations) into homogeneous clusters/strata before training a deep regressor on each stratum to predict hate intensity.

Figure 2 illustrates the overall architecture of DRAGNET++. The two-dimensional vector forms the training set of window-wise hate intensity profile and sentiment context value sequences.

$$\begin{aligned}\mathcal{R}_{s_{(1,n)}} &= \{\mathcal{R}_{(p,q)} : 1 \leq p \leq s, 1 \leq q \leq n\} \\ \mathcal{R}_{(p,q)} &= \{\mathcal{H}(\mathcal{T}_{k,k+\delta}^{\varphi_p}) : 1 \leq k \leq q - \delta\} \\ \mathcal{S}_{s_{(1,n)}} &= \{\mathcal{S}_{(p,q)} : 1 \leq p \leq s, 1 \leq q \leq n\} \\ \mathcal{S}_{(p,q)} &= \{\mathcal{CS}(\mathcal{T}_{k,k+\delta}^{\varphi_p}) : 1 \leq k \leq q - \delta\}\end{aligned}\tag{4}$$

where  $s$  represents the total number of conversations, and  $n$  represents the maximum conversation length. The elements  $\mathcal{R}_{(p,q)} \in \mathcal{R}_{s_{(1,n)}}$  and  $\mathcal{S}_{(p,q)} \in \mathcal{S}_{s_{(1,n)}}$  represent the  $p^{th}$  data point, where the conversation thread is of length  $q$  ( $\varphi_p$  represents the  $p^{th}$  root tweet).

DRAGNET++ uses an autoencoder to learn low-dimensional latent representations for the hate intensity profile of conversation threads. Specifically, the model learns two alternative latent representations:  $\mathcal{X}_h$ , which represents the first few replies, and  $\mathcal{X}_f$ , which represents the future hatred trend for the remaining replies. Once these representations,  $\mathcal{X}_h$  and  $\mathcal{X}_f$ , are combined, the model uses an unsupervised setting to apply a fuzzy clustering technique to give cluster membership probabilities and cluster centers to each conversation thread. The number of clusters is determined by a hyper-parameter  $j$ . Following this, the model trains with historical node and propagation structure  $G_\varphi^{(1,t_h)}$  using the tree encoder to learn the structure embedding  $\mathcal{X}_{hs}$ . The model then trains a new deep neural network unit to predict cluster membership probabilities given  $\mathcal{X}_h$ ,  $\mathcal{X}_{hs}$  and  $\mathcal{S}_{s_{(1,t_h)}}$ , which assign cluster centers

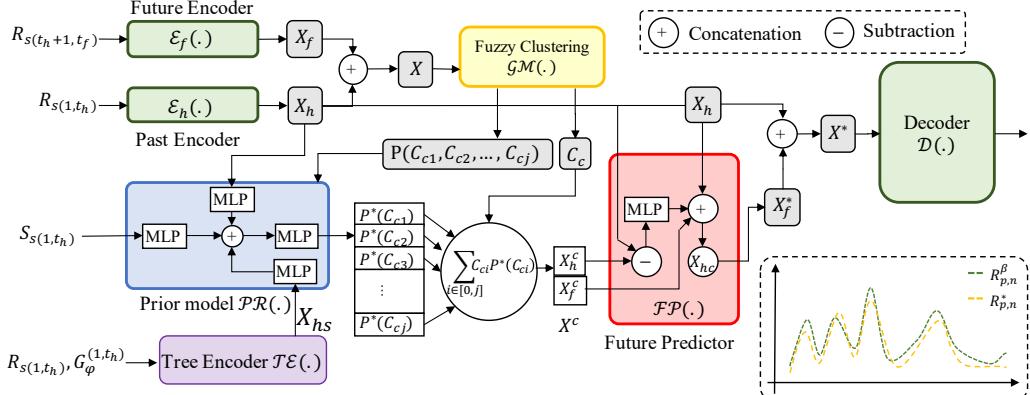


Figure 2: The overall framework of DRAGNET++. The autoencoder is trained on hate-intensity profiles of the entire conversation thread. Using the trained autoencoder, the history and future latent representations are concatenated and clustered using the fuzzy clustering algorithm  $\mathcal{GM}(\cdot)$ . In the Prior model,  $\mathcal{PR}(\cdot)$ , graph representations, history latent representations and sentiment features are concatenated to generate the Prior Knowledge vector. On inference, (a) the history latent representation, (b) sentiment similarity features of the history, and (c) the graph representation of the conversation thread are used to predict the future hate intensity profile.

for a new conversation thread. Finally, using  $\mathcal{X}_h$  and  $\mathcal{P}(C_{c1}, C_{c2}, \dots, C_{cj})$ , a novel deep regressor predicts the latent representation of the future hate trend  $\mathcal{X}_f^*$ , which, when coupled with  $\mathcal{X}_h$ , is transformed to the whole hate trend by the decoder trained during the autoencoder phase.

Figure 3 illustrates an example processing of an abstract tweet thread using our model. Four representations are derived from the input tweet thread: History, Future, Sentiment and Graph representations. The history and future representations are fed into the Fuzzy Clustering algorithm to predict the cluster centers. In parallel, a Prior model is trained on history, sentiment and graph representations forming the prior knowledge. This is combined with cluster centers to learn/assign the cluster membership probabilities in the fuzzy associations step. Further, these probabilities are combined with history representations in the Future Predictor to predict the latent representation of future hate trends. In the final step, the future latent representation is concatenated with the history representation and fed into the decoder to predict the overall hate trend of the entire conversation thread.

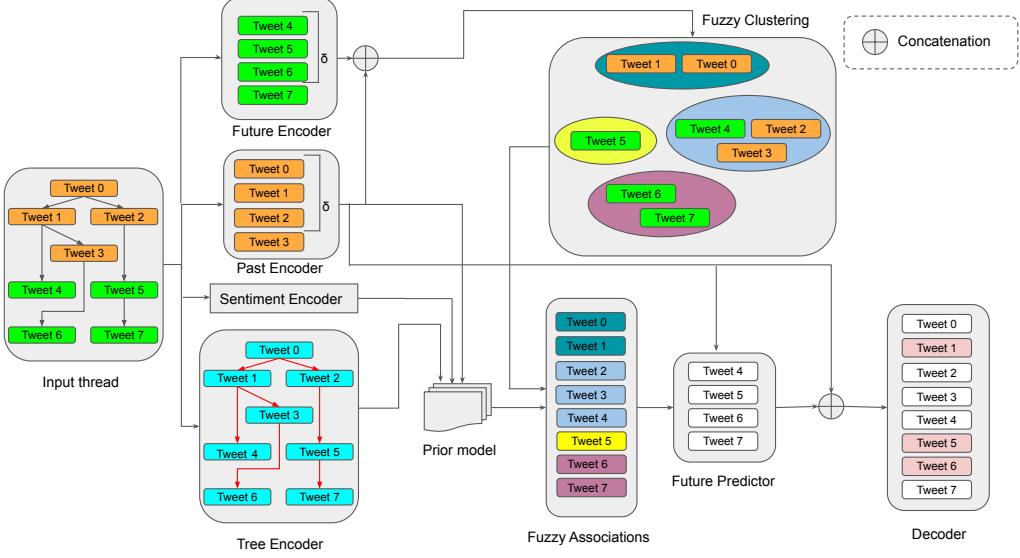


Figure 3: An example processing of an abstract tweet thread using our model. We show the various stages and representations learned by our model and how they combine to profile the final hate intensity of future replies. We differentiate the earlier and future tweets in the input threads using orange and green colors.  $\delta$  is the size of the sliding window of tweets. The tree encoder shows differences between the node and edge representations. Fuzzy clustering and fuzzy associations share a common color code to show the assignment of cluster memberships. The final decoder block represents some tweets being assigned high hate intensity scores (i.e., tweets in red)

#### 4.1. Time-series Representative Learning

The vector of the conversation thread  $\mathcal{R}_{s_{(1,n)}}$  can be viewed as a collection of irregularly lengthening time series (window-wise hatred intensity profiles). The Dynamic Time Warping (DTW) distance metric and its derivatives are used to group comparable trends together in state-of-the-art approaches for clustering irregular time series [60]. DTW's precision in mapping time series similarity is beneficial, but the noisy and volatile character of the data points in the present research prevents it from showing high efficiency in clustering comparable hatred patterns into a single stratum. We propose an autoencoder to translate each conversation thread  $\mathcal{R}_{(p,q)}$  in  $\mathcal{R}_{s_{(1,n)}}$  to a low-dimensional latent representation in order to capture a more suitable representation of the time series. We also propose a multi-encoder strategy instead of a single encoder-decoder design as proposed in [61].

## 4.2. Proposed Autoencoder

To capture the hate intensity series, the autoencoder module seeks to learn a low-dimensional representation. In DRAGNET++, we specifically construct two encoders and a single decoder. Using the two encoders, the model can learn the representations of historical and future hate intensity sequences separately. The decoder then uses the representations to reconstruct the original hate intensity sequence.

### 4.2.1. Encoder

Several studies provide models for univariate and multivariate [62, 63] time series to mine the hidden patterns of various sequences. We use a state-of-the-art Inception-Time [63] module to automatically extract features and represent hate intensity sequences, denoted as:

$$\mathcal{X}_m = \mathcal{E}_t(\mathcal{R}_{s_{(1,n)}}), \quad (5)$$

where  $\mathcal{X}_m \in \mathbb{R}^{s \times n \times 4}$  is the multivariate intermediate representation of  $\mathcal{R}_{s_{(1,n)}}$ . The Inception-Time module is denoted by  $\mathcal{E}_t(\cdot)$ . After the Inception-Time module, we flatten  $\mathcal{X}_m$  and use multilayer perceptrons as the classification stage. Finally, the representations are written down as follows:

$$\mathcal{X}_o = \mathcal{E}_{lt}(flatten(\mathcal{X}_m)), \quad (6)$$

Here,  $\mathcal{X}_m$  is transformed into a one-dimensional vector using  $flatten(\cdot)$ .  $\mathcal{E}_{lt}(\cdot)$  denotes multilayer perceptrons that have been trained to learn the best representation of the hate intensity sequence.

We develop the history encoder ( $\mathcal{E}_h$ ) and the future encoder ( $\mathcal{E}_f$ ) based on Equations (5) and (6) to encode hate intensity sequences of both historical and future conversation threads:

$$\begin{aligned} \mathcal{X}_h &= \mathcal{E}_h(\mathcal{R}_{s_{(1,t_h)}}) \\ \mathcal{X}_f &= \mathcal{E}_f(\mathcal{R}_{s_{(t_h+1,t_f)}}) \end{aligned} \quad (7)$$

where  $\mathcal{X}_h \in \mathbb{R}^{s \times N_{\mathcal{X}_h}}$  and  $\mathcal{X}_f \in \mathbb{R}^{s \times N_{\mathcal{X}_f}}$  denote the historical and future latent representations, respectively.  $N_{\mathcal{X}_h}$  and  $N_{\mathcal{X}_f}$  represent the length of  $\mathcal{X}_h$  and  $\mathcal{X}_f$ , respectively.

#### 4.2.2. Decoder

Although we use two encoders to convert the hate intensity of each conversation thread into two latent representations,  $\mathcal{X}_h$  and  $\mathcal{X}_f$ , we only employ one decoder,  $\mathcal{D}(\cdot)$  to return the latent representation to the original input. The decoder is trained to reconstruct the original hate intensity profile per conversation thread by concatenating the two latent representations. The decoder’s operation can be summarised as follows:

$$\mathcal{R}_{s_{(1,n)}}^* = \mathcal{D}([\mathcal{X}_h \oplus \mathcal{X}_f]), \quad (8)$$

where  $\mathcal{R}_{s_{(1,n)}}^*$  is the reconstructed hate intensity sequence containing both historical and future hate intensity sequences.

#### 4.3. Fuzzy Associations

The hate intensity profiles of conversation threads in our dataset are noisy, volatile, and lack a discernible pattern, as mentioned in Section 4.1. Our goal is to group similar profiles using low-dimensional latent representations of data obtained by autoencoder (as explained in Section 4.2). Recent research that supports this technique attests to the validity of deep learning-based models’ effectiveness in learning hidden features from time-series data for various applications [64, 65]. We apply a clustering strategy over the latent representations to aggregate heterogeneous hate intensity profiles into (near-) homogeneous clusters. Finding meaningful correlations in data is unduly dependent on the number of clusters  $j$  and the cluster centers in this unsupervised context. As a result, rather than restricting each profile to a single cluster, we apply a fuzzy clustering approach and use the membership probabilities as a feature embedding.

We define the combined latent space  $\mathcal{X}$  as,

$$\mathcal{X} = \mathcal{X}_h \oplus \mathcal{X}_f. \quad (9)$$

Rather than using a hard clustering strategy, in which each profile’s association to the nearest cluster is fixed, we use cluster membership probability using a fuzzy clustering approach instead. The membership probability vector, denoted by  $P(C_{c1}, C_{c2}, \dots, C_{cj})$ , reflects the associative probabilities of each cluster with the provided chain, where  $C_{ci}$  denotes the cluster center of the  $i^{th}$  cluster.

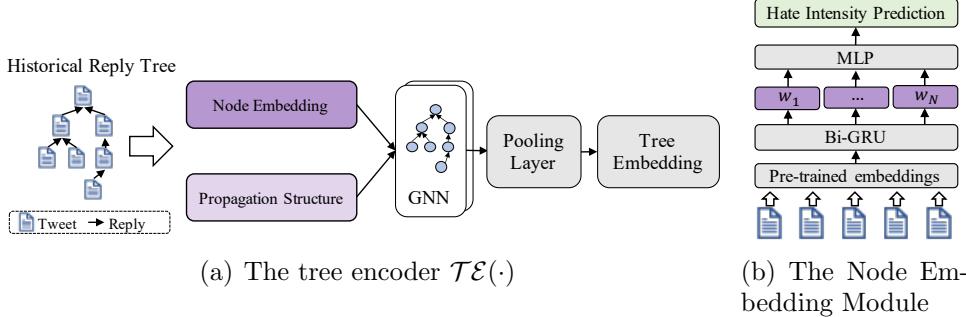


Figure 4: The tree encoder  $\mathcal{T}\mathcal{E}(\cdot)$  for generating graph-level representations for each conversation thread.

#### 4.3.1. Fuzzy Clustering

We cluster on the combined latent representation  $\mathcal{X}$  because our goal is to uncover correlations between each hate intensity profile and homogeneity groupings. To locate the clusters  $C_c$ , which is the collection of cluster centers, we use a state-of-the-art fuzzy clustering model [66], indicated by  $\mathcal{GM}(\cdot)$ :

$$C_c = \mathcal{GM}(\mathcal{X}) = (C_{c1}, C_{c2}, \dots, C_{cj}). \quad (10)$$

where  $j$  is the pre-defined number of clusters, and  $C_{ci}$  is the cluster centre of the  $i^{th}$  cluster.

#### 4.4. Tree Encoder

Twitter conversation threads typically contain numerous replies (i.e., tweets) with various propagation topological structures. These propagation structures capture the relationships between the tweets; modeling the topological structures would enrich the conversation thread’s representation for downstream machine learning tasks [67, 68]. For our hate intensity prediction task, we design a tree encoder  $\mathcal{T}\mathcal{E}(\cdot)$  to fuse the information of tweet content and the structural information to represent the conversation tree. The tree encoding process is presented in Fig 4(a). Specifically, to learn a conversation thread’s representation, we first learn the node embedding by designing a sub-module using a hate intensity prediction task to learn the tweet’s representation. Next, we leverage a Graph Neural Network (GNN) [69] to learn the conversation tree embedding by fusing the conversation tree structural information and the node embedding.

**Learning Node Embedding.** We adopt the tweet-level hate intensity prediction task as the supervised signal for learning the embedding of the

node (i.e., tweet). We first leverage pre-trained Word2Vec [70] word embedding to represent the words in individual tweets. Next, we use a two-layer bidirectional gated recurrent unit (BiGRU) to learn the individual post’s representation, and the tweet-level hate scores are used as training signals. Specifically, the post representations are fed into a two-layer MLP classifier to predict the post’s hate label. The model is formulated as follows:

$$\begin{aligned}\mathbf{w}_i &= Bi\text{-}GRU(< w_0, w_1, \dots, w_N >), \\ \hat{\mathbf{p}}_k &= Att(< w_0, w_1, \dots, w_N >), \\ \hat{y}_k &= MLP(\hat{\mathbf{p}}_k),\end{aligned}\tag{11}$$

where  $\hat{y}_k$  is the predicted hate score for the  $k$  tweet. We can train the model’s parameters with the loss function below and the Adam optimizer. We can obtain the tweet-level embedding  $\hat{\mathbf{p}}_k$ . The loss function ( $\mathcal{L}_h$ ) is defined as follows:

$$\mathcal{L}_h = \sum_{k=1}^N (y_k - \hat{y}_k)^2 + reg(\Theta)\tag{12}$$

where  $N$  is the number of tweets in all conversation threads,  $reg(\cdot)$  is a regularization function to alleviate overfitting, and  $\Theta$  is a parameter for the fine-tuning model.

**Learning Tree Embedding.** The goal of learning the tree embedding of the conversation thread is to leverage useful propagation structural information to perform early hate intensity prediction. Working towards this goal, we leverage GNNs to model the tree structure in the conversation threads. Specifically, GNNs model the conversation thread structural information as a directed graph with the node represented using the node embeddings learned from the previous section:

$$\mathbf{P}^{(l)} = \sigma(\mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2} \mathbf{P}^{(l-1)} \mathbf{W}^{(l-1)})\tag{13}$$

where  $\mathbf{W}$  is the trainable parameter,  $\mathbf{P}^{(l-1)} \in \mathbb{R}^{n \times d}$  is the representation of all tweets,  $\mathbf{P}^0$  is initialized by all tweet-level embeddings  $\hat{\mathbf{p}}_k, k \in \{1, 2, \dots, t_h\}$  learned using formula (11),  $\mathbf{A}$  and  $\mathbf{D}$  are the adjacency matrix and degree matrix of the conversation thread tree structure, respectively. Finally, using the average pooling operation, we can obtain the graph-level embedding of the reply tree, denoted  $\mathbf{X}_{hs}$ .

#### 4.5. Boosting Prediction with Prior Knowledge

The task of predicting the hate intensity of upcoming replies, provided limited history  $\mathcal{R}_{s(1,t_h)}^*$ , is strenuous even for state-of-the-art deep learning models due to the noisy, volatile, and heterogeneous nature of the time-series hate intensity profiles. To address this, we introduce the notion of prior knowledge to the prediction component of our pipeline as the weighted sum of the cluster centers, where the weights correspond to the cluster membership probabilities for the new chain, denoted by  $\mathcal{P}^*(C_{c1}, C_{c2}, \dots, C_{cj})$ . We define prior knowledge as follows:

$$\mathcal{X}^c = \sum_{i \in (0 \leq i \leq j)} C_{ci} P^*(C_{ci}) \quad (14)$$

Note that  $C_c$  is calculated over  $\mathcal{X}$ , i.e., the combined latent representation. Therefore, to calculate the complete membership probability vector for a new chain, we cannot directly use the fuzzy clustering model  $\mathcal{GM}(\cdot)$ . Instead, we construct a prior model  $\mathcal{PR}(\cdot)$  to predict the membership probabilities for new chains using only the latent representation of the history  $\mathcal{X}_h$ , the sentiment feature  $\mathcal{S}_{s(1,t_h)}$ , the historical replies  $\mathcal{R}_{s(1,t_h)}$  and propagation tree structure  $G_\varphi^{(1,t_h)}$ .

$$\mathcal{PR}(\mathcal{E}_h(\mathcal{R}_{s(1,t_h)}), \mathcal{S}_{s(1,t_h)}, \mathcal{T}\mathcal{E}(\mathcal{R}_{s(1,t_h)}, G_\varphi^{(1,t_h)})) = \mathcal{P}^*(C_{c1}, C_{c2}, \dots, C_{cj}) \quad (15)$$

The precision of the predictions by the prior regression model is measured by comparing  $\mathcal{P}^*(C_{c1}, C_{c2}, \dots, C_{cj})$  against  $\mathcal{P}(C_{c1}, C_{c2}, \dots, C_{cj})$ .

##### 4.5.1. Estimating Latent Representation of Upcoming Conversation Threads

The designed decoder needs the latent representations of historical and future conversation threads to reconstruct the complete hate intensity profile. However, our primary objective is to predict the future hate intensity profile based on historical information. To provide the decoder with the ability to perform prediction, we propose a future representation predictor that uses historical information to estimate the latent representation of future hate intensity profile  $\mathcal{X}_f$ . Then, combined with the latent representation of historical conversation threads, the output can be fed into the decoder.

Specifically, we utilize the prior knowledge extracted by the fuzzy clustering module and the latent representation  $\mathcal{X}_c$  of the historical conversation

threads. To avoid the estimation problem being unduly influenced by prior knowledge, we design the predictor in two steps. In the first step, for each conversation thread, the prior latent representation of historical conversation threads  $X_h^c$  contains the required information to predict the latent representation of future conversation threads. Moreover, we have the expected latent representation  $\mathcal{X}_h$  encoded from the initial historical replay chains. As a result, we use the difference operator on the expected ( $\mathcal{X}_h$ ) and estimated priors ( $\mathcal{X}_h^c$ ) of the historical conversation threads to evaluate the deviation of these two representations. Secondly, a single-layer perceptron  $\mathcal{FP}_d(\cdot)$  is used to learn the representation ( $X_d$ ) in a hidden space, indicating the dissimilarity between the prior knowledge and the history conversation threads.

$$\begin{aligned}\mathcal{X}_s &= \mathcal{X}_h \ominus \mathcal{X}_h^c \\ \mathcal{X}_d &= \mathcal{FP}_d(\mathcal{X}_s)\end{aligned}\tag{16}$$

We finally obtain the  $X_{hc}$  vector by concatenating the provided input  $X_h$ , the pre-processed prior  $X_d$  and  $X_f^c$  as,  $X_{hc} = X_h \oplus X_d \oplus X_f^c$ . The second stage is the deep linear transformation model  $\mathcal{FP}_p(\cdot)$  that predicts the upcoming hate intensity in the latent space  $X_f^*$  as follows:

$$\mathcal{X}_f^* = \mathcal{FP}_p(\mathcal{X}_{hc})\tag{17}$$

#### 4.6. Decoding the Future

We use the decoder module trained in Section 4.2.2 to forecast entire hate intensity profiles (i.e.,  $\mathcal{R}_{s(1,n)}^*$ ) based on the expected latent representation of the upcoming hate intensity  $\mathcal{X}_f^*$ . In particular, we concatenate the original latent representation ( $\mathcal{X}_h$ ) of historical hate intensity sequences with the expected future representation ( $\mathcal{X}_f^*$ ), denoted as:

$$\mathcal{X}^* = \mathcal{X}_h \oplus \mathcal{X}_f^*\tag{18}$$

where  $\mathcal{X}^*$  is the predicted hate intensity profile of the upcoming conversation thread in the latent space. The decoder is designed to be a mirror of the encoder. The decoder is defined as follows:

$$\mathcal{R}_{s(1,n)}^* = \mathcal{D}(\mathcal{X}^*)\tag{19}$$

where  $\mathcal{R}_{s(1,n)}^*$  denotes the expected hate intensity profile for the next future conversation thread. Finally, we compare the predicted and original hate intensity score sequences (i.e.,  $\mathcal{R}_{s(1,n)}$ ) to evaluate and report the performance on several metrics.

#### 4.7. Comparison with DRAGNET

From the modeling standpoint, DRAGNET++ has advanced DRAGNET [13] on two aspects: DRAGNET++ exploits the tweet-level semantics and the conversation-level structural information to improve hate intensity prediction. DRAGNET leveraged sentiment features, calculated as each tweet’s similarity in the conversation threads to the root tweet. However, this approach largely ignores the complex relationships among the tweets in the conversation threads. Furthermore, as illustrated in our earlier example in Fig. 1, the root tweet might be benign, and anchoring the sentiment features computation base on the root tweet may not provide an accurate forecast of the sentiment of the subsequent tweets in the conversation threads. Therefore, DRAGNET++ addressed this limitation by modeling the tweets’ semantics with an additional hate intensity prediction task at the tweet level and the conversation thread structure with GNNs as the tree encoder. The intuition is that by learning the node (i.e., tweets) and tree (i.e., conversation thread) representations using GNNs, we are able to capture the dependency among the tweets in the conversation threads and extract unique structural patterns that could improve the prediction of hate intensity for conversation thread.

## 5. Datasets

We evaluate DRAGNET++ on three publicly available large Twitter datasets. These datasets contain a large amount of Twitter conversation threads, which were originally collected for other hate speech-related studies. Table 2 shows the statistics of the datasets.

Dataset	#Conversation threads	Conversation thread length			#Tweets	#Unique users
		Min.	Max.	Avg.		
Anti-Racism	3,500	1	582	200	750,235	620,437
Anti-Social	668,082	1	20014	31	40,385,257	4,980,160
Anti-Asian	218,790	1	2822	35	206,348,565	23,895,911

Table 2: The statistics of the three datasets used in our experiments.

In the *Anti-Racism* dataset [13], the authors manually identified various real-world events using a hashtag-based matching via the Twitter API. During the tumultuous year of 2020, many topics polarised the discussion, such as the 2020 US Presidential election, the Brexit referendum in the UK, and

extending similar political issues across the US, the UK, and India. Adding to the diversity of the dataset, *sinophobic* tweets attributing coronavirus to China are also curated in the dataset, with most mentions (in tweets) as “China virus”. The final dataset comprises 3500 Twitter conversation threads with over 750K tweets.

In the *Anti-Social* dataset [71], the COVID-19-related tweets were collected from the Twitter platform. The authors performed query searches using case-sensitive keywords such as “covid-19”, “COVID-19”, “Coronavirus”, “coronavirus” and “corona”. Leveraging the Twitter Streaming API, the authors collected the conversations related to these queried tweets. Using this approach, the authors collected over 650K Twitter conversations with over 40M tweets published between March 17 and 2020 to April 28, 2020.

In the *Anti-Asian* dataset [72], the authors focused on the Asian hate speech surrounding the COVID-19 discussions. They followed a similar keyword-based approach but as a two-step process. The *covid-19 keywords* are used to narrow the COVID-19-related tweets, following which *hate keywords* about anti-Asian hate is used to segregate the tweets as defined by the collection for the dataset. Finally, *counter-speech keywords* are keywords and hashtags used to counter hate speech and support Asians. It comprises of 42 keywords. The authors used a combination of Twitter Streaming API and Twitter Search API to collect real-time tweets between January 15, 2020 and March 26, 2021. The dataset comprises over 210K Twitter conversation threads with more than 206M tweets.

As mentioned in Section 3, our main task is building hate-intensity profiles for each dataset’s tweets. However, it is important to note that we do not use the class labels in the datasets for our task as they do not contain ground truth for the hate intensity of tweets. Instead, we will be using the tweet text to derive the hate intensity scores (as discussed in Section 3).

**Length of Conversation Threads.** Figure 5(a) shows the length of conversation threads over the number of conversation threads for the *Anti-Racism* dataset. The average conversation thread length is around 200, with the maximum length being 582. On the other hand, Figures 5(b) and 5(c) show a clipped version of the entire graph for better visualization of the distribution. Most of the tweets constitute lengths less than or equal to 100. To understand the full picture, the numerical statistics of *Anti-Social* and *Anti-Asian* are represented in Table 2. The maximum length in the *Anti-Social* dataset goes as high as 20014. However, the average length of the dataset is

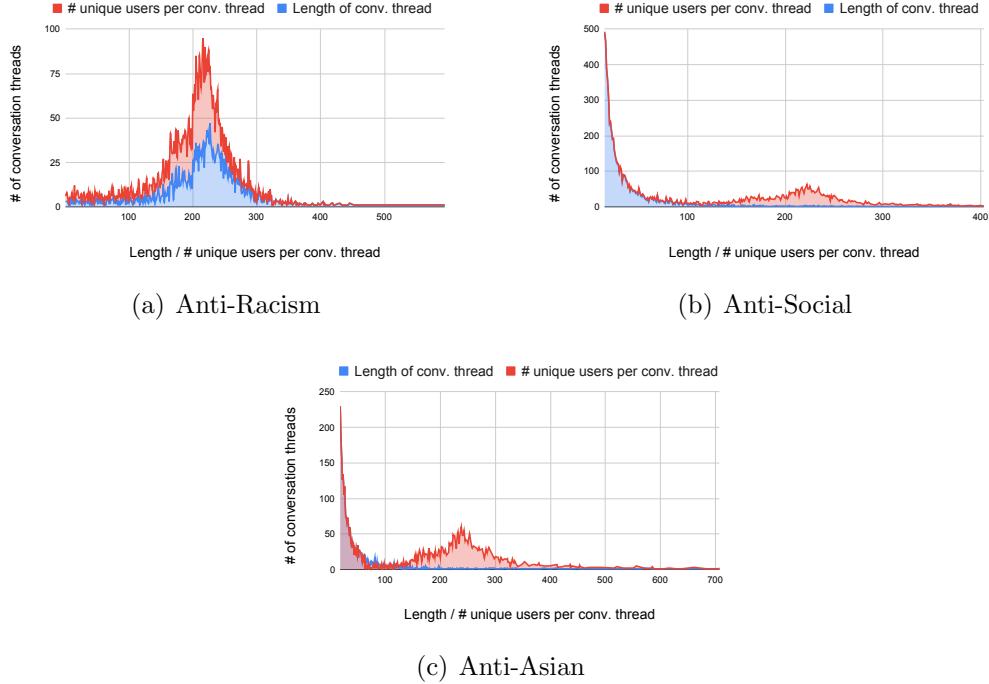


Figure 5: Distribution of number of conversation threads vs length of each conversation thread and the number of unique users in respective conversation threads for all three datasets. Note that for the *Anti-Social* and *Anti-Asian* datasets, the x-axis ticks start from 10 and ends earlier than the maximum length as described in Table 2 to show the distribution more clearly.

around 31. Along similar lines, the *Anti-Asian* dataset also touches a maximum conversation thread length of 2822 with an average length of 35 across the tweet threads.

**Number of Unique Users.** Figure 5 also highlights the number of unique users per reply thread. For the *Anti-Racism* dataset, it can be observed that the unique users and the lengths follow a similar distribution. The increased number of unique users can be attributed to limited re-engagement by the same users for a particular thread. Furthermore, both the *Anti-Social* and *Anti-Asian* datasets exhibit a high variation in the length of the threads with a consistent presence of threads in almost all lengths. Correspondingly, the unique users touch about 65 for about 225 threads in the case of *Anti-Social*, whereas 70 unique users for about 240 threads in *Anti-Asian*.

**Number of Conversation threads and Tweets.** The conversation threads and the total number of tweets are presented in Table 2. Compared to the *Anti-Racism* dataset, the other two datasets possess a vast number of conversation threads, forming the ideal datasets to test the performance of the models. However, the number of tweets is  $206M$  for *Anti-Asian*, whose conversation threads are less in number than *Anti-Social*, which has about  $40M$  tweets in total. This shows the variation in conversation thread length for different datasets and the corresponding unique users overall, adding diversity to the datasets.

## 6. Experiments

### 6.1. Baselines

There are few studies that predict hate intensity profiles in Twitter conversations [13, 34]. As a result, we use time-series forecasting and temporal pattern model based on both conventional and deep learning as baselines.

- **LSTM [73]:** Long short-term memory is a neural network architecture with feedback connections that is very effective. LSTMs, unlike RNNs, perform better on long-range predictions, such as time-series problems. As suggested by Elsworth and Güttel [73], we employ a stacked LSTM. We use ReLU as the activation function for each layer to forecast the hate intensity profile.
- **CNN [74]:** A convolutional neural network considers the hate intensities’ time information to better profile the tweet’s upcoming replies. We employ a 1-D CNN architecture with ReLU activation as the last layer. The kernel size is 2, and the number of filters employed is 64.
- **N-Beats [75]:** Neural Basis Expansion Analysis for interpretable Time Series forecasting is a deep learning model designed to tackle the univariate time-series problem. With a very deep stack of fully-connected layers, it incorporates forward and backward residual linkages.
- **DeepAR [55]:** It is a supervised learning approach for forecasting scalar time-series using Recurrent Neural Networks (RNN), which is achieved by auto-regressively training the RNN on numerous related time-series data.

- **TFT** [76]: Temporal Fusion Transformers are used in conjunction with multi-horizon forecasting to help the model mix known future inputs and extract exogenous correlations from the time series’ past input data. The model employs a self-attention-based architecture to give interpretable time-series insights.
- **ForGAN** [77]: Probabilistic forecasting of sensory data using generative adversarial networks employs an adversarial network to learn data generating distributions and compute probabilistic forecasts over them. In a time-series forecasting problem, the model claims to learn the conditional probability distribution of future values with no quantile crossing or reliance on the prior distribution.
- **DRAGNET** [13]: It is a deep stratified learning approach. To create the latent representation of past and future tweets from the specified point of reference as hyperparameter, an autoencoder is utilized using the Inception-Time module [63]. This is combined with a fuzzy clustering algorithm to forecast the future hate intensity sequence, which is reinforced with sentiment features to determine the correlation of each tweet with the root tweet.

## 6.2. Experimental Setup

The hyperparameters used for all the experiments are as follows:  $\delta = 10$ ,  $w = 0.6$ ,  $t_h = 25$ ,  $t_f = 275$ ,  $n = 300$ ,  $j = 15$ ,  $N_{X_h} = 32$ ,  $N_{X_f} = 128$ . In the auto-encoder step, DRAGNET++ is implemented with the Inception-Time module [63] for transformation with variable kernel sizes – 5, 7, and 9. Sentiment characteristics, history encodings from the auto-encoder, and graph structure information are fed into fully-connected layers in the prior model. A Gaussian mixture model with full covariance is utilized for fuzzy clustering. **Davidson model** [78] is the default hate-speech classifier. We utilize the *Adam* optimizer with an  $lr = 0.001$  for faster convergence, and the training and test split is set at 80 : 20. The train-test split is based on the number of conversation threads since we would like to train and test on complete tweet propagation.

Our model predicts the future hate intensity sequence, given the tweets and historical hate intensity sequence for the chosen window length. To evaluate, three metrics for this task are used: (i) **Pearson correlation coefficient** (PCC), where a higher value is better, (ii) **Root Mean Square**

**Error** (RMSE), and (iii) **Mean Forecast Error** (MFE), where lower values are better.

### 6.3. Experimental Results and Analysis

The overall performance of DRAGNET++ and baselines on the three datasets are shown in Table 3. Across all datasets, we find that DRAGNET++ outperforms the baselines. Specifically, for the *Anti-Racism* dataset, DRAGNET++ outperforms the best baseline (N-Beats) by 40% on PCC. Other assessment parameters show a similar trend, with a 100% reduction in RMSE and a 9 $\times$  reduction in MFE compared to N-Beats. On the contrary, as compared to the baselines on the *Anti-Asian* dataset, both DRAGNET and DRAGNET++ perform poorly on PCC. One possible explanation is the dataset’s restricted chain of responses. DRAGNET++ on the other hand, outperforms N-Beats by more than 3 $\times$  on RMSE when optimizing the inaccuracies on hate intensity profiles. TFT has a higher MFE score than DRAGNET, but DRAGNET++ has the highest total score. Along with LSTM and CNN, ForGAN appears to be among the worst performers. On the *Anti-Social* dataset, DRAGNET++ outperforms the best baseline, TFT, by 0.223, 0.476, and 0.202, respectively, on PCC, RMSE, and MFE. When DRAGNET and DRAGNET++ are directly compared, the latter performs better on all three datasets and on all three metrics. This demonstrates how graph information inherent in tweet threads can help forecast hate intensity profiles more accurately. The greatest significant improvement over RMSE is 22.72% on the *Anti-Social* dataset, and the highest increase on PCC is 0.066 points on the *Anti-Racism* dataset. On the MFE (*Anti-Racism* dataset), DRAGNET performs the best, which can be attributed to customized windowing strategies that reduce forecast error. DRAGNET’s improved performance does not apply to other datasets, where DRAGNET++ readily outperforms.

### 6.4. Ablation Study

The prior model  $\mathcal{PR}(\cdot)$  in DRAGNET++ uses the propagation graph structure, sentiment information, and fine-tuning operation to improve the representation of historical discussion threads and establish a link between historical and future data. To assess the performance of each component, we remove it one at a time and provide three variants: DRAGNET++ w/o graph structure, DRAGNET++ w/o sentiment, and DRAGNET++ w/o fine-tune. Then, using three datasets, we run experiments to see how the

Model	Anti-Racism			Anti-Social			Anti-Asian		
	PCC	RMSE	MFE	PCC	RMSE	MFE	PCC	RMSE	MFE
LSTM	0.145	0.611	0.500	0.160	0.511	0.315	0.680	0.722	0.692
CNN	0.105	0.644	0.509	0.112	0.542	0.320	0.675	0.731	0.699
DeepAR	0.310	0.484	0.065	0.180	0.490	0.275	0.682	0.748	0.708
TFT	0.469	0.437	0.076	0.376	0.630	0.333	0.562	0.866	0.125
N-Beats	0.380	0.544	0.085	0.340	0.633	0.271	<b>0.712</b>	0.462	0.173
ForGAN	0.240	0.603	0.360	0.172	0.897	0.785	0.563	0.871	0.475
DRAGNET	0.563	0.247	0.010	0.559	0.189	0.156	0.603	0.165	0.134
DRAGNET++	<b>0.629</b>	<b>0.218</b>	<b>0.008</b>	<b>0.599</b>	<b>0.154</b>	<b>0.131</b>	0.641	<b>0.147</b>	<b>0.116</b>

Table 3: Overall performance of DRAGNET++ and baselines on the three datasets. The best results are **bold**.

alternative models perform in the end, and the findings are displayed in Fig 6.

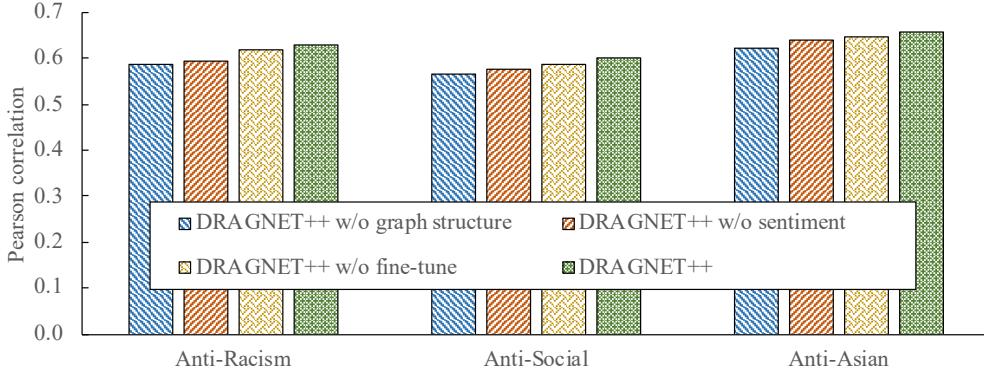


Figure 6: Ablation study: The Effectiveness of components in prior model  $\mathcal{PR}(\cdot)$  on three datasets.

The model’s performance degrades for each of the versions, as seen in Fig 6. It indicates that each component contributes differently to the prediction of conversation thread labels, resulting in improved hate intensity prediction results. Across all three datasets, DRAGNET++ w/o graph structure performs the worst among the four variations. The diffusion patterns and informative material are present in the propagation graph structures of conversation threads; therefore, this is reasonable. Capturing earlier responses and projecting future patterns would help the data. When comparing DRAGNET++ w/o sentiment to DRAGNET++, the performance drops slightly, but not as much as when comparing DRAGNET++ w/o graph structure. A possible reason could be that the tweet’s sentiment is assessed in relation to the root tweet. On the other hand, structural graph information

is a more advanced method of recording hate intensity distribution. Furthermore, DRAGNET++ outperforms DRAGNET++ w/o fine-tuning. It suggests that using the fine-tune technique, word embedding can precisely capture hate-intensity information.

### 6.5. Parameter Analysis

We further analyze the parameters of DRAGNET++ to understand its superiority and limitations better. Throughout this study, we compare DRAGNET++ and DRAGNET.

#### 6.5.1. The impact of different window sizes

The window size utilised in the average rolling process is represented by the hyperparameter  $\delta$ . It is possible that the smaller window size would result in more pronounced hate intensity profiles. We chose the window size in  $\{5, 10, 15, 20\}$  to analyse the influence of different window sizes, and the experimental findings are displayed in Fig 7. Across all three datasets, we observe that increasing the value of  $\delta$  improves the performance of both DRAGNET++ and DRAGNET. A bigger window size ( $\delta$ ) smooths the values in hate intensity sequences, making it easier for the models to learn the hidden hate intensity pattern. Furthermore, DRAGNET++ consistently outperforms DRAGNET across all window size settings, demonstrating its efficacy.

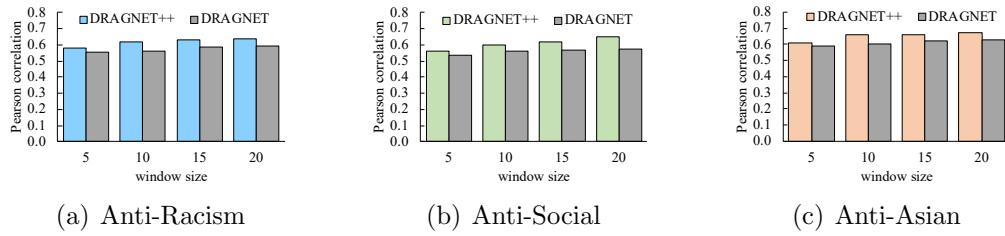


Figure 7: Performance comparison based on window size ( $\delta$ ) of hate intensity profiles.

#### 6.5.2. The impact of history sizes

The beginning history size  $t_h$  is the data needed to estimate the whole hate intensity profile for a new conversation thread. The value of the Pearson correlation coefficient for DRAGNET++ shows relatively little change as  $t_h$  increases, as seen in Fig 8. We can observe that DRAGNET++ outperforms

DRAGNET. The model’s capacity to generate early predictions decreases as the number of initial responses submitted to the model grows. As a result, in the experiments, we used  $t_h=25$  to predict hate intensity.

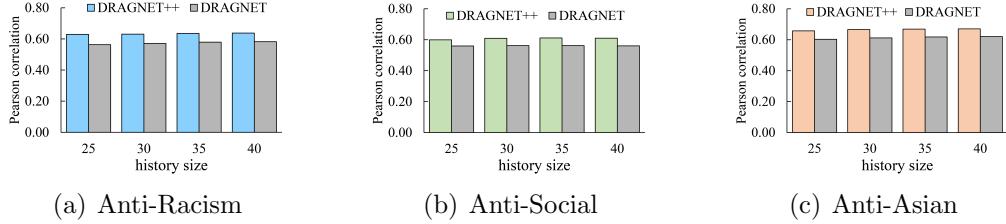


Figure 8: Performance of DRAGNET++ on three datasets based on changes to tweet history length at input.

### 6.5.3. The impact of different numbers of clusters

One of the most important hyperparameters of DRAGNET++ in the fuzzy clustering step is the number of clusters  $j$ . As a result, we choose a number of clusters in the range of  $\{5, 10, 15, 20\}$  to assess the impact of cluster size. The experiment outcomes are shown in Fig 9. We notice that as the number of clusters grows, the DRAGNET++’s performance improves, and it is consistently better than DRAGNET. It is difficult to discern between different types of conversation threads when we use a small number of clusters. However, the performance begins to deteriorate beyond a certain number of clusters. One probable explanation is that grouping conversation threads into too many clusters produces noise, making it harder for the prior model to detect the correct cluster labels with minimal historical data.

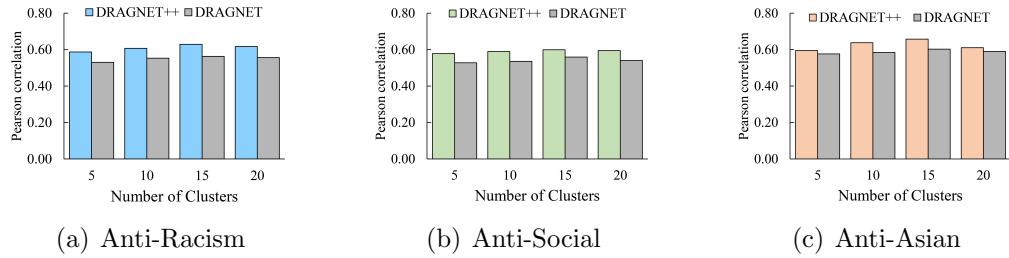


Figure 9: Performance comparison shown by changing the hyperparameter of number of clusters in fuzzy clustering algorithm.

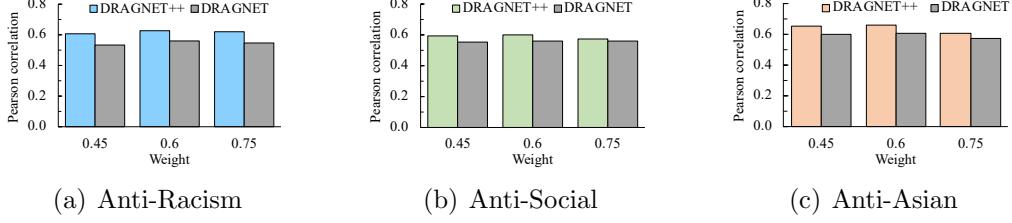


Figure 10: Performance comparison depicted by varying the weight ( $w$ ) which optimizes the trade-off between hate detection/lexicon.

#### 6.5.4. The impact of different weights in hate intensity score

Our hate intensity score leverages  $w$  to simplify the trade-off between the hate detection model and the hate lexicon component in Section 3. Our suggested DRAGNET++ model definitely outperforms DRAGNET for all three values of  $w$  (i.e., 0.45, 0.6, 0.75) on all three datasets, as demonstrated in Fig 10.

#### 6.5.5. The impact of hate speech detection models

During the data preprocessing phase, we utilized a hate speech detection model to compute hate intensity scores for replies. To examine the effect of different hate speech detection models, we evaluated three models proposed by Davidson et al. [79] (model used in DRAGNET++), Founta et al. [24], and Waseem and Hovy Waseem and Hovy [6]. As illustrated in Figure 11, the selected hate speech detection models had a minor impact on the ultimate performance. Nonetheless, our proposed model exhibited consistent superiority over DRAGNET across all three hate detection models.

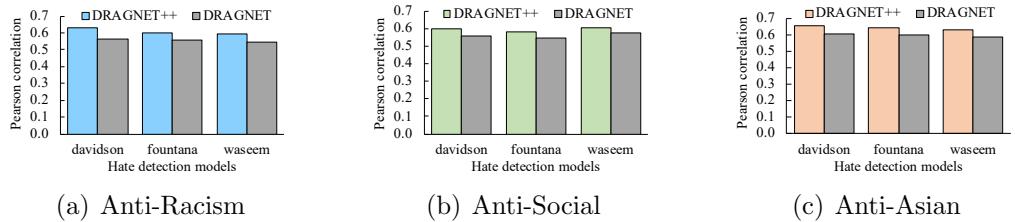


Figure 11: The impact of different hate detection algorithms on DRAGNET++ and DRAGNET

## 7. Conclusion

In this paper, we build on our previous work using two new datasets: *Anti-Social* and *Anti-Asian*. The new datasets gave enough bandwidth to continue experimenting with the old model DRAGNET to forecast hate intensity profiles. By adding the structural graph information critical for any Twitter reply chain, we suggested a new model, DRAGNET++. GNN-augmented DRAGNET++ outperformed all baselines, including DRAGNET, by a large margin. We then used several ablations to compare DRAGNET and DRAGNET++ to figure out why our model outperformed the existing architecture.

In the future, we would like to build a meaningful association of hate intensity profiles at the branch level of the reply chain’s tree structure using sophisticated structural graph information. We would also like to look into how user data can be used to predict hate intensity scores. Human-annotated hate intensity datasets can be included. Our ultimate goal, sooner or later, would be to stop the spread of hate speech as soon as possible after the first tweet appears on Twitter.

## References

- [1] K. Müller, C. Schwarz, Fanning the flames of hate: Social media and hate crime, *Journal of the European Economic Association* (2018).
- [2] Times, Facebook says it’s removing more hate speech than ever before. but there’s a catch, 2019. URL: <https://time.com/5739688/facebook-hate-speech-languages/>.
- [3] Bloomberg, Twitter, facebook join global pledge to fight hate speech online, 2019. URL: <https://www.bloomberg.com/news/articles/2019-05-15/macron-ardern-to-meet-twitter-facebook-google-on-hate-speech>.
- [4] Facebook, Hate speech, 2022. URL: <https://transparency.fb.com/en-gb/policies/community-standards/hate-speech/>.
- [5] Twitter, Hateful conduct policy, 2022. URL: <https://help.twitter.com/en/rules-and-policies/hateful-conduct-policy>.
- [6] Z. Waseem, D. Hovy, Hateful symbols or hateful people? predictive features for hate speech detection on twitter, in: *Proceedings of the NAACL student research workshop*, 2016, pp. 88–93.

- [7] B. Gambäck, U. K. Sikdar, Using convolutional neural networks to classify hate-speech, in: Proceedings of the first workshop on abusive language online, 2017, pp. 85–90.
- [8] A. Schmidt, M. Wiegand, A survey on hate speech detection using natural language processing, in: Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media, Association for Computational Linguistics, Valencia, Spain, 2017, pp. 1–10. URL: <https://aclanthology.org/W17-1101>. doi:10.18653/v1/W17-1101.
- [9] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Computing Surveys (CSUR) 51 (2018) 1–30.
- [10] F. Poletto, V. Basile, M. Sanguinetti, C. Bosco, V. Patti, Resources and benchmark corpora for hate speech detection: a systematic review, Language Resources and Evaluation 55 (2021) 477–523. URL: <https://doi.org/10.1007/s10579-020-09502-8>. doi:10.1007/s10579-020-09502-8.
- [11] P. Liu, J. Guberman, L. Hemphill, A. Culotta, Forecasting the presence and intensity of hostility on instagram using linguistic and social features, in: Twelfth international aaai conference on web and social media, 2018.
- [12] K.-Y. Lin, R. K.-W. Lee, W. Gao, W.-C. Peng, Early prediction of hate speech propagation, in: 2021 International Conference on Data Mining Workshops (ICDMW), IEEE, 2021, pp. 967–974.
- [13] D. Sahnani, S. Dahiya, V. Goel, A. Bandhakavi, T. Chakraborty, Better prevent than react: Deep stratified learning to predict hate intensity of twitter reply chains, in: 2021 IEEE International Conference on Data Mining (ICDM), IEEE, 2021, pp. 549–558.
- [14] J. Navarro, The psychology of hatred, The Open Criminology Journal 6 (2013) 10–17. doi:10.2174/1874917801306010010.
- [15] J. Suler, The online disinhibition effect, CyberPsychology & Behavior 7 (2004) 321–326. URL: <https://doi.org/10.1089/1094931041291295>. doi:10.1089/1094931041291295. arXiv:<https://doi.org/10.1089/1094931041291295>, pMID: 15257832.

- [16] L. Maizland, Hate speech on social media: Global comparisons — council on foreign relations (2019). URL: <https://www.cfr.org/backgrounder/hate-speech-social-media-global-comparisons>.
- [17] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1746–1751.
- [18] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, *Advances in neural information processing systems* 27 (2014).
- [19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* 30 (2017).
- [20] Y. Mehdad, J. Tetreault, Do characters abuse more than words?, in: Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 2016, pp. 299–303.
- [21] P. Badijatiya, S. Gupta, M. Gupta, V. Varma, Deep learning for hate speech detection in tweets, in: Proceedings of the 26th international conference on World Wide Web companion, 2017, pp. 759–760.
- [22] Z. Zhang, D. Robinson, J. Tepper, Detecting hate speech on twitter using a convolution-gru based deep neural network, in: European semantic web conference, Springer, 2018, pp. 745–760.
- [23] A. Arango, J. Pérez, B. Poblete, Hate speech detection is not as easy as you may think: A closer look at model validation, in: Proceedings of the 42nd international acm sigir conference on research and development in information retrieval, 2019, pp. 45–54.
- [24] A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, I. Leontiadis, A unified deep learning architecture for abuse detection, in: Proceedings of the 10th ACM conference on web science, 2019, pp. 105–114.

- [25] M. R. Awal, R. K.-W. Lee, E. Tanwar, T. Garg, T. Chakraborty, Model-agnostic meta-learning for multilingual hate speech detection, *IEEE Transactions on Computational Social Systems* (2023).
- [26] J. H. Park, P. Fung, One-step and two-step classification for abusive language detection on twitter, *arXiv preprint arXiv:1706.01206* (2017).
- [27] F. Del Vigna12, A. Cimino23, F. Dell’Orletta, M. Petrocchi, M. Tesconi, Hate me, hate me not: Hate speech detection on facebook, in: *Proceedings of the First Italian Conference on Cybersecurity (ITASEC17)*, 2017, pp. 86–95.
- [28] R. Cao, R. K.-W. Lee, T.-A. Hoang, Deepbate: Hate speech detection via multi-faceted text representations, in: *12th ACM conference on web science*, 2020, pp. 11–20.
- [29] R. K.-W. Lee, R. Cao, Z. Fan, J. Jiang, W.-H. Chong, Disentangling hate in online memes, in: *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 5138–5147.
- [30] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).
- [31] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training (2018).
- [32] M. Mozafari, R. Farahbakhsh, N. Crespi, A bert-based transfer learning approach for hate speech detection in online social media, in: *International Conference on Complex Networks and Their Applications*, Springer, 2019, pp. 928–940.
- [33] M. R. Awal, R. Cao, R. K.-W. Lee, S. Mitrović, Angrybert: Joint learning target and emotion for hate speech detection, in: *Pacific-Asia conference on knowledge discovery and data mining*, Springer, 2021, pp. 701–713.
- [34] S. Dahiya, S. Sharma, D. Sahnani, V. Goel, E. Chouzenoux, V. Elvira, A. Majumdar, A. Bandhakavi, T. Chakraborty, Would Your Tweet Invoke Hate on the Fly? Forecasting Hate Intensity of Reply Threads on Twitter, in: *Proceedings of the 27th ACM SIGKDD Conference*

- on Knowledge Discovery & Data Mining, ACM, Virtual Event Singapore, 2021, pp. 2732–2742. URL: <https://dl.acm.org/doi/10.1145/3447548.3467150>. doi:10.1145/3447548.3467150.
- [35] V. Basile, C. Bosco, E. Fersini, D. Nozza, V. Patti, F. M. Rangel Pardo, P. Rosso, M. Sanguinetti, SemEval-2019 task 5: Multilingual detection of hate speech against immigrants and women in Twitter, in: Proceedings of the 13th International Workshop on Semantic Evaluation, Association for Computational Linguistics, Minneapolis, Minnesota, USA, 2019, pp. 54–63. URL: <https://aclanthology.org/S19-2007>. doi:10.18653/v1/S19-2007.
  - [36] M. Sanguinetti, G. Comandini, E. Nuovo, S. Frenda, M. Stranisci, C. Bosco, T. Caselli, V. Patti, I. Russo, Haspeede 2 @ evalita2020: Overview of the evalita 2020 hate speech detection task, 2020. doi:10.4000/books.aaccademia.6897.
  - [37] F. Rangel, G. L. D. la Peña Sarracén, B. Chulvi, E. Fersini, P. Rosso, Profiling hate speech spreaders on twitter task at pan 2021, in: CLEF, 2021.
  - [38] J. Pavlopoulos, J. Sorensen, L. Laugier, I. Androutsopoulos, SemEval-2021 task 5: Toxic spans detection, in: Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021), Association for Computational Linguistics, Online, 2021, pp. 59–69. URL: <https://aclanthology.org/2021.semeval-1.6>. doi:10.18653/v1/2021.semeval-1.6.
  - [39] X. Ma, H. Zhong, Y. Li, J. Ma, Z. Cui, Y. Wang, Forecasting transportation network speed using deep capsule networks with nested lstm models, IEEE Transactions on Intelligent Transportation Systems 22 (2020) 4813–4824.
  - [40] T. G. Andersen, T. Bollerslev, P. Christoffersen, F. X. Diebold, Volatility forecasting, 2005.
  - [41] D. Ding, M. Zhang, X. Pan, M. Yang, X. He, Modeling extreme events in time series prediction, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 1114–1122.

- [42] Y. Matsubara, Y. Sakurai, W. G. Van Panhuis, C. Faloutsos, Funnel: automatic mining of spatially coevolving epidemics, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, 2014, pp. 105–114.
- [43] I. Rojas, O. Valenzuela, F. Rojas, A. Guillén, L. J. Herrera, H. Pomares, L. Marquez, M. Pasadas, Soft-computing techniques and arma model for time series prediction, Neurocomputing 71 (2008) 519–537.
- [44] E. S. Gardner Jr, Exponential smoothing: The state of the art, Journal of forecasting 4 (1985) 1–28.
- [45] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcuoglu, Wavenet: A generative model for raw audio, arXiv preprint arXiv:1609.03499 (2016).
- [46] A. Borovykh, S. Bohte, C. W. Oosterlee, Conditional time series forecasting with convolutional neural networks, arXiv preprint arXiv:1703.04691 (2017).
- [47] X. Wei, L. Zhang, H.-Q. Yang, L. Zhang, Y.-P. Yao, Machine learning for pore-water pressure time-series prediction: Application of recurrent neural networks, Geoscience Frontiers 12 (2021) 453–467.
- [48] S. Elsworth, S. Güttel, Time series forecasting using lstm networks: A symbolic approach, arXiv preprint arXiv:2003.05672 (2020).
- [49] Z. Mariet, V. Kuznetsov, Foundations of sequence-to-sequence modeling for time series, in: The 22nd international conference on artificial intelligence and statistics, PMLR, 2019, pp. 408–417.
- [50] C. Fan, Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, H. Huang, Multi-horizon time series forecasting with temporal attention learning, in: Proceedings of the 25th ACM SIGKDD International conference on knowledge discovery & data mining, 2019, pp. 2527–2535.
- [51] R. Sen, H.-F. Yu, I. S. Dhillon, Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting, Advances in neural information processing systems 32 (2019).

- [52] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, *Advances in Neural Information Processing Systems* 32 (2019).
- [53] B. N. Oreshkin, D. Carpow, N. Chapados, Y. Bengio, N-beats: Neural basis expansion analysis for interpretable time series forecasting, arXiv preprint arXiv:1905.10437 (2019).
- [54] Y. Yuan, K. Kitani, Diverse trajectory forecasting with determinantal point processes, arXiv preprint arXiv:1907.04967 (2019).
- [55] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, Deepar: Probabilistic forecasting with autoregressive recurrent networks, *International Journal of Forecasting* 36 (2020) 1181–1191.
- [56] A. Koochali, P. Schichtel, A. Dengel, S. Ahmed, Probabilistic forecasting of sensory data with generative adversarial networks–forgan, *IEEE Access* 7 (2019) 63868–63880.
- [57] M. Wiegand, J. Ruppenhofer, A. Schmidt, C. Greenberg, Inducing a lexicon of abusive words—a feature-based approach, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, June 1–June 6, 2018, New Orleans, Louisiana, Volume 1 (Long Papers), Association for Computational Linguistics, 2019, pp. 1046–1056.
- [58] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, *Advances in neural information processing systems* 32 (2019).
- [59] P. Hastings, S. Hughes, D. Blaum, P. Wallace, M. A. Britt, Stratified learning for reducing training set size, in: *International Conference on Intelligent Tutoring Systems*, Springer, 2016, pp. 341–346.
- [60] V. Niennattrakul, C. A. Ratanamahatana, On clustering multimedia time series data using k-means and dynamic time warping, in: *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, IEEE, 2007, pp. 733–738.

- [61] J. Sun, Y. Li, H.-S. Fang, C. Lu, Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 13250–13259.
- [62] C. Yang, J. Qiao, H. Han, L. Wang, Design of polynomial echo state networks for time series prediction, *Neurocomputing* 290 (2018) 148–160.
- [63] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, *Data Mining and Knowledge Discovery* 34 (2020) 1936–1962.
- [64] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, *arXiv preprint arXiv:1603.06995* (2016).
- [65] A. Ziat, E. Delasalles, L. Denoyer, P. Gallinari, Spatio-temporal neural networks for space-time series forecasting and relations discovery, in: 2017 IEEE International Conference on Data Mining (ICDM), IEEE, 2017, pp. 705–714.
- [66] D. A. Reynolds, Gaussian mixture models., *Encyclopedia of biometrics* 741 (2009).
- [67] Q. Cao, H. Shen, J. Gao, B. Wei, X. Cheng, Popularity prediction on social platforms with coupled graph neural networks, in: Proceedings of the 13th International Conference on Web Search and Data Mining, 2020, pp. 70–78.
- [68] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, J. Huang, Rumor detection on social media with bi-directional graph convolutional networks, in: Proceedings of the AAAI conference on artificial intelligence, volume 34, 2020, pp. 549–556.
- [69] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE transactions on neural networks* 20 (2008) 61–80.

- [70] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems* 26 (2013).
- [71] M. R. Awal, R. Cao, S. Mitrovic, R. K.-W. Lee, On Analyzing Antisocial Behaviors Amid COVID-19 Pandemic, arXiv:2007.10712 [cs] (2020). URL: <http://arxiv.org/abs/2007.10712>, arXiv: 2007.10712.
- [72] B. He, C. Ziems, S. Soni, N. Ramakrishnan, D. Yang, S. Kumar, Racism is a virus: anti-asian hate and counterspeech in social media during the COVID-19 crisis, in: *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ACM, Virtual Event Netherlands, 2021, pp. 90–94. URL: <https://dl.acm.org/doi/10.1145/3487351.3488324>. doi:10.1145/3487351.3488324.
- [73] S. Elsworth, S. Güttel, Time Series Forecasting Using LSTM Networks: A Symbolic Approach, arXiv:2003.05672 [cs, stat] (2020). URL: <http://arxiv.org/abs/2003.05672>, arXiv: 2003.05672.
- [74] S. Mehtab, J. Sen, S. Dasgupta, Robust Analysis of Stock Price Time Series Using CNN and LSTM-Based Deep Learning Models, in: *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, IEEE, Coimbatore, India, 2020, pp. 1481–1486. URL: <https://ieeexplore.ieee.org/document/9297652/>. doi:10.1109/ICECA49313.2020.9297652.
- [75] B. N. Oreshkin, D. Carpow, N. Chapados, Y. Bengio, N-BEATS: Neural basis expansion analysis for interpretable time series forecasting, arXiv:1905.10437 [cs, stat] (2020). URL: <http://arxiv.org/abs/1905.10437>, arXiv: 1905.10437.
- [76] B. Lim, S. O. Arik, N. Loeff, T. Pfister, Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting, arXiv:1912.09363 [cs, stat] (2020). URL: <http://arxiv.org/abs/1912.09363>, arXiv: 1912.09363.
- [77] A. Koochali, P. Schichtel, S. Ahmed, A. Dengel, Probabilistic Forecasting of Sensory Data with Generative Adversarial Networks - ForGAN, *IEEE Access* 7 (2019) 63868–63880. URL: <http://arxiv.org/abs/1903.12549>. doi:10.1109/ACCESS.2019.2915544, arXiv: 1903.12549.

- [78] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated Hate Speech Detection and the Problem of Offensive Language, arXiv:1703.04009 [cs] (2017). URL: <http://arxiv.org/abs/1703.04009>, arXiv: 1703.04009.
- [79] T. Davidson, D. Warmsley, M. Macy, I. Weber, Automated hate speech detection and the problem of offensive language, in: Proceedings of the International AAAI Conference on Web and Social Media, volume 11, 2017, pp. 512–515.