Would Your Tweet Invoke Hate on the Fly? Forecasting Hate Intensity of Reply Threads on Twitter

Snehil Dahiya*, Shalini Sharma*, Dhruv Sahnan*, Vasu Goel*, Emilie Chouzenoux◇, Víctor Elvira‡, Angshul Majumdar*, Anil Bandhakavi†, Tanmoy Chakraborty*
*IIIT-Delhi, India; ◇Inria Saclay, University Paris Saclay, France; ‡University of Edinburgh, UK; †Logically, UK
{snehil19046,shalinis,dhruv18230,vasu18322,angshul,tanmoy}@iiitd.ac.in
victor.elvira@ed.ac.uk;emilie.chouzenoux@centralesupelec.fr;anil@logically.ai

ABSTRACT

Curbing hate speech is undoubtedly a major challenge for online microblogging platforms like Twitter. While there have been studies around hate speech detection, it is not clear how hate speech finds its way into an online discussion. It is important for a content moderator to not only identify which tweet is hateful, but also to predict which tweet will be responsible for accumulating hate speech. This would help in prioritizing tweets that need constant monitoring. Our analysis reveals that for hate speech to manifest in an ongoing discussion, the source tweet may not necessarily be *hateful*; rather, there are plenty of such non-hateful tweets which gradually invoke hateful replies, resulting in the entire reply threads becoming provocative.

In this paper, we define a novel problem – *given a source tweet and a few of its initial replies, the task is to forecast the hate intensity of upcoming replies*. To this end, we curate a novel dataset constituting $\sim 4.5k$ contemporary tweets and their entire reply threads. Our preliminary analysis confirms that the evolution patterns along time of hate intensity among reply threads have highly diverse patterns, and there is no significant correlation between the hate intensity of the source tweets and that of their reply threads. We employ seven state-of-the-art dynamic models (either statistical signal processing or deep learning based) and show that they fail badly to forecast the hate intensity. We then propose DESSERT, a novel deep state-space model that leverages the function approximation capability of deep neural networks with the capacity to quantify the uncertainty of statistical signal processing models. Exhaustive experiments and ablation study show that DESSERT outperforms all the baselines substantially. Further, its deployment in an advanced AI platform designed to monitor real-world problematic hateful content has improved the aggregated insights extracted for countering the spread of online harms.

T. Chakraborty would like to acknowledge the support of Logically, the Ramanujan Fellowship, and the Infosys Centre for AI, IIIT Delhi. We also thank Sarah Masud for her help in writing the paper.

CCS CONCEPTS

• **Computing methodologies → Machine learning algorithms**; • **Information systems → Social tagging systems**; • **Human-centered computing → Social network analysis**.

KEYWORDS

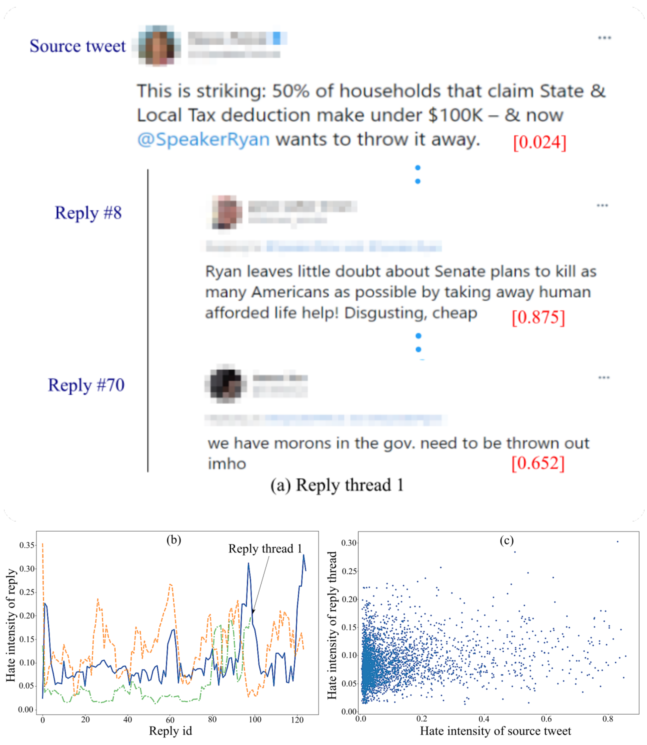
Hate speech, Signal processing, State-space model, Time series forecasting

ACM Reference Format:
Snehil Dahiya, Shalini Sharma, Dhruv Sahnan, Vasu Goel, Emilie Chouzenoux, Víctor Elvira, Angshul Majumdar, Anil Bandhakavi, Tanmoy Chakraborty. 2021. 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 and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467150

1 INTRODUCTION

Online discussion forums such as Twitter, and Reddit, provide users with the opportunity to express their opinion freely and that too, anonymously. Although freedom of speech enables users to seek, express and impart information about ideas and opinions, its misuse often leads to social instability both online and offline, resulting in cyber-crimes such as hate speech and cyberbullying. Hate speech is highly subjective – its nature varies across demography; its effect varies across religion, identity and social groups. Therefore, most of the hate speech detection models which leverage hand-crafted lexicons in one form or the other, often struggle to generalize. Recently, there has been a growing body of research in detection and characterizing online hate speech and modeling its spread [11].

Although hate speech detection is highly pertinent towards the broader goal of sanitizing online content, it is often observed that a non-hate (benign) post, over the time, evolves as a source of provocative discussion, fostering the generation of hateful or offensive posts. For example, Figure 1(a) shows a (partial) reply thread along with the hate intensity (defined in Section 3.2) of each reply. As observed, although the source tweet is less hate intensive and is not detected as hate speech by the hate speech classifier, the replies generated under this tweet are highly hateful. Figure 1(b) shows the hate intensity profile of the entire reply thread of the previous example – it does not follow any particular pattern. None of the hate speech detection models would be able to predict that such a benign online post would ever invoke hatred in near future. This


Source tweet

This is striking: 50% of households that claim State & Local Tax deduction make under $100K – & now @SpeakerRyan wants to throw it away. [0.024]

Reply #8

Ryan leaves little doubt about Senate plans to kill as many Americans as possible by taking away human afforded life help! Disgusting, cheap [0.875]

Reply #70

we have morons in the gov. need to be thrown out imho [0.652]

(a) Reply thread 1

(b)
Reply thread 1
Hate intensity of reply
Reply id
(c)
Hate intensity of reply thread
Hate intensity of source tweet

Figure 1: (a) An example reply thread with the hate intensity per reply (within brackets). (b) Hate intensity profile of three example reply threads, one of which is (a). We observe that that the hate intensity of reply threads does not follow any particular pattern. (c) Scatter plot of hate intensity of the source tweets and their corresponding reply threads, indicating they are uncorrelated (Pearson's r =0.11).

calls for an early prediction model to forecast whether a tweet will invoke hate speech.

Hate intensity profiles of reply threads are diverse. We formally pose the problem as follows: *given a tweet* (*and the initial history of its reply thread*), *we aim to forecast the hate intensity that its upcoming replies will express.* To this end, we curate entire reply threads of $\sim 4.5k$ tweets; the average length of a thread is ~ 200. We quantify the hate intensity of a reply thread by leveraging a hate speech classifier and a benchmark hate lexicon. We observe that the hate intensity profiles of reply threads are non-uniform, exhibiting diverse patterns (c.f. Figure 1(b)). Moreover, the hate intensity of reply threads is not correlated with that of the source tweet (c.f. Figure 1(c)), indicating that identifying hateful source tweet may not be enough to predict the upcoming toxicity of a reply thread. It is also uncertain which quarter of reply threads is more prone to hate speech (c.f. Figure 5 in the Appendix).

Limitations of state-of-the-art methods. One can readily map this into a dynamical modeling problem – given an ordered sequence of data points representing hate intensity of current reply thread of a source tweet, the aim is to predict the hate intensity of the upcoming replies. One can borrow two off-the-shelf approaches to address this problem. First is the classical signal processing models like auto-regressive moving average (ARMA) and its variants (ARIMA, ARIMAX, etc.) or its state-space counterparts. The problem with such models is that one needs to *apriori* specify the underlying dynamical function; this is not possible for the current problem. The second approach is to use deep learning models such as recurrent neural networks (such as LSTM and GRU) and generative adversarial network (GAN). These techniques do not require specification of the underlying function; they can learn it from the data. However, the problem with them is that they cannot quantify the *uncertainty* about the prediction - which the signal processing models can.

Our proposed approach: DESSERT. In this work, we propose to keep the best of both worlds mentioned above – the function approximation ability of neural networks and the uncertainty quantification capability of state-space models. We propose DESSERT[1], a **deep blind state-space** model. Here 'blind' means that we do not need to specify the underlying dynamical model / function; we learn it from the data. However, learning one operator for the state-evolution and one operator for the observation limits the function approximation capability of the blind state-space model. Moreover, by making the model deep, we aim to better capture the non-linearity dynamics of online interactions.

Experimental results. We perform an exhaustive evaluation of DESSERT with seven state-of-the-art baselines (borrowed from both classical signal processing and deep learning domains) on our curated dataset. We observe that DESSERT outperforms all the baselines across four evaluation metrics (both correlation-based and error-based). DESSERT achieves 0.67 Pearson's r and 31.08 Mean Absolute Percentage Error (MAPE), which is significantly better than the best baseline [18] (r = 0.557, MAPE=43.47). We also present a detailed ablation study of DESSERT. We further dig deeper into the results to explain how the models respond to reply threads – (i) whose source tweets are originated from across demography, (ii) whose source tweets are posted by different types of users, (iii) whose source tweets are of different types (hate, fake, controversial and others), and (iv) with different thread length. Unlike other baselines, DESSERT also outputs a confidence score with each prediction, which a content moderator may use to prioritize the tweets that need constant inspection.

Real-world deployment. DESSERT has been deployed in an advanced AI platform at Logically[2] designed to monitor real-world problematic hateful content in large volumes of social media data. It has been tested by integrating into proprietary and custom pipelines that extract insights and intelligence to counter the spread of online harms. We observe that DESSERT offers complementary insights and has significant potential to improve the overall accuracy of the pipelines in detecting patterns and extracting insights concerning hate speech on real-world online content.

In particular, our contributions are five-fold: ▲ **Novel problem definition.** To our knowledge, we are the first to address the problem of forecasting hate intensity of upcoming reply threads of tweets. ▲ **Novel model.** Our proposed model, DESSERT leverages the advantages of classical signal processing and recently proposed deep learning techniques. ▲ **New dataset:** Our curated dataset consists of contemporary tweets and the *entire reply threads* on

[1]**DE**ep **S**tate-**S**pac**E** model for hate intensity prediction of **R**eply **T**hreads.
[2]https://www.logically.ai (see Appendix for more details about Logically).

diverse topics that are posted across demography. ▲ **Exhaustive results.** DESSERT outperforms seven state-of-the-art baselines with significant margin. ▲ **System deployment.** DESSERT has been extensively tested and deployed by Logically in an advanced AI platform for detecting real-world problematic hateful content to extract insights for countering the spread of online harms (see Appendix).

Reproducibility. We present detailed hyper-parameter configurations in the Appendix. Source codes and datasets are available at https://github.com/LCS2-IIITD/DESSERt_KDD21.

2 RELATED WORK

We discuss the literature survey into three parts - hate speech detection, other studies related to hate speech, and some of the studies on time-series modeling which are pertinent to our paper. We also present how our method is different from existing studies.

Hate speech detection. Initial studies on hate speech detection [7] made use of n-gram based Logistic Regression and SVM; later Badjatiya et al. [1] extended their work to LSTM. In last two years, bigger datasets [12] and BERT-based models have shown to perform better [27]. In addition, Gomez et al. [14] explored the impact of multi-modality for hate detection. Over time, the focus also shifted from English datasets to diverse regional languages like Indonesian [17], Vietnamese [8], Urdu [32], etc. In order to make the results of automatic hate speech classification explainable, Mathew et al. [23] proposed to capture words and phrases contributing to hate class. Poletto et al. [29] presented a comprehensive comparative study of various hate detection models. Note, in this work, we do not propose a new hate detection model, rather we leverage the existing state of art hate classifiers [7, 12] for measuring hate intensity of a post.

Other studies on hate speech. Apart from the task of hate speech detection, hateful content has also been studied for analysing user characteristics and countering hateful information. Ribeiro et al. [31] performed various statistical analysis to characterise hateful users on Twitter. Meanwhile, similar studies have been conducted on platforms like Gab [40], BitChute [38], etc. To further study the dynamics of hate speech diffusion, Bagavathi et al. [2] performed exploratory analysis comparing hateful and non-hateful information cascades on Gab. We recently proposed a prediction model for hate speech diffusion on Twitter using exogenous signals [21]. There have been studies around automatic hate countering on Facebook [25] and Twitter [22]. The benchmark datasets [6] and strategies [37] can be employed to build generative models that can act as first line of defence in countering hateful content [30].

Time-series modeling. The classical techniques like ARMA [33] and its variants (ARIMA, SARIMA) have shown outstanding performance in modeling time-series signals. These models are dependent on the assumption of data being linear and stationary, neither of which hold for real-time scenarios. They are collectively known as the Box-Jenkins method [3]. Another alternate approach is the state-space model (SSM). Kalman filter is one such model widely used to forecast time-series signals when the model is linear, and the noise is modeled as Gaussian. Real application such as ours are even challenging for SSM as it requires prior knowledge of the state and observation parameters. Real-time streaming data is highly volatile; therefore, it becomes difficult to assume the model parameters in advance. The inability of signal processing techniques helped machine learning approaches gain popularity. These approaches can successfully address the learning-based system [16] with no prior assumptions on the model parameters. One of its variant, RNNs are designed explicitly to address time-series signals. However, the performance of RNN degrades when the sequences grows longer due to long-term dependencies. To overcome this, Long Short Term Memory (LSTM) [9] came into existence. LSTM still could not model the long-term dependencies well. These limitations paved the path for 1D CNN [24]. The neural network approaches do not require specifications and prior knowledge of the model parameters. However, they do not provide any confidence score on the point estimate. Most of the research in time-series forecasting does not discuss the confidence score associated with the prediction. In recent years, its necessity has become prominent and has been considered crucial in addressing challenging applications [20]. Several recent studies of time-series modeling capture the underlying non-linearity into the models [34].

Differences with the existing studies. Our work proposes a novel learning-based approach to bridge the gap between the state-space models and neural network approaches. Our proposed approach, DESSERT is a learning-based approach to model time-series signals using deep state-space. DESSERT needs no structural details of model parameters and can model confidence score associated with the next timestamp prediction, thereby keeping both worlds' best.

3 PRELIMINARIES

3.1 Problem formulation

Given a tweet τ and its reply thread till t, denoted by $\mathcal{R}^{\tau}_{1,t} = \langle r^{\tau}_1, r^{\tau}_2, \ldots, r^{\tau}_t \rangle$, where r^{τ}_t denotes a reply tweet at time t, we aim to predict the intensity of hatred $\mathcal{H}(.)$ that the upcoming chunks of replies (each chunk with size Δ time window) will express, i.e., $\mathcal{H}(\mathcal{R}^{\tau}_{t+1,t+\Delta}), \ldots, \mathcal{H}(\mathcal{R}^{\tau}_{t+(n-1)\Delta,t+n\Delta})$. Here, t denotes an integer index associated to t-th reply in the sequence, not the actual (continuous) time. A **reply chunk** is a set of Δ consecutive replies to a source tweet τ. This problem can be readily mapped to a classical time-series prediction problem as follows. We break $\mathcal{R}^{\tau}_{1,t}$ into a sequence of chunks each with size Δ and calculate the hate intensity per window, i.e., $\mathcal{H}(\mathcal{R}^{\tau}_{1,\Delta}), \mathcal{H}(\mathcal{R}^{\tau}_{\Delta+1,2\Delta}), \ldots, \mathcal{H}(\mathcal{R}^{\tau}_{t-\Delta,t})$. This constitutes the training data points. We then aim to predict the hate intensity of the next chunks of replies.

3.2 Quantifying hate intensity

In order to quantify $\mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta})$, the hate intensity of a chunk of replies posted between k and $k+\Delta$, with $1 \le k \le t-\Delta$, we rely on two scores: a model-dependent score, and a lexicon-based score.

- **Hate speech classifier:** We employ a state-of-the-art hate speech classifier C which takes a reply r and classifies it into hate speech with a score $C(r)$, indicating the probability of reply being a hate speech. We consider $C(r)$ in calculating $\mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta})$.
- **Lexical hate score:** Wiegand et al. [39] proposed a domain-independent hate lexicon with 2,895 hate words. They also assigned hate score to each word in the lexicon. Given a reply r, we check the presence of such hate words and sum up their hate

scores to obtain a model-independent lexicon-based hate score per reply, denoted by $\mathcal{L}(r)$.

The hate intensity of a reply r is measured as,

$$\mathcal{H}(r) = w\mathcal{C}(r) + (1 - w)\mathcal{L}(r), \tag{1}$$

where w ($0 \leq w \leq 1$) adjusts the weights of two components.

The hate intensity of a chunk of replies is measured by the sum of hate intensity of all the replies in that chunk, i.e.,

$$\mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta}) = \sum_{r\in\mathcal{R}^{\tau}_{k,k+\Delta}} \mathcal{H}(r) = w\left(\sum_{r\in\mathcal{R}^{\tau}_{k,k+\Delta}} \mathcal{C}(r)\right)+(1-w)\left(\sum_{r\in\mathcal{R}^{\tau}_{k,k+\Delta}} \mathcal{L}(r)\right). \tag{2}$$

Note that $0 \leq \mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta}) \leq \infty$. Section 7.2 will show the results with different hate speech classifiers $\mathcal{C}$, and varying w. For the chunks of replies till t in the training set, we measure the hate intensity of each chunk, $\mathcal{H}(\mathcal{R}^{\tau}_{1,\Delta})$, $\mathcal{H}(\mathcal{R}^{\tau}_{\Delta+1,2\Delta})$, $\ldots$, $\mathcal{H}(\mathcal{R}^{\tau}_{t-\Delta,t})$.

3.3 Feature extraction from a reply chunk

In the training set, from a given chunk $\mathcal{R}^{\tau}_{k,k+\Delta}$, we measure the hate intensity score of its constituent replies, i.e., $\mathcal{H}(r^{\tau}_{k})$, $\mathcal{H}(r^{\tau}_{k+1})$, $\ldots$, $\mathcal{H}(r^{\tau}_{k+\Delta})$. From this, we extract five quantities, namely:
(i) sum of hate intensities, $\mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta})$,
(ii) hate intensity of the first reply for the chunk, $\mathcal{H}(r^{\tau}_{k})$,
(iii) hate intensity of the last reply for the chunk, $\mathcal{H}(r^{\tau}_{k+\Delta})$,
(iv) max hate intensity, $\mathcal{H}\left(\mathcal{R}^{\tau}_{k,k+\Delta}\right)_{\max} = \max_{r\in\mathcal{R}^{\tau}_{k,k+\Delta}} \mathcal{H}(r)$,
(v) min hate intensity, $\mathcal{H}\left(\mathcal{R}^{\tau}_{k,i+\Delta}\right)_{\min} = \min_{r\in\mathcal{R}^{\tau}_{k,k+\Delta}} \mathcal{H}(r)$.

This results in a 5-dimensional observed feature vector associated to $\mathcal{R}^{\tau}_{k,k+\Delta}$:

$$\mathbf{x}_k = \left[\mathcal{H}(\mathcal{R}^{\tau}_{k,k+\Delta}), \mathcal{H}(r^{\tau}_{k}), \mathcal{H}(r^{\tau}_{k+\Delta}), \mathcal{H}\left(\mathcal{R}^{\tau}_{k,i+\Delta}\right)_{\max}, \mathcal{H}\left(\mathcal{R}^{\tau}_{k,i+\Delta}\right)_{\min}\right].$$

The training set is formed by $(\mathbf{x}_k)_{1\leq k\leq K}$ where each $\mathbf{x}_k \in \mathbb{R}^5$, and $K = t - \Delta$ is the number of chunks in the training set.

4 PROPOSED MODEL: DESSERT

Our objective is to predict the hate intensity of a reply thread for a given source tweet. We formulate this problem as a dynamical modeling problem. Off-the-shelf approaches to address such problems like ARMA, its variants, and state-space models, are proposed either in classical signal processing or in machine learning, namely RNN and its variants like LSTM and GRU. The disadvantage of signal processing approaches is that the underlying dynamical evolution function needs to be specified. This may be inappropriate in a typical scenario such as ours as the underlying evolution function is non-stationary and highly volatile in nature. On the other hand, machine learning approaches learn the underlying function from the data [15]. However, the shortcoming of off-the-shelf RNNs is that unlike state-space models, they cannot predict the confidence score around the estimate. The confidence score is highly relevant to our problem as based on this, the online content moderators may decide which thread to inspect manually.

To overcome the limitations of both, we propose DESSERT, a **Deep State-Space** (**DSS**) model. DESSERT is built on the advantages of both schools of methods. First, unlike the standard state-space model where the state evolution and the observation operators are supposed to be known *apriori*, we will learn it from the data. Second, in order to handle non-linearity, we introduce multiple layers of learnable parameters for both the observation and the state evolution. DESSERT keeps the ability of function approximation of neural networks (since it learns the parameters from the data) and the capacity to model uncertainty from classical state-space models. The schematic diagram of DESSERT is shown in Figure 2

4.1 Basis architecture

We consider $(\mathbf{x}_k)_{1\leq k\leq K}$ the observed time-ordered sequence vector of size N_x, $(\mathbf{z}_k)_{1\leq k\leq K}$ is the latent state-space vector of size N_z that we want to infer, and $(\mathbf{v}_k)_{1\leq k\leq K}$ models the noise. As explained in Section 3.3, we consider $N_x = 5$ features. Moreover, we consider the possibility of a multivariate hidden feature space, i.e., N_z can be greater than 1. The original linear state-space model is given by,

$$\begin{cases} \mathbf{z}_k = \mathbf{A}\mathbf{z}_{k-1} + \mathbf{v}_{1,k}, \\ \mathbf{x}_k = \mathbf{H}\mathbf{z}_k + \mathbf{v}_{2,k}, \end{cases} \tag{3}$$

where the matrices, $\mathbf{A}$ and $\mathbf{H}$, are the learnable parameters. When $\mathbf{A}$ and $\mathbf{H}$ are known, the solution is the celebrated Kalman filter. When the parameters are unknown, the solution to the blind Kalman filtering problem has been recently investigated in [36]. A limitation of this aforementioned work is that it cannot handle non-linearity in the dynamical system. In this present study, we propose to account for the non-linearity by introducing multiple layers (3 layers in our case) in the model. The DSS model is given as follows. For every $k \in \{1, \ldots, K\}$,

$$\begin{cases} \mathbf{z}_k = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{z}_{k-1} + \mathbf{v}_{1,k}, \\ \mathbf{x}_k = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k + \mathbf{v}_{2,k}, \end{cases} \tag{4}$$

where $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ can be understood as latent factors for the state matrix $\mathbf{A}$ and the observation matrix $\mathbf{H}$, respectively. Process noise $(\mathbf{v}_{1,k})_{1\leq k\leq K}$ is zero-mean Gaussian with covariance matrix $\mathbf{Q}$, and the observation noise $(\mathbf{v}_{2,k})_{1\leq k\leq K}$ is zero-mean Gaussian with covariance matrix $\mathbf{R}$. Then Equation 4 describes a first-order Markovian multi-linear Gaussian model, where $(\mathbf{z}_k)_{1\leq k\leq K}$ is the sequence of K unknown states, which can be seen as learned features for describing the data. The goal is the joint inference from the observed sequence $(\mathbf{x}_k)_{1\leq k\leq K}$ of the factors $\mathbf{A}_0 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{A}_1 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{A}_2 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{H}_0 \in \mathbb{R}^{N_x\times N_z}$, $\mathbf{H}_1 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{H}_2 \in \mathbb{R}^{N_z\times N_z}$ and of the hidden state sequence $(\mathbf{z}_k)_{1\leq k\leq K}$.

This problem can be considered as blind filtering inference where both the estimated series and model parameters are unknown initially, and will be learnt and estimated from the data. The model is expected to understand trends and update its parameters for every chunk from the data. This can be done following an expectation-majorization algorithm [35]. The problem is solved in an alternating manner where we alternate between (i) the estimation of the state, considering operators $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ being fixed, and (ii) the update of the parameters, assuming fixed state. Both steps are elaborated below.

4.2 State update

This is the first step where we consider the parameters $\mathbf{A}_0$, $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ to be fixed and known, and the goal is to infer the latent

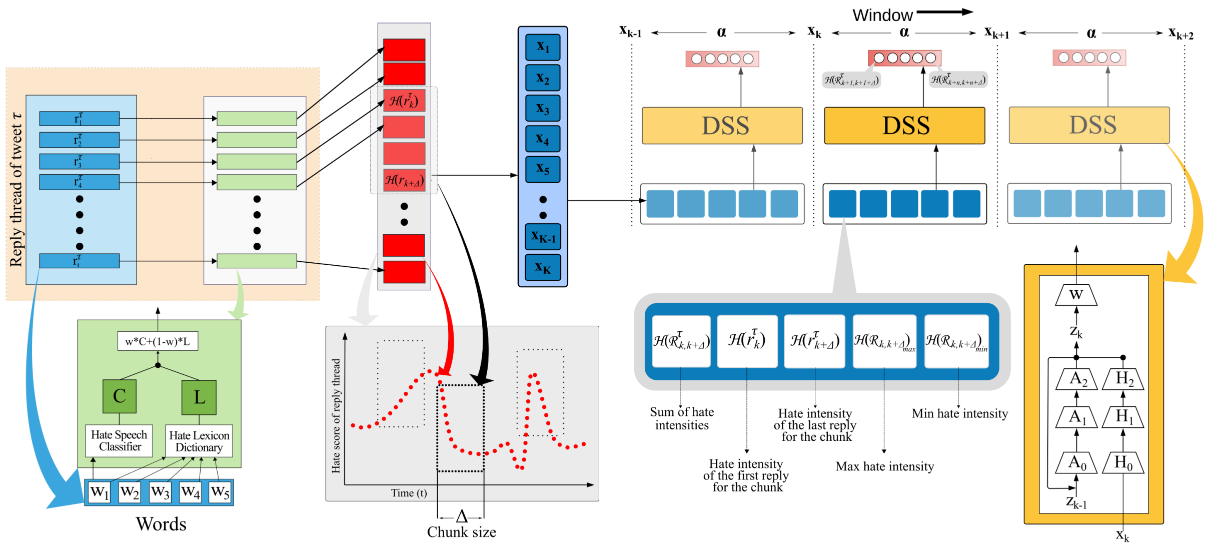

Reply thread of tweet τ
r_1^τ
r_2^τ
r_3^τ
r_4^τ
r_t^τ
w*C+(1-w)*L
C
L
Hate Speech Classifier
Hate Lexicon Dictionary
W_1 W_2 W_3 W_4 W_5
Words
$\mathcal{H}(r_k^\tau)$
$\mathcal{H}(r_{k+\Delta})$
Hate score of reply thread
Time (t)
Δ
Chunk size
x_1 x_2 x_3 x_4 x_5 x_{K-1} x_K
Window
x_{k-1} α x_k α x_{k+1} α x_{k+2}
$\mathcal{H}(\mathcal{R}^\tau_{k+1,k+1+\Delta})$
$\mathcal{H}(\mathcal{R}^\tau_{k+n,k+n+\Delta})$
DSS
DSS
DSS
$\mathcal{H}(\mathcal{R}^\tau_{k,k+\Delta})$
$\mathcal{H}(r_k^\tau)$
$\mathcal{H}(r^\tau_{k+\Delta})$
$\mathcal{H}(\mathcal{R}_{k,k+\Delta})_{max}$
$\mathcal{H}(\mathcal{R}_{k,k+\Delta})_{min}$
Sum of hate intensities
Hate intensity of the first reply for the chunk
Hate intensity of the last reply for the chunk
Max hate intensity
Min hate intensity
W
z_k
A_2 H_2
A_1 H_1
A_0 H_0
z_{k-1}
x_k

Figure 2: Schematic diagram of DESSERT where x_k represents the input (ordered sequence of vectors), z_{k-1} represents the hidden state-space vector (i.e. feature) for previous timestamp. $A_0, A_1, A_2, H_0, H_1, H_2$ are the learnable parameters, and z_k represents the hidden state-space vector for current timestamp.

state. Assume that the initial state follows $\mathbf{z}_0 \sim \mathcal{N}(\bar{\mathbf{z}}_0, \mathbf{P}_0)$, where $\mathbf{P}_0$ is a symmetric definite positive matrix of $\mathbb{R}^{N_z \times N_z}$ and $\bar{\mathbf{z}}_0 \in \mathbb{R}$. Then, Equation 4 reads as a first-order Markovian multi-linear Gaussian model, where $(\mathbf{z}_k)_{1\le k \le K}$ is the sequence of K unknown states. The Kalman filter provides a probabilistic estimate of the hidden state at each time step k, conditioned to all available data up to time k, through the filtering distribution:

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}) = \mathcal{N}(\mathbf{z}_k; \bar{\mathbf{z}}_k, \mathbf{P}_k). \tag{5}$$

where $\bar{\mathbf{z}}_k \in \mathbb{R}^{N_z}$ and $\mathbf{P}_k \in \mathbb{R}^{N_z \times N_z}$ are the mean and covariance, respectively, of the filtering distribution. For every $k \in \{1, \dots, K\}$, $\bar{\mathbf{z}}_k$ and $\mathbf{P}_k$ can be computed by means of the Kalman filter recursions as follows: For $k = 1, \dots, K$

Predict state:

$$\begin{cases} \mathbf{z}_k^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\bar{\mathbf{z}}_{k-1}, \\ \mathbf{P}_k^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{P}_{k-1}(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top + \mathbf{Q}. \end{cases} \tag{6}$$

Update state:

$$\begin{cases} \mathbf{y}_k = \mathbf{x}_k - \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2 z_k^-, \ \mathbf{S}_k = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{P}_k^-(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top + \mathbf{R}, \\ \mathbf{K}_k = \mathbf{P}_k^-(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top\mathbf{S}_k^{-1}, \\ \bar{\mathbf{z}}_k = \mathbf{z}_k^- + \mathbf{K}_k\mathbf{y}_k, \\ \mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \end{cases} \tag{7}$$

The Rauch-Tung-Striebel (RTS) smoother [35] makes a backward recursion on the data which makes use of the filtering distributions computed by the Kalman filter to obtain the smoothing distribution $p(\mathbf{z}_k|\mathbf{x}_{1\dots K})$. Below, we summarize the RTS recursions:
For $k = K, \dots, 1$

Backward Recursion (*Bayesian Smoothing*):

$$\begin{cases} \mathbf{z}_{k+1}^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\bar{\mathbf{z}}_k, \ \mathbf{P}_{k+1}^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{P}_k(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top + \mathbf{Q}, \\ \mathbf{G}_k = \mathbf{P}_k(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top(\mathbf{P}_{k+1}^-)^{-1}, \\ \mathbf{z}_k^s = \mathbf{z}_k + \mathbf{G}_k[\mathbf{z}_{k+1}^s - \mathbf{z}_{k+1}^-], \\ \mathbf{P}_k^s = \mathbf{P}_k + \mathbf{G}_k(\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-)\mathbf{G}_k^\top. \end{cases} \tag{8}$$

As a result, the smoothing distribution at each time k is a multivariate Gaussian with closed-form given by, $p(\mathbf{z}_k|\mathbf{x}_{1\dots K}) = \mathcal{N}(\mathbf{z}_k; \mathbf{z}_k^s, \mathbf{P}_k^s)$.

4.3 Parameter updates

The training procedure of DESSERT requires an alternating iteration of Kalman filter/smoother, considering parameters to be known and fixed initially (E-step), followed by the model/state parameters estimation assuming latent states to be fixed (M-step) [35]. It starts with an initialization stage $\mathbf{A}_0^{[0]}, \mathbf{A}_1^{[0]}, \mathbf{A}_2^{[0]}$ and $\mathbf{H}_0^{[0]}, \mathbf{H}_1^{[0]}, \mathbf{H}_2^{[0]}$. For every iteration i, it runs the RTS scheme, presented in Section 4.2, then computes the updates $\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2^{[i+1]}$ and $\mathbf{H}_0^{[i+1]}, \mathbf{H}_1^{[i+1]}, \mathbf{H}_2^{[i+1]}$, given their estimates at previous EM iteration, i.e., $\mathbf{A}_0^{[i]}, \mathbf{A}_1^{[i]}, \mathbf{A}_2^{[i]}$ and $\mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}$. This amounts to maximizing this lower bound, so as to increase the marginal log-likelihood of these six deep latent factors, given the observed data. Let us introduce some useful quantities, defined from the RTS recursion:

$$\begin{aligned} \Sigma^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^s + \mathbf{z}_k^s(\mathbf{z}_k^s)^\top; \ \Phi^{[i]} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_{k-1}^s + \mathbf{z}_{k-1}^s(\mathbf{z}_{k-1}^s)^\top, \\ \mathbf{B}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k(\mathbf{z}_k^s)^\top; \ \mathbf{D}^{[i]} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^s\mathbf{G}_{k-1}^\top + \mathbf{z}_k^s(\mathbf{z}_{k-1}^s)^\top, \\ \Gamma^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k\mathbf{x}_k^\top. \end{aligned} \tag{9}$$

Then, the computation of $\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2^{[i+1]}$ and $\mathbf{H}_0^{[i+1]}, \mathbf{H}_1^{[i+1]}, \mathbf{H}_2^{[i+1]}$ amounts to solving the following subproblems (see Appendix B for detailed derivations):

$$\begin{aligned} \mathbf{A}_0^{[i+1]} = \operatorname{argmin}_{\mathbf{A}_0} \Big(\frac{K}{2} \operatorname{tr} \big(\mathbf{Q}^{-1}\Sigma^{[i]} - \mathbf{D}^{[i]}(\mathbf{A}_0\mathbf{A}_1^{[i]}\mathbf{A}_2^{[i]})^\top \\ - \mathbf{A}_0\mathbf{A}_1^{[i]}\mathbf{A}_2^{[i]}(\mathbf{D}^{[i]})^\top + \mathbf{A}_0\mathbf{A}_1^{[i]}\mathbf{A}_2^{[i]}\Phi^{[i]}(\mathbf{A}_0\mathbf{A}_1^{[i]}\mathbf{A}_2^{[i]})^\top \big) \Big), \end{aligned}$$

$$\mathbf{A}_1^{[i+1]} = \operatorname{argmin}_{\mathbf{A}_1}\left(\frac{K}{2}\operatorname{tr}\left(\mathbf{Q}^{-1}\Sigma^{[i]} - \mathbf{D}^{[i]}(\mathbf{A}_0^{[i+1]}\mathbf{A}_1\mathbf{A}_2^{[i]})^\top \right.\right.$$
$$\left.\left. -\mathbf{A}_0^{[i+1]}\mathbf{A}_1\mathbf{A}_2^{[i]}(\mathbf{D}^{[i]})^\top + \mathbf{A}_0^{[i+1]}\mathbf{A}_1\mathbf{A}_2^{[i]}\Phi^{[i]}(\mathbf{A}_0^{[i+1]}\mathbf{A}_1\mathbf{A}_2^{[i]})^\top\right)\right),$$

$$\mathbf{A}_2^{[i+1]} = \operatorname{argmin}_{\mathbf{A}_2}\left(\frac{K}{2}\operatorname{tr}\left(\mathbf{Q}^{-1}\Sigma^{[i]} - \mathbf{D}^{[i]}(\mathbf{A}_0^{[i+1]}\mathbf{A}_1^{[i+1]}\mathbf{A}_2)^\top \right.\right.$$
$$\left.\left. -\mathbf{A}_0^{[i+1]}\mathbf{A}_1^{[i+1]}\mathbf{A}_2(\mathbf{D}^{[i]})^\top + \mathbf{A}_0^{[i+1]}\mathbf{A}_1^{[i+1]}\mathbf{A}_2\Phi^{[i]}(\mathbf{A}_0^{[i+1]}\mathbf{A}_1^{[i+1]}\mathbf{A}_2)^\top\right)\right).$$

And,

$$\mathbf{H}_0^{[i+1]} = \operatorname{argmin}_{\mathbf{H}_0}\left(\frac{K}{2}\operatorname{tr}\left(\mathbf{R}^{-1}\Gamma^{[i]} - \mathbf{B}^{[i]}(\mathbf{H}_0\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]})^\top \right.\right.$$
$$\left.\left. -\mathbf{H}_0\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}(\mathbf{B}^{[i]})^\top + \mathbf{H}_0\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}\Sigma^{[i]}(\mathbf{H}_0\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]})^\top\right)\right),$$

$$\mathbf{H}_1^{[i+1]} = \operatorname{argmin}_{\mathbf{H}_1}\left(\frac{K}{2}\operatorname{tr}\left(\mathbf{R}^{-1}\Gamma^{[i]} - \mathbf{B}^{[i]}(\mathbf{H}_0^{[i+1]}\mathbf{H}_1\mathbf{H}_2^{[i]})^\top \right.\right.$$
$$\left.\left. -\mathbf{H}_0^{[i+1]}\mathbf{H}_1\mathbf{H}_2^{[i]}(\mathbf{B}^{[i]})^\top + \mathbf{H}_0^{[i+1]}\mathbf{H}_1\mathbf{H}_2^{[i]}\Sigma^{[i]}(\mathbf{H}_0^{[i+1]}\mathbf{H}_1\mathbf{H}_2^{[i]})^\top\right)\right),$$

$$\mathbf{H}_2^{[i+1]} = \operatorname{argmin}_{\mathbf{H}_2}\left(\frac{K}{2}\operatorname{tr}\left(\mathbf{R}^{-1}(\Gamma^{[i]} - \mathbf{B}^{[i]}(\mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]}\mathbf{H}_2)^\top \right.\right.$$
$$\left.\left. -\mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]}\mathbf{H}_2(\mathbf{B}^{[i]})^\top + \mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]}\mathbf{H}_2\Sigma^{[i]}(\mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]}\mathbf{H}_2)^\top\right)\right).$$

where $\operatorname{tr}(\cdot)$ denotes the trace operator, $\mathbf{Q}$ is covariance matrix for the process noise, and $\mathbf{R}$ is the covariance matrix for the observation noise.

Due to the quadratic form of the above problems, the update of each deep factor takes a closed form:

$$\mathbf{A}_0^{[i+1]} = \mathbf{D}^{[i]}(\mathbf{A}_2^{[i]})^\top(\mathbf{A}_1^{[i]})^\top\left(\mathbf{A}_1^{[i]}\mathbf{A}_2^{[i]}\Phi^{[i]}(\mathbf{A}_2^{[i]})^\top(\mathbf{A}_1^{[i]})^\top\right)^{-1},$$
$$\mathbf{A}_1^{[i+1]} = \left((\mathbf{A}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{A}_0^{[i+1]}\right)^{-1}(\mathbf{A}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{D}^{[i]}(\mathbf{A}_2^{[i]})^\top$$
$$\left(\mathbf{A}_2^{[i]}\Phi^{[i]}(\mathbf{A}_2^{[i]})^\top\right)^{-1},$$
$$\mathbf{A}_2^{[i+1]} = \left((\mathbf{A}_1^{[i+1]})^\top(\mathbf{A}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{A}_0^{[i+1]}\mathbf{A}_1^{[i+1]}\right)^{-1}(\mathbf{A}_1^{[i+1]})^\top$$
$$(\mathbf{A}_0^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{D}^{[i]}(\Phi^{[i]})^{-1}.$$

And,

$$\mathbf{H}_0^{[i+1]} = \mathbf{B}^{[i]}(\mathbf{H}_2^{[i]})^\top\left(\mathbf{H}_1^{[i]}\right)^\top(\mathbf{H}_1^{[i]}\mathbf{H}_2^{[i]}\Sigma^{[i]}(\mathbf{H}_2^{[i]})^\top(\mathbf{H}_1^{[i]})^\top)^{-1},$$
$$\mathbf{H}_1^{[i+1]} = \left((\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{H}_0^{[i+1]}\right)^{-1}(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\mathbf{H}_2^{[i]})^\top$$
$$\left(\mathbf{H}_2^{[i]}\Sigma^{[i]}(\mathbf{H}_2^{[i]})^\top\right)^{-1},$$
$$\mathbf{H}_2^{[i+1]} = \left((\mathbf{H}_1^{[i+1]})^\top(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{H}_0^{[i+1]}\mathbf{H}_1^{[i+1]}\right)^{-1}(\mathbf{H}_1^{[i+1]})^\top$$
$$(\mathbf{H}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\Sigma^{[i]})^{-1}.$$

4.4 On the fly training

A limitation of the EM strategy is that it requires reprocessing *the full dataset* to compute the updates for the state and observation model parameters. On top of being computationally cumbersome, this strategy implicitly assumes the static values of these parameters over time during the processing of the whole sequence which may not be a well-suited practice due to the lack of stationarity in the data. Indeed, tweeter trends are expected to evolve over time. Moreover, in real-time applications, users may want rapid feedback on the hate intensity evolution. We thus propose here a strategy to make DESSERT suitable for *online training*, reminiscent of recent implementations of stochastic majorization-minimization algorithms [5]. We set a window (or mini-batch) size $\alpha \geq 1$. At each time step k, the static parameters are estimated using the last α observations contained in the set $\mathbf{X}_k = \{\mathbf{x}_j\}_{j=k-\alpha+1}^{k}$. In particular, we run RTS only on the recent α observed data; then we update the parameters using the smoothing results. This sliding-window strategy presents two advantages. First, introducing α leads to faster processing. Second, it also allows better modeling of piece-wise linear processes that vary faster. The price to pay is that a smaller number of observations also limits the estimation capabilities. For initializing the RTS iterations, we use a warm start strategy. We set the parameters to their last updated values for the next timestamp. We initialize the mean and covariance of the state at $k - \alpha + 1$ using the smoothing results from the last update of the parameters in this window. Note that if $\alpha = K$, the algorithm goes back to the original offline version.

4.5 Forecasting

As explained in Section 4.1, we proceed in a sliding window fashion. For each particular observed window $\mathbf{X}_k$, the training of DESSERT allows to extract features and finds the update for parameters by iterating alternatively EM steps. After stabilization of the EM iterations (for this task, 10 iterations were sufficient for the EM to reach convergence), we can then use the output to predict the hate intensity of upcoming chunk of replies. More precisely, for every $k \in \{0, \ldots, K - \alpha\}$, we apply DESSERT on $\{\mathbf{x}_j\}_{k \leq j \leq k+\alpha}^{k}$ which provides the estimate,

$$\widehat{\mathbf{x}}_{k+\alpha+1} = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_{k+\alpha}^{-}, \tag{10}$$

associated with a covariance matrix $\mathbf{S}_{k+\alpha}$, for the immediate next reply chunk indexed by $k + \alpha + 1$. Note that although DESSERT will perform the prediction on the whole 5-dimensional vector, we will particularly be interested in the ability of our model to perform prediction on a single entry of the vector of interest, i.e., the overall hate intensity of the chunk $\mathcal{H}(\mathcal{R}_{k,k+\Delta}^{\tau})$.

4.6 Model confidence

Due to the probabilistic approach in Kalman filter, our DESSERT model can estimate the confidence score associated with the prediction. DESSERT indeed provides distribution of the next observation conditioned to the previously seen data, which is also called predictive distribution of the observation. Such probabilistic validation allows to quantify how much the model is (un)certain while predicting the next chunk hate intensity. Let us express, for every chunk index k, the distribution of the forecasted $\widehat{\mathbf{x}}_k$ given the past observations:

$$p(\widehat{\mathbf{x}}_k|\mathbf{x}_{1:k-1}) = \mathcal{N}\left(\widehat{\mathbf{x}}_k; \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k^{-}, \mathbf{S}_k\right), \tag{11}$$

where $\mathbf{S}_k = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2((\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)\mathbf{P}_{k-1}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3)^\top + \mathbf{Q}) + \mathbf{R}$, and $\mathbf{z}_k^{-}$, $\mathbf{P}_k$ are byproducts of the Kalman filter, defined in Section 4.2 (see [10] for more details about the predictive distribution of the observations). In our case, we would like to quantify our confidence score about the prediction given by the model for the hate intensity. More precisely, we focus on forecasting the sum of hate intensity, as defined in Section 3.3, for the next chunk of replies, i.e., predicting the first entry of $\widehat{\mathbf{x}}_k$, denoted by $\widehat{\mathbf{x}}_k[0]$. The value of the first row/column of S_k, denoted by $\mathrm{S}_k[0,0]$, gives us the uncertainty quantification about such prediction. Unless otherwise stated, DESSERT puts 95% of confidence on $\widehat{\mathbf{x}}_k[0]$ to belong to the interval $[\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k^-][0] \pm \mathrm{S}_k[0,0]$. We can even go further in the analysis. For instance, we can define the confidence score about an increase of the hate intensity as:

$$\widehat{p}_k = \int_{\widehat{\mathbf{x}}_k[0]}^{+\infty} \mathcal{N}\left(y; \left[\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k^-\right][0], \mathrm{S}_k[0,0]\right) dy \tag{12}$$

$$= 1 - \mathrm{CDF}(\widehat{\mathbf{x}}_k[0] \,|\, \left[\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k^-\right][0], \mathrm{S}_k[0,0]), \tag{13}$$

where CDF denotes the cumulative distribution function for the multivariate Gaussian model. Equation 12 quantifies the probability that the hate intensity will grow in the next time step. Once we have determined $\widehat{p}_k$ for every chunk index k, we can validate the method by using the standard cross-entropy loss as:

$$\text{log-loss} = \frac{1}{K}\sum_{k=1}^{K} -\left(L_k[i]\log(\widehat{p}_k)\right), \tag{14}$$

where $L_k \in \{0,1\}$ represents the ground-truth at time k for increase (i.e., $L_k = 1$) /decrease (i.e., $L_k = 0$) of $\mathbf{x}_k[0]$ from $k-1$ to k.

4.7 Time complexity

▷ **Training.** Let us express the complexity for training DESSERT in a given window of length α. To this aim, we rely on the complexity analysis for Kalman-based approaches available in [26], which leads to a complexity of $O(\alpha N_z^{2.376})$. ▷ **Testing.** The testing phase just amounts to evaluating the multi-linear equation (Equation 10). This has complexity of $O(N_x N_z^2)$ for each chunk. It can be reduced to $O(N_z^2)$ if we want to forecast only one feature (which is the case in our setting, i.e., only the overall hate intensity).

5 DATASET DESCRIPTION

For our problem setting, we require reply threads of tweets. Since existing Twitter API does not provide the entire reply thread of a tweet, we first identified source tweets and systematically scraped timelines of users who replied to these tweets. To identify the source tweets, we searched with hashtags related to 2020 US presidential elections (e.g., #MagaTerrorists, #StopTheSteal), COVID-19 (e.g., #ChinaVirus, #TrumpVirus), the Brexit referendum in the UK (e.g., #brexitshambles, #stopbrexit), and other political issues in the US, the UK and India (e.g., #BorisResign, #ToryCovidCatastrophe, #CAA, #NRC). We focused on the US and the UK more to avoid tweets in regional/code-mixed languages. The data collection was carried out during October 2020 to January 2021. The complete dataset contains 4,533 reply threads. In the Appendix, Figures 6(a) and 6(b) show the length distribution of reply threads and the distribution of the number of unique users involved in reply threads, respectively. The average thread length is 213, and the average number of unique users per thread is 187.

In Section 7.2, we study how the hate intensity of reply threads varies across *different types* of source tweets such as hate, fake, and controversial (these labels were provided by Logically's content moderators). Each reply thread is split into two parts: first 70% replies are used for training and remaining 30% for forecasting.

Model	r	RMSE ↓	MAPE (%) ↓	SMAPE (%) ↓
ARIMA	0.138	0.584	70.17	54.73
LSTM	0.331	0.515	76.53	46.34
CNN	0.251	0.454	54.68	43.40
N-Beats	0.322	0.388	47.25	39.94
DeepAR	0.308	0.386	48.95	38.56
TFT	0.511	0.413	45.88	40.39
ForGAN	*0.557*	*0.397*	*43.47*	*38.58*
DESSERT (1 layer)	0.671	0.342	32.28	35.28
DESSERT (2 layers)	0.665	0.394	32.69	35.66
DESSERT (3 layers)	**0.670**	**0.332**	**31.08**	**34.01**

Table 1: Overall performance (↓: lower value is better).

6 BASELINE METHODS

To the best of our knowledge, there is no prior work on forecasting hate intensity of a reply threads on Twitter. We, therefore, consider statistical and deep learning based time-series and temporal pattern modeling approaches.

ARIMA. It is one of the classical time-series prediction models based on auto regressive integrated moving average.

LSTM. We use a stacked 2-layer LSTM with 50 cells each. We use a dense layer with ReLU activation for final prediction [9].

CNN. We use a 1D Convolutional layer with 64 filters and a kernel size of 2. This is followed by a max-pooling layer which feeds result into a fully-connected layer with ReLU activation for final prediction [24].

N-Beats. It uses the idea that the base architecture should be simple and generic, yet expressive (deep). It does not rely on time-series specific feature engineering or input scaling [28].

DeepAR. It aims to produce probabilistic forecasts specifically on a large number of related time series [34].

TFT. Temporal Fusion Transformers is an attention-based architecture which combines high-performance multi-horizon forecasting with interpretable insights into temporal dynamics [19].

ForGAN. It utilizes conditional generative adversarial network to learn the data generating distribution and compute probabilistic forecasts [18].

7 EXPERIMENTAL RESULTS

For comparison, we use one correlation-based metric (higher is better) – **Pearson correlation coefficient** (r), and three metrics for error calculation (lower is better) – **Root mean square error** (RMSE), **Mean Absolute Percentage Error** (MAPE) and **Symmetric Mean Absolute Percentage Error** (SMAPE).

Default setup. Unless otherwise mentioned, we consider the following setup as default: $w = 0.5$, $\Delta = 10$, $\alpha = 20$, N_z= 5, N_x=5 and DESSERT with 3 layers, and hyperparameters $\mathbf{Q} = \sigma_Q^2\mathbf{I}$, $\mathbf{R} = \sigma_R^2\mathbf{I}$, $\mathbf{P}_0 = \sigma_P^2\mathbf{I}$, with $(\sigma_Q, \sigma_R, \sigma_P) = (10^{-5}, 10^{-1}, 10^{-1})$, $\bar{z}_0 = \mathbf{0}$ ($\mathbf{I}$: identity matrix), and the hate speech classifier C by Founta et al. [13].

7.1 Overall performance

Table 1 shows the performance of the competing models. Clearly, DESSERT outperforms all the baselines with a significant margin. DESSERT (3 layers) achieves 0.113 points gain in Pearson's r, 0.065 points drop in RMSE, and 12.39% and 4.57% drop in MAPE and

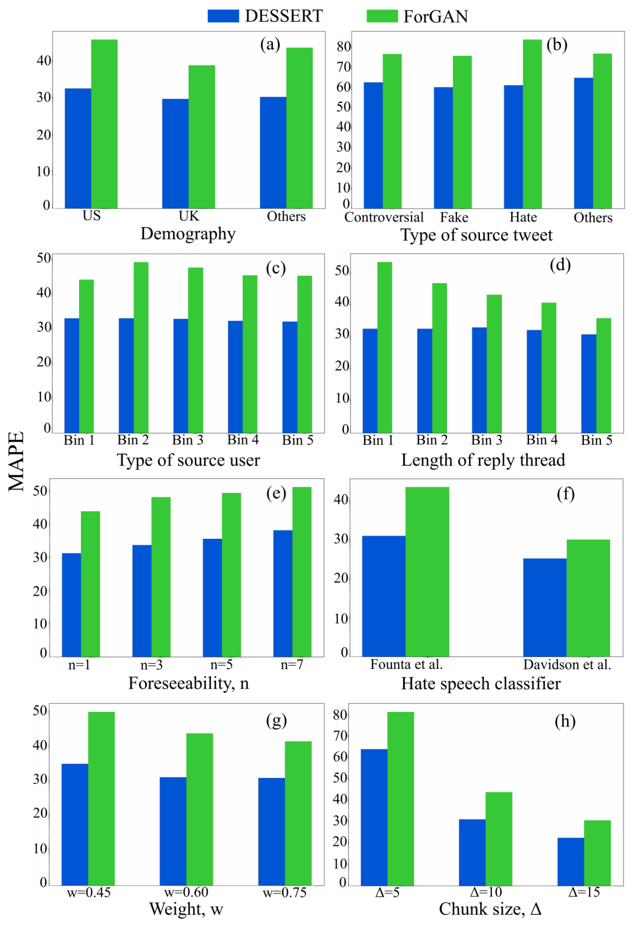


Figure 3: Performance (MAPE) of DESSERT and ForGAN w.r.t (a) demography, (b) types of source tweets, (c) types of source users, (d) length of reply threads, (e) foreseeability, (f) hate speech classifiers (Founta et al. [13] and Davidson et al. [7]), (g) weight w, and (h) chunk size Δ. See the Appendix for the correlation-based (Pearson's r) performance.

SMAPE respectively compared to ForGAN (the best baseline). Among the baselines, ARIMA and LSTM/CNN perform the worst.

An advantage of DESSERT over other deep learning based baselines is that it estimates a confidence score along with the prediction (which statistical signal processing models also produce). As discussed earlier in Section 4.6, DESSERT helps to estimate the probabilistic validation. One can estimate the (un)certainty of the prediction of an increase/decrease of the hate intensity, from the log-loss value (Equation 14). Smaller is the loss, more accurate is the model for its prediction of an increase/decrease of hate intensity. This information helps the practitioners to understand the impact, in terms of hate causality, for a given thread. For instance, in our dataset, the log-loss associated to a thread (i.e., 'Hate' category, c.f. Section 7.2) is 2.51, meaning that the consequences of this category of thread is highly predictable. It is expected as 'Hate' content will tend to generate more hateful reactions. In contrast, the broader category 'Others', corresponding to another thread, seems more difficult to assess, having a log-loss of 5.38.

7.2 Detailed analysis

Further, we delve deeper into the results of DESSERT and the best baseline (ForGAN) to understand how they generalize.

Results across demography. Figure 3(a) shows how DESSERT and ForGAN perform on the reply threads of the source tweets originated from different countries – the US, the UK, and others (India, Brazil, Australia). DESSERT is consistent across geographic locations.

Types of Source tweets. The content moderation team at Logically further annotated 1830 randomly selected source tweets into 'fake' (518), 'hate' (582), 'controversial' (550) and 'others' (180). We aim to understand how the models perform for different types of source tweets. Once again, DESSERT shows steady performance across different types (c.f. Figure 3(b)).

Types of source users. One may wonder if the prestige of online users drives the toxicity of the reply thread. To check this, we divide the reply threads into five equal bins based on the follower count of the users who posted the source tweets. Figure 3(c) shows that unlike ForGAN, DESSERT is agnostic to types of the source users.

Length of reply threads. One may wonder how the forecasting varies with the overall length of the reply threads. Here, we divide the reply threads into five equal bins based on the length of the threads. Figure 3(d) shows that although ForGAN improves with the increase of length, DESSERT remains highly consistent.

Foreseeability & early prediction. So far, we have considered reply tweets till t and reported the prediction for the immediate reply thread, i.e., $\mathcal{R}^{\tau}_{t,t+\Delta}$. Here, we are interested to check that with the same training data, how far the models can predict. We further allow the models to predict for $\mathcal{R}^{\tau}_{t+(n-1)\Delta,t+n\Delta}$, with $n = 1, 3, 5, 7$. Also, it captures how early our model can predict. Figure 3(e) shows that DESSERT does not deteriorate much even at $n = 7$.

Hate speech classifier. To compute hate intensity, we have used a hate speech classifier C in Equation 1. Here, we check how the models respond if we change C. We replace our default C [12] by the hate speech detection method proposed by Davidson et al. [7] and observe that DESSERT still outperforms ForGAN (Figure 3(f)).

Varying w. In Eq. 1, the weight w balances the effect of the hate speech classifier and the lexicon to measure hate intensity. Figure 3(g) shows that DESSERT's accuracy does not depend much on w.

Varying Δ. Figure 3(h) shows that with the increase of chunk size Δ, both ForGAN and DESSERT improve.

7.3 Ablation study

Varying layers. Table 1 shows that DESSERT performs the best with three layers.

Varying α. In DESSERT, the parameter α indicated how much historical data we consider for prediction. As expected, Figure 4(a) shows that increasing α results in better prediction.

Changing training size. We experiment with three different sizes of the training set – 50%, 60% and 70%. Fig. 4(b) shows that unlike ForGAN, DESSERT remains effective with varying training size.

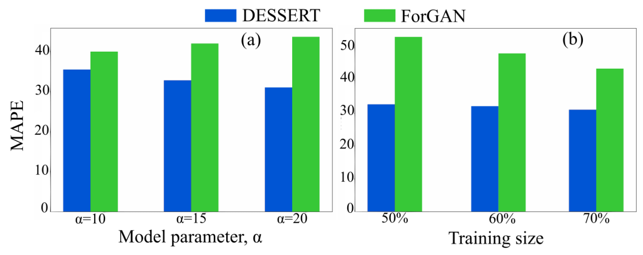


Figure 4: Model ablation w.r.t. (a) α and (b) training size.

8 CONCLUSION

In this paper, we attempted to predict the hate intensity of upcoming replies for a source tweet. We employed a series of time-series forecasting models and showed that they do not perform well on the dataset we curated. We further proposed DESSERT, a deep state-space model that significantly outperforms all the baselines. Detailed study further brought out how DESSERT generalizes w.r.t. several decision choices. Further, DESSERT has been deployed in Logically's advanced AI platform for monitoring online problematic hateful content and recommend countermeasures to minimise its reach and reduce the damage (see Appendix). In future, we intend to capture user metadata and graph-level signals to enhance the accuracy.

REFERENCES

[1] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep Learning for Hate Speech Detection in Tweets. In *WWW*. 759–760.

[2] A. Bagavathi, P. Bashiri, S. Reid, M. Phillips, and S. Krishnan. 2019. Examining Untempered Social Media: Analyzing Cascades of Polarized Conversations. In *ASONAM*. 625–632.

[3] G.E.P. Box and G.C. Tiao. 1975. Intervention analysis with applications to economic and environmental problems. *J. Amer. Statist. Assoc.* 70, 349 (1975), 70–79.

[4] Emilie Chouzenoux and Víctor Elvira. 2020. GraphEM: EM algorithm for blind Kalman filtering under graphical sparsity constraints. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*. IEEE, 5840–5844.

[5] Emilie Chouzenoux and Jean-Christophe Pesquet. 2017. A Stochastic Majorize-Minimize Subspace Algorithm for Online Penalized Least Squares Estimation. *IEEE Transactions on Signal Processing* 65, 18 (2017), 4770–4783.

[6] Yi-Ling Chung, Elizaveta Kuzmenko, Serra Sinem Tekiroglu, and Marco Guerini. 2019. CONAN - COunter NArratives through Nichesourcing: a Multilingual Dataset of Responses to Fight Online Hate Speech. In *ACL*. 2819–2829.

[7] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated Hate Speech Detection and the Problem of Offensive Language. In *ICWSM*. 512–515.

[8] Hang Thi-Thuy Do, Huy Duc Huynh, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen, and Anh Gia-Tuan Nguyen. 2019. Hate Speech Detection on Vietnamese Social Media Text using the Bidirectional-LSTM Model. *aeXiv:1911.03648* (2019).

[9] Steven Elsworth and Stefan Güttel. 2020. Time series forecasting using LSTM networks: A symbolic approach. *arXiv preprint arXiv:2003.05672* (2020).

[10] V. Elvira, J. Míguez, and P. .M. Djurić. 2017. Adapting the Number of Particles in Sequential Monte Carlo Methods through an Online Scheme for Convergence Assessment. *IEEE Transactions on Signal Processing* 65, 7 (2017), 1781–1794.

[11] Paula Fortuna and Sérgio Nunes. 2018. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–30.

[12] Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior. In *ICWSM*. 491–500.

[13] Antigoni Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, and Ilias Leontiadis. 2019. A Unified Deep Learning Architecture for Abuse Detection. In *WebSci*. 105–114.

[14] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas. 2020. Exploring Hate Speech Detection in Multimodal Publications. In *WACV*. 1459–1467.

[15] Barbara Hammer. 2000. On the approximation capability of recurrent neural networks. *Neurocomputing* 31, 1-4 (2000), 107–123.

[16] M Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. 2018. NSE stock market prediction using deep-learning models. *Procedia computer science* 132 (2018), 1351–1362.

[17] Muhammad Okky Ibrohim and Indra Budi. 2019. Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter. In *Proceedings of the Third Workshop on Abusive Language Online*. 46–57.

[18] Alireza Koochali, Peter Schichtel, Andreas Dengel, and Sheraz Ahmed. 2019. Probabilistic forecasting of sensory data with generative adversarial networks–forgan. *IEEE Access* 7 (2019), 63868–63880.

[19] Bryan Lim, Sercan Ömer Arik, Nicolas Loeff, and Tomas Pfister. 2019. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *CoRR* abs/1912.09363 (2019). arXiv:1912.09363

[20] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*. 13153–13164.

[21] Sarah Masud, Subhabrata Dutta, Sakshi Makkar, Chhavi Jain, Vikram Goyal, Amitava Das, and Tanmoy Chakraborty. 2020. Hate is the New Infodemic: A Topic-aware Modeling of Hate Speech Diffusion on Twitter. *arXiv preprint arXiv:2010.04377* (2020).

[22] Binny Mathew, Navish Kumar, Pawan Goyal, and Animesh Mukherjee. 2020. Interaction Dynamics between Hate and Counter Users on Twitter. In *CoDS COMAD*. 116–124.

[23] Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection. *arXiv preprint arXiv:2012.10289* (2020).

[24] Sidra Mehtab, Jaydip Sen, and Subhasis Dasgupta. 2020. Robust Analysis of Stock Price Time Series Using CNN and LSTM-Based Deep Learning Models. In *ICECA*. 1481–1486.

[25] Jozef Mivskolci, Lucia Kovávcová, and Edita Rigová. 2020. Countering Hate Speech on Facebook: The Case of the Roma Minority in Slovakia. *Social Science Computer Review* 38, 2 (2020), 128–146.

[26] Corey Montella. 2011. *The Kalman filter and related algorithms: A literature review.* Technical Report. https://www.researchgate.net/publication/236897001_The_Kalman_Filter_and_Related_Algorithms_A_Literature_Review.

[27] Marzieh Mozafari, Reza Farahbakhsh, and Noël Crespi. 2020. A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. In *Complex Networks and Their Applications VIII*, Hocine Cherifi, Sabrina Gaito, José Fernendo Mendes, Esteban Moro, and Luis Mateus Rocha (Eds.). 928–940.

[28] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437* (2019).

[29] Fabio Poletto, Valerio Basile, Manuela Sanguinetti, Cristina Bosco, and Viviana Patti. 2020. Resources and benchmark corpora for hate speech detection: a systematic review. *LREC* (2020).

[30] Jing Qian, Anna Bethke, Yinyin Liu, Elizabeth Belding, and William Yang Wang. 2019. A Benchmark Dataset for Learning to Intervene in Online Hate Speech. In *EMNLP-IJCNLP*. 4755–4764.

[31] M. H. Ribeiro, Pedro H. Calais, Yuri A. Santos, V. Almeida, and W. Meira. 2018. Characterizing and Detecting Hateful Users on Twitter. In *ICWSM*.

[32] Hammad Rizwan, Muhammad Haroon Shakeel, and Asim Karim. 2020. Hate-Speech and Offensive Language Detection in Roman Urdu. In *EMNLP*. 2512–2522.

[33] Ignacio Rojas, Olga Valenzuela, Fernando Rojas, Alberto Guillén, Luis Javier Herrera, Héctor Pomares, Luisa Marquez, and Miguel Pasadas. 2008. Soft-computing techniques and ARMA model for time series prediction. *Neurocomputing* 71, 4-6 (2008), 519–537.

[34] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020), 1181–1191.

[35] Simo Särkkä. 2013. *Bayesian filtering and smoothing.* Number 3. Cambridge University Press.

[36] Shalini Sharma, Angshul Majumdar, Víctor Elvira, and Emilie Chouzenoux. 2020. Blind Kalman Filtering for Short-term Load Forecasting. *IEEE Transactions on Power Systems* 35, 6 (2020), 4916–4919.

[37] Serra Sinem Tekiroglu, Yi-Ling Chung, and Marco Guerini. 2020. Generating Counter Narratives against Online Hate Speech: Data and Strategies. In *ACL*. Online, 1177–1190.

[38] Milo Trujillo, Maurício Gruppi, Cody Buntain, and Benjamin D. Horne. 2020. What is BitChute? Characterizing The. In *HT*. 139–140.

[39] Michael Wiegand, Josef Ruppenhofer, Anna Schmidt, and Clayton Greenberg. 2018. Inducing a Lexicon of Abusive Words – a Feature-Based Approach. In *NAACL*. 1046–1056.

[40] Savvas Zannettou, Barry Bradlyn, Emiliano De Cristofaro, Haewoon Kwak, Michael Sirivianos, Gianluca Stringini, and Jeremy Blackburn. 2018. What is Gab: A Bastion of Free Speech or an Alt-Right Echo Chamber. In *WWW*. 1007–1014.

(Appendix / Supplementary Material)

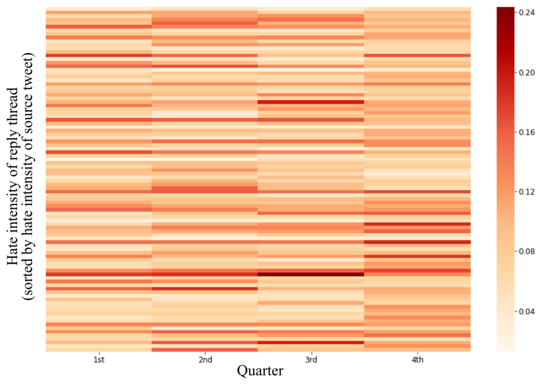


Figure 5: Heat map showing the hate intensity per quarter of the reply threads (sorted by the hate intensity of the source tweets). We divide each reply thread into four equal quarters and measure hate intensity per quarter. It indicates that there is no single quarter which is always more hateful than the others across reply threads.

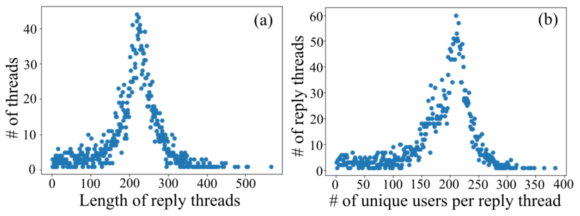


Figure 6: Distribution of (a) the length of reply threads and (b) the number of unique users involved in reply threads.

A ADDITIONAL RESULTS

Figure 6 shows the distribution of the reply thread length and the number of unique users per reply thread. Figure 5 shows that there is not single quarter of the threads which is most hateful across all the threads. This indicates that the forecasting is not straightforward, and there is no underlying function which models the hate intensity pattern. Figure 7 shows the detailed analysis of the results in terms of Pearson's r. Figure 8 shows the ablation results in terms of Pearson's r.

B EXPECTATION-MINIMIZATION

The training of DESSERT requires the estimation of the matrices A_0, A_1, A_2 and H_0, H_1, H_2 jointly with the state. This problem can be solved efficiently using an expectation-maximization (EM) algorithm [35], that alternates classic Kalman filtering/smoothing and the update of matrix parameters, with the aim of reaching the maximum likelihood (ML) estimator of those parameters. With the proposed DESSERT model in Eq. 4, the EM consists in searching for A_0, A_1, A_2 and H_0, H_1, H_2 maximizing $p(A_0, A_1, A_2, H_0, H_1, H_2|\mathbf{x}_{1:K})$ or, equivalently, maximizing $\varphi_K(A_0, A_1, A_2, H_0, H_1, H_2)$ $= \log p(\mathbf{x}_{1:K}|A_0, A_1, A_2, H_0, H_1, H_2)$. The EM algorithm is a type of majorization-minimization [5] approach. It allows to compute a lower bound Q of the marginal likelihood, satisfying that, for any A_0, A_1, A_2, and H_0, H_1, H_2, $\varphi_K(A_0, A_1, A_2, H_0, H_1, H_2)$ $\geq Q(A_0, A_1, A_2, H_0, H_1, H_2; \Theta^{[i]})$, where $\Theta^{[i]}$ gathers the outputs of the RTS smoother at previous EM iterate i, listed in Equation 9. The application of [35, Theo.12.4] to our DESSERT model in Equation 4 and the cancellation of constant terms, leads to the function

$$\begin{aligned}
Q(A_0, A_1, A_2, H_0, H_1, H_2, \Theta^{[i]}) = \\
&- \frac{K}{2} \mathrm{tr}\Big(Q^{-1}\Sigma^{[i]} - D^{[i]}(A_0A_1A_2)^\top - A_0A_1A_2(D^{[i]})^\top \\
&\qquad + A_0A_1A_2\Phi^{[i]}(A_0A_1A_2)^\top\Big) \\
&- \frac{K}{2} \mathrm{tr}\Big(R^{-1}\Gamma^{[i]} - B^{[i]}(H_0H_1H_2)^\top - H_0H_1H_2(B^{[i]})^\top \\
&\qquad + H_0H_1H_2\Sigma^{[i]}(H_0H_1H_2)^\top\Big).
\end{aligned}$$

The proposed update in Section 4.3 amounts to maximizing $Q(\cdot, \Theta^{[i]})$, for $i = 1, 2, \ldots$, using a coordinate descent algorithm, i.e.,

$$\begin{aligned}
A_0^{[i+1]} &= \mathrm{argmin}_{A_0} - Q(A_0, A_1^{[i]}, A_2^{[i]}, H_0^{[i]}, H_1^{[i]}, H_2^{[i]}, \Theta^{[i]}) \\
A_1^{[i+1]} &= \mathrm{argmin}_{A_1} - Q(A_0^{[i+1]}, A_1, A_2^{[i]}, H_0^{[i]}, H_1^{[i]}, H_2^{[i]}, \Theta^{[i]}) \\
A_2^{[i+1]} &= \mathrm{argmin}_{A_2} - Q(A_0^{[i+1]}, A_1^{[i+1]}, A_2, H_0^{[i]}, H_1^{[i]}, H_2^{[i]}, \Theta^{[i]}) \\
H_0^{[i+1]} &= \mathrm{argmin}_{H_0} - Q(A_0^{[i+1]}, A_1^{[i+1]}, A_2^{[i+1]}, H_0, H_1^{[i]}, H_2^{[i]}, \Theta^{[i]}) \\
H_1^{[i+1]} &= \mathrm{argmin}_{H_1} - Q(A_0^{[i+1]}, A_1^{[i+1]}, A_2^{[i+1]}, H_0^{[i+1]}, H_1, H_2^{[i]}, \Theta^{[i]}) \\
H_2^{[i+1]} &= \mathrm{argmin}_{H_2} - Q(A_0^{[i+1]}, A_1^{[i+1]}, A_2^{[i+1]}, H_0^{[i+1]}, H_1^{[i+1]}, H_2, \Theta^{[i]}),
\end{aligned}$$

which leads to the six sub-problems provided in Section 4.3. This procedure is theoretically sound (see [4, 35] for more details). In particular, it is guaranteed to yield a monotonic increase of the marginal log-likelihood function φ_K and convergence to a stationary point of it.

C HYPER-PARAMETERS

ARIMA. We use the python library pmdarima. For the hyper-parameters (*start_p*, *start_q*, *max_p*, *max_q*, *m*, *start_P*, *d* and *D*), we set them to 0, 0, 5, 5, 12, 0, *false*, 0 and 1 respectively. The non-seasonal version of ARIMA works best for our task.
LSTM. We use a stacked LSTM with 2 layers, each with 50 units followed by a dense layer for final prediction. We use ReLU for activation in both layers. The model is trained on a batch size 16 for 100 epochs. We add an early stopping layer to avoid overfitting.
CNN. We use a CNN with 1-D convolutional layer of filter size 64 and kernel size of 2. We pass it through a max-pooling 1D layer of pool size 2. We train the model on a batch size of 16 for 100 epochs. To avoid overfitting, we use early stopping.
N-Beats. We use the default implementation and run the model in the interpretable mode, with the 3 blocks per stack, 3 fully-connected layers per block of width chosen from [32, 512] (whichever works best). We set *log_interval*, *log_val_interval*, *weightdecay* to 10, 1, 0.01 respectively. We run the trainer for a maximum of 100 epochs on a batch size of 16.

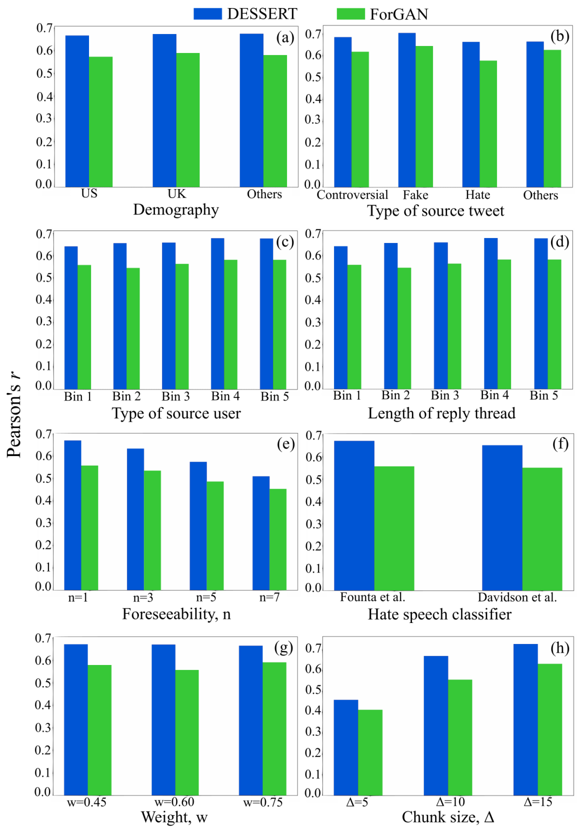


Figure 7: Performance (Pearson's r) of DESSERT and ForGAN w.r.t (a) demography, (b) types of source tweets, (c) types of source users, (d) length of reply threads, (e) foreseeability, (f) hate speech classifiers (Founta et al. [13] and Davidson et al. [7]), (g) weight w, and (h) chunk size Δ.

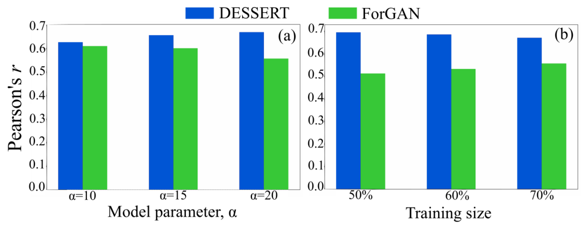


Figure 8: Model ablation w.r.t. (a) α and (b) training size.

DeepAR. We use the default implementation for this model, with 2 RNN layers having LSTM cells with the hidden recurrent size as 10. We set *log_interval*, *log_val_interval*, *weightdecay* to 10, 1, 0.01 respectively. We run the trainer for a maximum of 200 epochs on a batch size of 16.

TFT. We use the default implementation, with the number of LSTM layers set to 2 and the hidden size of the network set to 8. We set the *log_interval*, *log_val_interval* and *weightdecay* to 10, 1, 0.01 respectively. The model runs for a maximum of 200 epochs.

ForGAN. We use the default architecture, with LSTM layers of sizes 8 and 64 for the RNN based generator and discriminator, respectively. The model runs for a maximum of 800 epochs.

DESSERT. Thanks to our warm start strategy, the initialisation of hyper-parameters $\mathbf{A}_0^{[0]}, \mathbf{A}_1^{[0]}, \mathbf{A}_2^{[0]}$ and $\mathbf{H}_0^{[0]}, \mathbf{H}_1^{[0]}, \mathbf{H}_2^{[0]}$ is only required for the very first chunk of the training set, i.e., for $k = t = 0$. In practice, we initialized $\mathbf{A}_0^{[0]}, \mathbf{A}_1^{[0]}, \mathbf{A}_2^{[0]}$ with identity matrix, and $\mathbf{H}_0^{[0]}, \mathbf{H}_1^{[0]}, \mathbf{H}_2^{[0]}$ with uniform independent entries in $[0, 1]$. All results are averaged over a set of four random initializations. Moreover, we empirically tuned hyper-parameters σ_Q, σ_R, σ_P, N_z, N_x and window size α to 10^{-5}, 10^{-1}, 10^{-1}, 5, 5, 20 respectively.

D DEPLOYMENT DETAILS

Logically's advanced AI platform is a real-world system that can ingest and analyse data from millions of media sources as well as social media posts. Proprietary models and custom pipelines in the platform harness state-of-the-art machine learning and NLP to identify and analyse online problematic harmful content at-scale. Built on cutting-edge, secure and highly scalable cloud infrastructure, the platform brings together Logically's capabilities in granular analysis, classification and detection of damaging harmful content, its origins and impact. It also provides access to a diverse set of stakeholders in different market segments, a suite of countermeasures to tackle identified problematic content by leveraging in-house expert analysts in fact-checking and OSINT research.

The technology side of the platform is uniquely developed by implementing state-of-the-art in AI research and industry best practices in software architecture and engineering. In order to achieve high-quality analytical throughput on large volumes of data, the platform leverages cutting-edge cloud technologies such as Kubernetes, which works with a range of container tools including Docker to make software applications highly scalable. On the other hand, the platform applies state-of-the-art AI research to customise and iteratively evolve its methodologies for effective modelling of heterogeneous (articles, social media posts, multimedia) data sources to build reliable AI models that can augment its expert intelligence network of editors, fact checkers, content moderators and OSINT analysts.

Deployment of multiple AI models is possible in the platform as it implements micro services architecture to empower a range of products built by Logically with AI-based insights through an ensemble of REST based machine learning services. This feature of the platform offered the flexibility to integrate DESSERT as a REST micro-service to evaluate its hate speech intensity annotations alongside the company's proprietary models and custom pipelines for problematic hateful content analysis. Further, the evaluation revealed that the intensity scores from DESSERT offer additional knowledge to the proprietary models for ranking and prioritization of high risk online harms for enforcement of countermeasures to minimize their impact and damage. We plan to further test DESSERT extensively on multiple heterogeneous social media data streams ingested by the platform to understand its abilities to generalise in accurately and reliably detecting hate speech patterns. Also we intend to evaluate DESSERT and its future enhanced versions in the AI platform to extract insights to better detect custom online harms across domains such as health, finance and geopolitics.