Forecasting Hate Intensity of Reply Threads on Twitter

by
Snehil

Under the Supervision of
Dr. Tanmoy Chakraborty
and
Dr. Md. Shad Akhtar

Indraprastha Institute of Information Technology
Delhi
July, 2021

Forecasting Hate Intensity of Reply Threads on Twitter

by
Snehil

Submitted
in partial fulfillment of the requirements for the
degree of
Master of Technology

to

Indraprastha Institute of Information Technology
Delhi
July, 2021

Certificate

This is to certify that the thesis titled "**Forecasting Hate Intensity of Reply Threads on Twitter**" being submitted by **Snehil** to the Indraprastha Institute of Information Technology Delhi, for the award of the Master of Technology, an original research work carried out by her under our supervision. In our opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree.

The results contained in this thesis have not been submitted in part or full to any other university or institute for the award of any degree/diploma.

July, 2021

Dr. Tanmoy Chakraborty
Department of Computer Science & Engineering
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

Dr. Md. Shad Akhtar
Department of Computer Science & Engineering
Indraprastha Institute of Information Technology Delhi
New Delhi 110020

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.
In this thesis, 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 comprising the entire reply chains of a $\sim 4.5k$ root tweets catalogued into four controversial topics. Our preliminary analysis confirms that the evolution patterns along time of hate intensity among reply chains have highly diverse patterns, and there is no significant correlation between the hate intensity of the source tweets and that of their reply chains. We notice a handful of such cases where despite the root tweets being non-hateful, the succeeding replies inject an enormous amount of toxicity into the discussions. We employ several state-of-the-art dynamic models and show that they fail to forecast the hate intensity.
We then propose DESSERT a novel deep state-space model trained in real time. The model leverages the function approximation capability of deep neural networks with the capacity to quantify the uncertainty of statistical signal processing models. We observe that the DESSERT outperforms all the baselines across four evaluation metrics (both correlation-based and error-based). This model achieves 0.67 Pearson's r and 31.08 Mean Absolute Percentage Error (MAPE), which is significantly better than the best baseline (r=0.557, MAPE=43.47).
In addition to this, we also address this problem through a deep stratified learning framework DRAGNETİt groups hate intensity profiles of reply chains into clusters used to formulate prior knowledge, which is employed to predict hate intensity of upcoming replies for a new reply chain. The DRAGNET turns out to be highly effective, significantly outperforming six baselines. It beats the best baseline with an increase of 9.4% in the Pearson correlation coefficient and a decrease of 19% in Root Mean Square Error.
Further, both the models' 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.

Acknowledgements

I express my profound gratitude to my advisor Dr. Tanmoy Chakraborty without whom my thesis would not have been possible. I am thankful to him for providing guidance and opportunity to work on real world problem. His expertise in the fields of Natural Language Processing and Network Science was invaluable in formulating the research methodology. His constant encouragement, suggestions and unconditional support was a major driving force to bring my thesis to a higher level. I would also like to thank Dr. Md. Shad Akhtar for his continuous guidance and suggestions.
I feel very pleased to have been able to collaborate with Shalini Sharma and Dr. Angshul Majumdar on Deep State Space Modelling. I would also like to acknowledge the support of Logically in making it possible to deploy this thesis work in real work scenario.
Many thanks to Dhruv Sahnan and Vasu Goel for working along side me on this research problem. Their continuous efforts and enthusiasm has been very inspiring. Special thanks to Sarah Masud for her constant support and sparing time for brain storming sessions with me. It is an immense pleasure to express my thanks to all the members of Laboratory for Computational Social Systems for their motivation and patient support.
To conclude I would like to dedicate this thesis to my family who have been the cornerstone of strength during my weak moments. I would like to thank them for their constant encouragement, support and belief in me. I would also like to thank my friends and college mates for motivating and supporting me.

Snehil

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

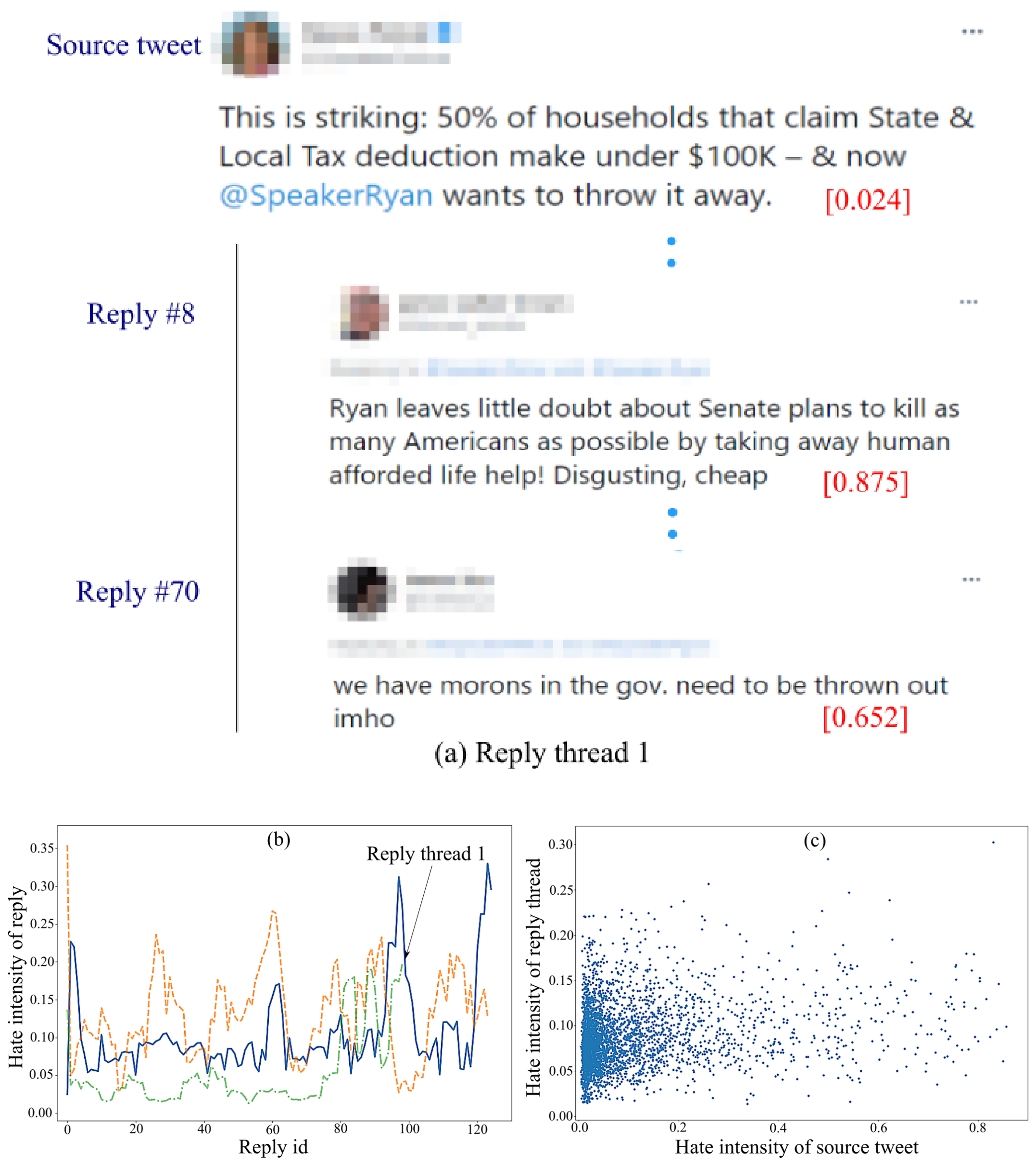
Online discussion forums such as Twitter, and Reddit, provide users with the opportunity to express their opinion freely and that too, anonymously. Although freedom of speech enables users to seek, express and impart information about ideas and opinions, its misuse often leads to social instability both online and offline, resulting in cyber-crimes such as hate speech and cyberbullying. Hate crimes around the world have intensified owing to the rampant onslaught of provocative content; there have been violent incidents ranging from lynchings [28] to even ethnic cleansing. While a handful of corporate firms are continuously attempting to reconfigure the terms and conduction of social media usage, they are also constrained by domestic laws on censorship.

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. The urgent need to combat hate speech has been recognized, resulting in a handful of methods to *detect* online hate speech [38]. Attempts have also been made to study how hate speech *breeds* on social media [31] – how to mathematically model hate speech propagation [30], who is likely to engage in hate speech [41], and so on. However, as Liu et al. [26] highlighted, the aforementioned methods consider a *reactive* strategy – given an online post, these methods attempt to detect whether it is hate speech or not; sometimes, they also classify a hateful post into a fine-grained category (offensive, provocative, aggressive, etc.). One may think of developing methods for the *early detection* of hate speech; however, it is unknown how expeditiously its adverse effects manoeuvre and manipulate online users upon exposition. The promptness and the extent of damage that an online post can cause is hard to quantify and predict.

1.2 Motivation and Observations for the Thesis

1.2.1 Benign Posts can Manifest Hate Speech

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 1.1(a) shows a (partial) reply threads along with the hate intensity (defined in Section 3.1) 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



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 1.1: (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).

1.1(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.

The retrospective nature of the state-of-the-art hate speech detection models has been a major bottleneck for the content moderators to intervene before the online hate crime takes place. A better strategy would be to *proactively prevent* the hate crime from happening, in place of letting it materialize and then detecting it.A conversation around a benign post often bypasses the lens of the content moderators, and therefore receives ample opportunity for inviting hate speech perpetrators to adversely act upon it. Figure 1.2 shows the *hate intensity* (defined in Section 3.1) of the reply thread of tweets related to 'Donald Trump's COVID crisis' and 'Joe Biden's campaign'. Two perceptible observations are as follows: (i) The root tweets and their initial replies are not very hate intensive; however, as the discussions progress, the hate intensity score rises. (ii) There is no consistent pattern across the hate intensity profiles of reply threads from two topics. The former observation confirms why a proactive (prevention) strategy is a requisite, while the latter shows the non-triviality of the hate intensity prediction problem. Furthermore, we believe that combating online hate speech is not as simple as a binary classification problem (hate vs not-hate); rather, it is about the *intensity of hatred* that a post can exhibit. Predicting the intensity of hatred that stems from the root tweet and its reply thread is crucial for a content moderator to prioritize which reply thread needs greater manual inspection and further intervention.

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

1.2.2 Hate Intensity Profiles of Reply Threads are Diverse

In this thesis, we solve a novel problem – *given a root tweet and a few of its initial replies, can we predict the hate intensity of the upcoming replies of the tweet*? We begin by suitably quantifying the hate intensity of a tweet (and a reply thread) and then converting a reply thread into a sequence of hate intensities.. 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 1.1(b)). Moreover, the hate intensity of reply threads is not correlated with that of the source tweet (c.f. Figure 1.1(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 A.1 in the Appendix).

1.2.3 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.

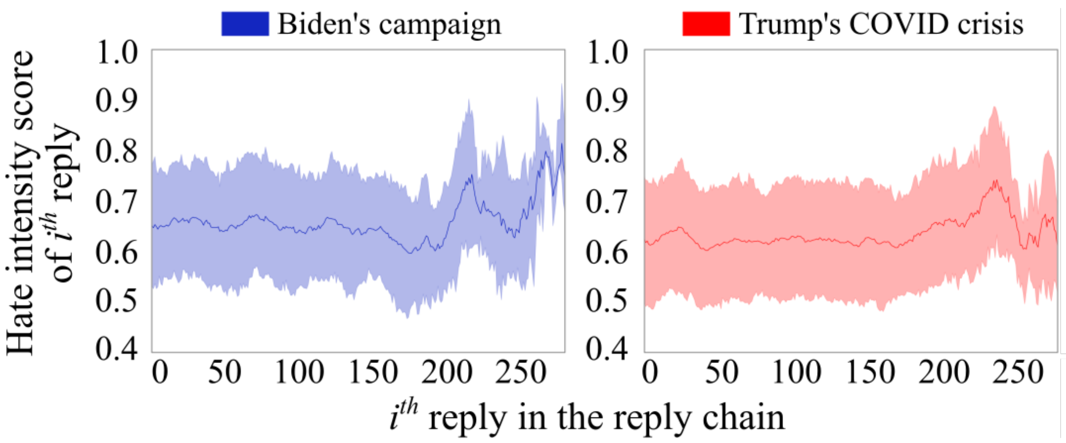


FIGURE 1.2: Temporal change of hate intensity score in the reply threads related to two topics – Joe Biden's campaign and Donald Trump's COVID crisis. Solid lines (shaded regions) signify the average (confidence) over related reply threads.

1.3 Our Proposed Approaches

1.3.1 DRAGNET

We then propose a novel model, called DRAGNET. It works underneath a stratified learning framework which aims to capture the heterogeneity amongst the hate intensity profiles of reply threads and to categorize them into homogeneous clusters/strata. We train DRAGNET to predict the weights representing the cluster representative probabilities. The prior knowledge vector formulated by unifying the cluster information and the predicted weights, is used in cohesion with the new reply thread to predict the hate intensity of upcoming replies. **Experimental results.**We then present a detailed comparative analysis of DRAGNET against six baselines (adopted to the current setting). DRAGNET proves to be the best model, outperforming the strongest baseline with a 9.4% higher Pearson correlation, a 19% lower Root Mean Square Error (RMSE) and a 6.6% lower Mean Forecast Error (MFE) (for latter two metrics, lower value is better). Further analysis on the implication in the change of model parameters and various ablation studies help us diagnose DRAGNET. in terms of efficiency, consistency and robustness.

1.3.2 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 [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

[1]**DE**ep **S**tate-**S**pac**E** model for hate intensity prediction of **R**eply **T**hreads.

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 [23] ($r = 0.557$, 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.

1.4 Related Work

Summary. *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.*

1.4.1 Studies on Hate Speech

Plethora of techniques have been proposed for the detection of online hate speech in various languages [2, 3, 18, 27, 36, 37]. An overview is provided in [10]. These methods cover lexicon-based logistic regression [9, 50] and multi-modality by capturing different aspects of hate [22]. While deep learning based models have shown efficacy of detection [1, 16], recent studies are now focusing on building explainable systems [21]. Overcoming the use of static hate-lexicon by prototypical learning has also been explored [39]. Note that the objective of our research is not to propose a new hate detection model; however, this work depends on a hate detection model (Section 4.2.3). Readers are encouraged to go through [15] for a detailed survey on hate detection models. Mathew et al. [31] performed an exploratory analysis of Gab [56], and established the presence of strong ties (cohesiveness) among users participating in hateful content. This behaviour has since been independently verified for other platforms as well [49]. Masud et al. [30] further proposed a predictive model to determine the followers who are more likely to retweet a hateful post. Another aspect of user interaction is the reply thread. Our analysis of reply thread on Twitter uncovered a nuanced and fluctuating sentiment of hate and non-hate, which we explore in this work.

1.4.2 Time-series models

Since the hate intensity of a reply thread can be mapped as a time series, we consider several time-series forecasting (TSF) models as baselines. The conventional methods for TSF such as ARMA [42], including exponential smoothing and linear space models, achieved superlative performance. In many applications, these models are used to estimate one-step-ahead prediction.These modelling strategies manouvre the structural assumptions of temporal data of being linear ,sesonal and stationary which do not hold true for real-world time series which are very volatile in nature i.e. have unanticipated distribution changes. Deep learning based models such as CNNs, RNNs and LSTMs have become popular for TSF because they do not assume prior structure of data. RNNs [51] are capable of retaining information of preceding results; however, their performance degrades with the increasing length of the sequence. LSTM [11] overcomes the long-term dependency problem to a certain extent. Some of the recent studies used sequence-to-sequence (Seq2Seq) models which align with our interest of multi-step forecasting [13, 29]. New architectures were proposed for the same in [24, 46] to remit the error buildup in multi-step forecasting. Transformer-based models like N-Beats [35] have also been successful for time-series data.

To handle uncertainty in real-world time-series data, many probabilistic forecasting models were proposed, one of them being DeepAR [44]. Deep learning based state space models have also been in certain applications because of there probabilistic construct. Some studies leverage Generative Adversarial Network (GAN) to estimate prediction distributions using cVAEs [24], cGANs [55] and ForGAN [23]. However, they fail to encode the multiplicity in time series. In this study, we consider most of the models discussed above as baselines.

Chapter 2

Dataset and Baselines

2.1 Input Corpus

To the best of our knowledge, there is no existing dataset that contains *complete* reply threads on Twitter. Since the Twitter API does not furnish to fetch the entire reply thread for a particular tweet, we opt for an alternative method. We manually identify tweets corresponding to various real-world events; we essentially perform a hashtag-based search related to the 2020 US presidential election, the Brexit referendum in the UK and various other political issues particularly in the US, the UK and India. The dataset comprises $\sim$ 4500 root tweets (with their reply threads), a total of $\sim$ 1.1 million tweets and $\sim$ 950k unique users across all threads. Table 2.1 shows that our curated dataset consists of tweets from various geographical locations spanning multiple topics over which mass discussions took place during and before our collection process.

We further map the extracted reply threads to the following major topics: (i) *Donald Trump's COVID crisis*: root tweets posted as official statements from the Trump administration, public reactions of Donald Trump's handling of COVID-19 and controversial claims of Twitter users about the situation in the USA; (ii) *Joe Biden's campaign*: root tweets posted during the 2020 US presidential campaign, particularly with Joe Biden as the person of interest, consisting of controversial claims about Biden's history, rumours about his campaign's claims and some official statements from his campaign team; (iii) *Brexit referendum*: root tweets posted by the people of the UK expressing their opinions about the Brexit situation, some about the mistrust in Boris Johnson's administration and the Conservative (Tory) party; (iv) *COVID-19 impelled xenophobia - specifically Sinophobia*: this subset contains the Twitter mentions of COVID-19, from across the world, as the "China virus", wherein the reply threads hold the Chinese community responsible for the global pandemic.

Length of the reply thread. Figure 2.1 shows that the reply threads are of different length, with an average (maximum) length close to 200 (566) replies in the entire dataset. These numbers are quite similar for the four topics.

Number of unique users per reply thread. Figure 2.1 also illustrates that the distribution of the number of unique users in the reply threads appears to be very

Geolocation	Hashtag / Keyword
United States of America	#TrumpVirus, #CreepyJoe, #MAGA, MAGA terrorist, biden not my president
United Kingdom	brexit, #BrexitShambles, tory, #RejoinEU, boris, #Tories
India	#NRC, #CAA, Sushant Singh Rajput
Other	china virus, chinese virus, covid crisis, #COVID19

TABLE 2.1: List of hashtags and keywords used to curate the threads by location.

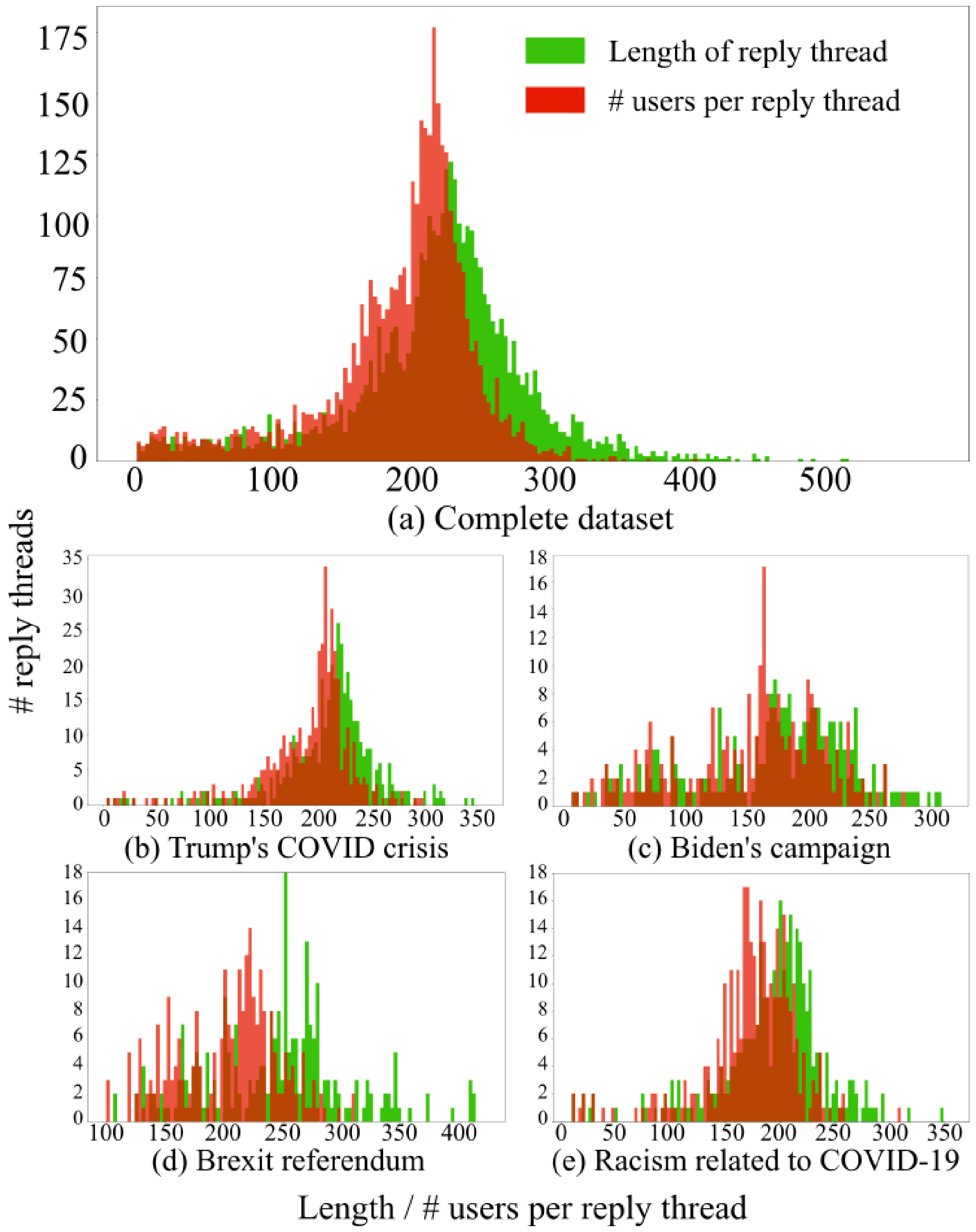


FIGURE 2.1: Distributions of the length of the reply threads and the number of unique users per reply thread (a) in the complete dataset and (b-e) across topics.

similar to the distribution of lengths of the reply threads for the overall dataset and across topics. This indicates that there is a high percentage of unique users in a reply thread (on an average, 88% of replies in a reply thread originate from unique users), i.e., a single user avoids taking part in the same reply thread multiple times.

Distribution of the lifetime of threads. Figure 2.2 highlights the correlation between the percentage of tweets per thread and the amount of time it takes to cover them. We see that 72% of the threads attract 90% of their total replies within the first 24 hours (1 day). It jumps up to 90% when we account for 72 hours (3 days). Interestingly, only 60% of the threads terminate within the first week, while more

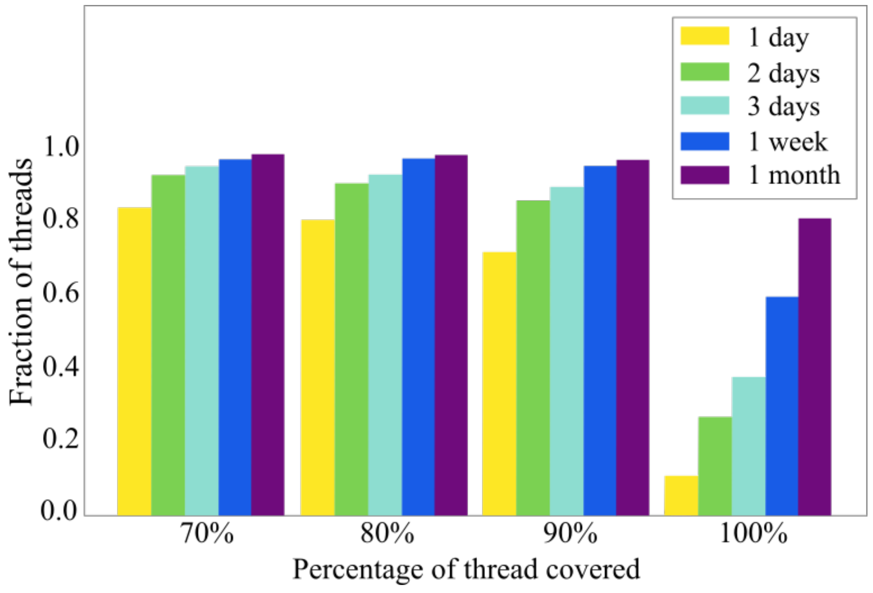


FIGURE 2.2: Lifetime of reply threads. We show how much time a thread takes to grow upon the posting of the root tweet.

than 20% of the threads do not reach full length even after the first month of the root tweet being posted. Therefore, a large fraction of the activity in reply threads is observed within just the first 3 days.

Dissimilarity in the hate intensity profiles. As explained in Section 3.1, we introduce the concept of "windowing" to smoothen the hate intensity profile by introducing a rolling average on the hate intensity profile of the reply thread. Figure 2.3 illustrates the distribution of the pairwise Dynamic Time Warping (DTW)[1] distances over the dataset. We observe that the pairwise DTW distance distribution has a huge range and a high variance; the median value lies close to 5 which is quite high given that the range of values in the hate intensity profiles is $[0 - 1]$. This further indicates that there is no coherent pattern across hate intensity profiles, neither in the complete dataset nor within the topic-specific datasets, indicating the non-triviality of the problem under study.

2.2 Baseline Methods

To our knowledge, there is no prior work on forecasting hate intensity of reply threads on Twitter. Since this problem is readily mapped to a time series prediction model, we therefore, adopt both classical and deep learning based time-series forecasting and temporal pattern modeling approaches.

- **ARIMA:** It is a statistical time-series prediction model that captures different standard temporal structures in time series data using auto regressive integrated moving average.
- **LSTM:** We use a stacked LSTM with 2-layers of 50 cells each. Finally, we use a dense layer with ReLU activation to obtain predictions [11].
- **CNN:** We use 1-dimensional CNN architecture followed by a fully-connected layer with ReLU activation. For the convolutional layer we use 64 filters and a kernel size of 2 [32].

[1]See [34] for the formulation of DTW.

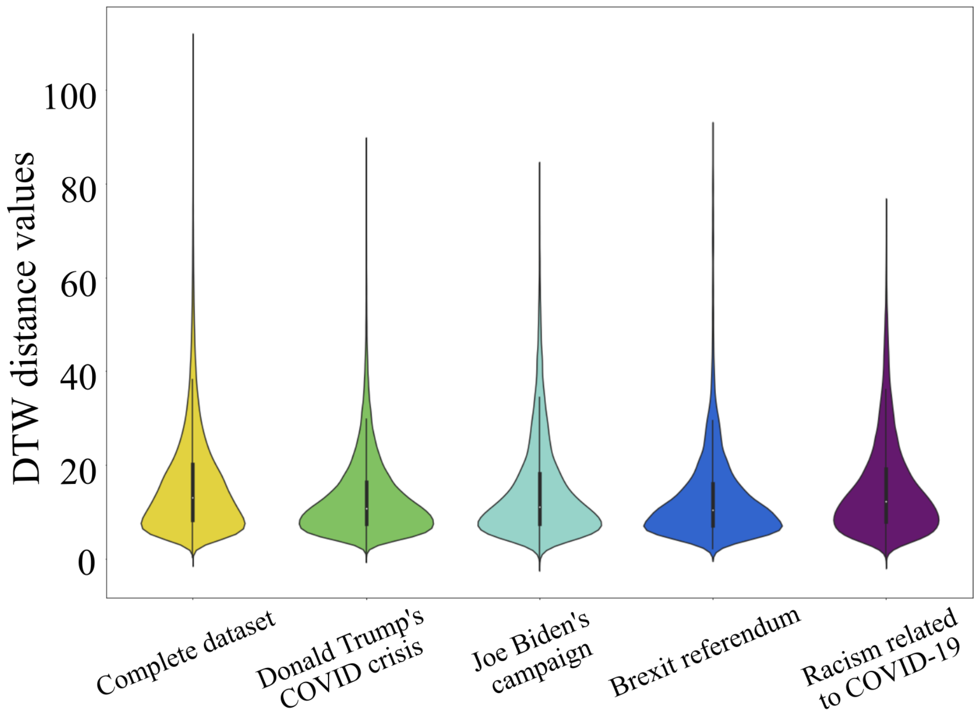


FIGURE 2.3: Violin distribution for the pairwise DTW distances of reply threads.

- **N-Beats:** It is a deep learning based model used for univariate time-series forecasting. It relies on forward and backward residual links. [35].
- **DeepAR:** It is an auto-regressive recurrent network primarily used for time-series forecasting [44].
- **TFT:** It is deep neural network architecture for multi-horizon forecasting using self-attention [25].
- **ForGAN:** It is a conditional GAN based model designed to learn data distribution with modules for feature selection. It is used for probabilistic time-series forecasting. [23].

Chapter 3

Deep Stratified Learning Based Model

3.1 Preliminaries

Table 3.1 summarizes the denotations of the important notations. Let an ordered sequence of first t replies to a root tweet φ be $\mathcal{T}^{\varphi}_{1,t} = \langle c^{\varphi}_1, c^{\varphi}_2, \ldots, c^{\varphi}_t \rangle$, where c^{φ}_i denotes the i^{th} reply to the tweet. (Note that t denotes an integer index associated to t^{th} reply in the sequence, not the actual/continuous time.) For each reply c, we quantify its hate intensity using $\mathcal{H}(.)$ which is a weighted sum of two measures,

$$\mathcal{H}(c) = w\mathcal{H}_c(c) + (1-w)\mathcal{H}_l(c) \tag{3.1}$$

where w ($0 \leq w \leq 1$) is a hyper-parameter. $\mathcal{H}_c$ refers to the probability that the reply is hateful as indexed by a state-of-the-art hate speech detection model[1] (Section 3.3.1). $\mathcal{H}_l$ is defined as the average score for all words in a reply from a model-independent hate lexicon that comprises 2,895 words (scores are normalised using min-max scaling) as proposed in [52]. Since, $0 \leq \mathcal{H}_c(c) \leq 1$ and $0 \leq \mathcal{H}_l(c) \leq 1$, therefore, $0 \leq \mathcal{H}(c) \leq 1$. Each reply thread $\mathcal{T}^{\varphi}_{1,t}$ can be mapped to a sequence of hate intensities, $\mathcal{H}(\mathcal{T}^{\varphi}_{1,t}) = \{\mathcal{H}(c^{\varphi}_1), \cdots, \mathcal{H}(c^{\varphi}_t)\}$.

We further smooth each such sequence $\mathcal{H}(\mathcal{T}^{\varphi}_{1,t})$ using a *rolling average operation* with window size δ. A **window** is a set of δ consecutive replies to a root tweet φ. The hate intensity of a window consisting of a sequence of replies $\mathcal{T}^{\varphi}_{k,k+\delta}$ for a tweet φ is measured as,

$$\begin{aligned} \mathcal{H}(\mathcal{T}^{\varphi}_{k,k+\delta}) &= \sum_{c \in \mathcal{T}^{\tau}_{k,k+\delta}} \mathcal{H}(c) \\ &= w \sum_{c \in \mathcal{T}^{\varphi}_{k,k+\delta}} \mathcal{H}_c(c) + (1-w) \sum_{c \in \mathcal{T}^{\varphi}_{k,k+\delta}} \mathcal{H}_l(c). \end{aligned} \tag{3.2}$$

Note that $0 \leq \mathcal{H}(\mathcal{T}^{\varphi}_{k,k+\delta}) \leq \delta$.

Sentiment features: To capture sentimental context flow in a new reply thread w.r.t. the initial tweet, we calculate the cosine similarity between the sentiment embedding of the root tweet φ and its corresponding replies, $c^{\varphi}_1, c^{\varphi}_2, \ldots$, $\mathcal{CS}(c^{\varphi}_i) = CosineSim(Embed(c^{\varphi}_i), Embed(\varphi))$. The sentiment embedding is the second last fully-connected layer from the pre-trained XLNet model [54] for sentiment classification. We apply the same *rolling average operation* to $\mathcal{CS}(\mathcal{T}^{\varphi}_{1,t})$ with the same window size

[1]We use the Davidson model [9] as the default hate speech classifier. However, we also show the results with other hate speech classifiers in Section 3.3.2.

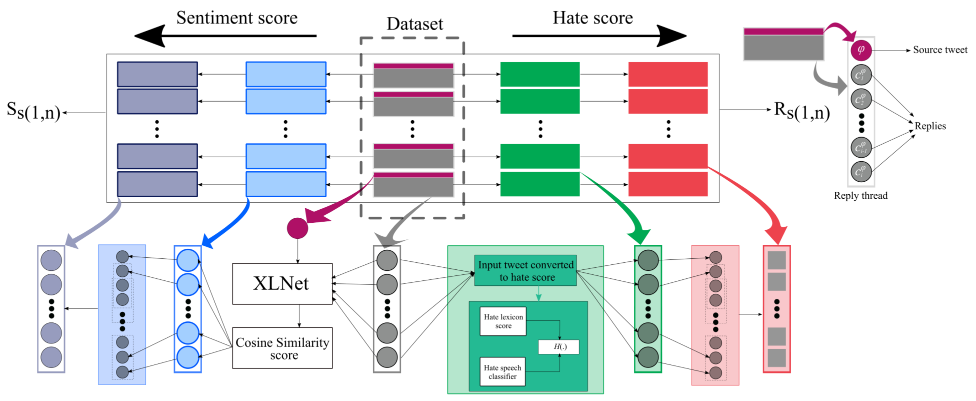


FIGURE 3.1: Schematic diagram of the data transformation module. $\mathcal{S}_{s_{(1,n)}}$ is the sequence of cosine similarity values (calculated between the sentiment embedding of root tweet φ and its corresponding sequence of replies) for all reply threads in the dataset, and $\mathcal{R}_{s_{(1,n)}}$ is the set of all hate intensity profiles.

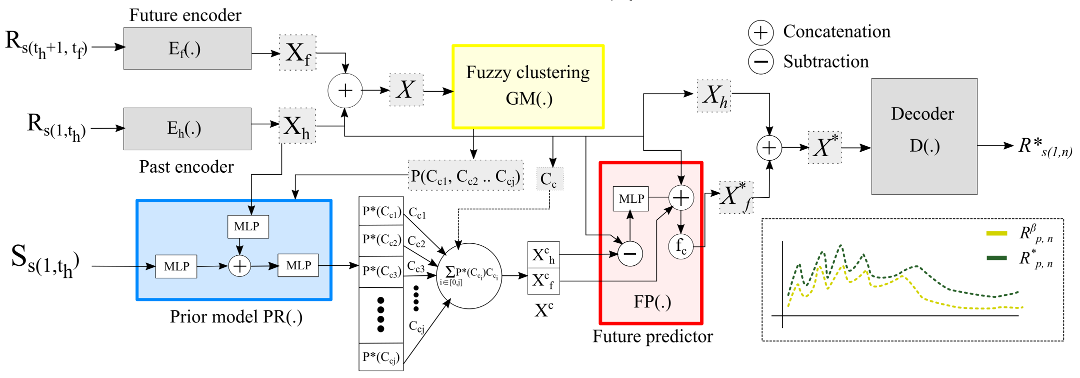


FIGURE 3.2: Overall architecture of DRAGNET. After training the autoencoder, the concatenated history and future latent representations are clustered using a fuzzy clustering algorithm. For a new reply thread, the future hate intensity profile is predicted using (i) the history latent representation, (ii) the sentiment similarity sequence of the history, and (iii) the prior knowledge vector extracted from $\mathcal{PR}(.)$.

δ as performed on $\mathcal{H}(\mathcal{T}^{\varphi}_{1,t})$. Figure 3.1 illustrates the complete data preprocessing pipeline.

Problem definition: Given a new tweet φ and the last t_h replies in its reply thread $\mathcal{T}^{\varphi}_{1,t_h}$ (used as a training set or history), we aim to predict the hate intensity of the upcoming replies $c^{\varphi}_{t'}$ (where $t' > t_h$). However, instead of predicting the hate intensity per reply, we consider each window of δ future replies and predict the hate intensity per window, $\mathcal{H}(\mathcal{T}^{\varphi}_{t',t'+\delta})$.

3.2 Our Proposed Model: DRAGNET

In this section, we explain our proposed method, DRAGNET[2] for hate intensity prediction. DRAGNET is a deep stratified learning [20] approach, which first divides the heterogeneous data points (reply threads in our case) into homogeneous clusters/strata and then trains a deep regressor on each strata to predict the hate intensity. The schematic diagram of DRAGNET is shown in Figure 3.2.

[2]**D**eep st**R**atified le**A**rnin**G** for hate i**N**tensity of r**E**ply threads on **T**witter

Symbol	Description
φ	A tweet
c_i^{φ}	i^{th} reply to the tweet
$\mathcal{R}_{p,q}$	p^{th} datapoint where q denotes the length of its hate intensity profile
$\mathcal{R}^*{}_{p,q}$	The recreated p^{th} datapoint of the dataset from the autoencoder with q denoting the length of its hate intensity profile
$\mathcal{S}_{s_{(1,n)}}$	Sequence of cosine similarity values
$\mathcal{R}_{s_{(1,n)}}$	Set of all hate intensity profiles
j	Number of clusters
δ	Window size
t_h	Number of replies in history
t_f	Index of the last reply in the reply thread
n	Maximum length of the reply thread
s	Total number of reply threads
N_{X_h}	Size of encoded history latent vector
N_{X_f}	Size of encoded future latent vector
$\mathcal{X}_h$	Encoded history latent vector
$\mathcal{X}_f$	Encoded future latent vector
$\mathcal{X}^*{}_f$	Predicted future latent vector
$\mathcal{X}$	Latent vector of $\mathcal{R}_{p,q}$
$\mathcal{X}^*$	Latent vector of $\mathcal{R}^*{}_{p,q}$
C_c	List of cluster centres
$\mathcal{P}(C_{ci})$	Likelihood of belongingness to i^{th} cluster identified with C_{ci}
$\mathcal{X}^c$	Prior knowledge
$\mathcal{P}^*(C_{ci})$	Predicted weight for i^{th} cluster centre C_{ci} to calculate $\mathcal{X}^c$
$\mathcal{X}_d$	Pre-processed prior vector
$\mathcal{X}_{hc}$	Intermediate prediction vector
$\mathcal{GM}(.)$	Clustering model
$\mathcal{PR}(.)$	Classification model
$\mathcal{FP}(.)$	Prediction model
$\mathcal{FP}_d(.)$	1^{st} segment of prediction model
$\mathcal{FP}_p(.)$	2^{nd} Prediction model
$\mathcal{D}(.)$	Decoder model

TABLE 3.1: Notations and denotations.

The training set is formed by the two dimensional vector of window-wise hate intensity profile and sentiment context value sequences,

$$\begin{aligned}
\mathcal{R}_{s_{(1,n)}} &= \{\mathcal{R}_{p,q} : 1 \le p \le s, 1 \le q \le n\} \\
\mathcal{R}_{p,q} &= \{\mathcal{H}(\mathcal{T}^{\varphi_p}_{k,k+\delta}) : 1 \le k \le q - \delta\} \\
\mathcal{S}_{s_{(1,n)}} &= \{\mathcal{S}_{p,q} : 1 \le p \le s, 1 \le q \le n\} \\
\mathcal{S}_{p,q} &= \{\mathcal{CS}(\mathcal{T}^{\varphi_p}_{k,k+\delta}) : 1 \le k \le q - \delta\}
\end{aligned} \tag{3.3}$$

where s is the total number of reply threads, and n is the maximum length of the reply thread. The elements $\mathcal{R}_{p,q} \in \mathcal{R}_{s_{(1,n)}}$ and $\mathcal{S}_{p,q} \in \mathcal{S}_{s_{(1,n)}}$ are the p^{th} data point, whose reply thread is of length q (φ_p represents the p^{th} root tweet). DRAGNET starts by first learning low-dimensional latent representations for the hate intensity profile of reply threads (of irregular lengths) using an autoencoder. In this setting, we learn two separate latent representations – $\mathcal{X}_h$, the initial few replies which are treated as the history, and $\mathcal{X}_f$, the future hate trend for the rest of the replies. We then employ a fuzzy clustering approach in an unsupervised setting to assign cluster membership probabilities $\mathcal{P}(C_{c1}, C_{c2}, .., C_{cj})$ and cluster centres $(C_{c1}, C_{c2}, .., C_{cj})$ to each reply thread, where j is a hyper-parameter indicating the number of clusters. Following this, we train a novel deep neural network unit that predicts the cluster membership probabilities, given $\mathcal{X}_h$ and $\mathcal{S}_{s_{(1,t_h)}}$, which assigns cluster centres for a new reply thread. Finally, a novel deep regressor predicts the latent representation of the future hate trend $\mathcal{X}_f$ using $\mathcal{X}_h$ and $\mathcal{P}(C_{c1}, C_{c2}, .., C_{cj})$, which, when combined with $\mathcal{X}_h$, is converted to the complete hate trend by the decoder trained during the autoencoder phase.

3.2.1 Time-Series Representative Learning

The reply thread vector $\mathcal{R}_{s_{(1,n)}}$ can be treated as a collection of time series (window-wise hate intensity profiles) with irregular lengths. State-of-the-art methods on clustering irregular time series involve the use of the Dynamic Time Warping (DTW) distance metric and its variants to group similar trends together [34]. Even with favorable outcomes in terms of precision by DTW to map time series similarity, the noisy and volatile nature of the data points in the current study does not allow it to show high efficiency in clustering similar hate trends into a single stratum. To capture a more suitable representation of the time series, we propose an autoencoder to map each reply thread $\mathcal{R}_{p,q}$ in $\mathcal{R}_{s_{1,n}}$ to a low-dimensional latent representation. Additionally, instead of a single encoder-decoder architecture, we propose a multi-encoder approach as proposed in [48]

3.2.2 Proposed Autoencoder

We propose an autoencoder using the Inception-Time module [14] for the initial transformation stage. The autoencoder consists of dual encoders to make up the latent space and a single decoder to recreate the complete hate intensity profile from the latent representation.

Encoder. Many studies proposed models for both uni-variate and multivariate time series [14, 53]. Time-series modeling involves a transformation stage. Instead of manually engineering it, we use the Inception-Time module, denoted as $\mathcal{E}_t(.)$ [14]. It provides a multivariate transformation, which is the concatenation of various univariate time series created by using different kernel sizes along the original time series. We represent it as,

$$\mathcal{X}_m = \mathcal{E}_t(\mathcal{R}_{s_{(1,n)}}) \tag{3.4}$$

where $\mathcal{X}_m \in \mathbb{R}^{s \times n \times 4}$ is the intermediate multivariate representation of $\mathcal{R}_{s_{(1,n)}}$. The intermediate representation $\mathcal{X}_m$ is subjected to the $\mathcal{F}latten(.)$ operator that converts $\mathcal{X}_m$ to a one-dimensional vector $\mathcal{X}_{flat}$. It is followed by layers of linear transformations which are trained to learn the best representative transformation from $\mathcal{X}_{flat}$ and correspondingly produce the final representation of $\mathcal{R}_{s_{(1,n)}}$ in the latent space. If $\mathcal{E}_{lt}$ denotes the layers of linear transformations, we can write,

$$\begin{aligned} \mathcal{X}_{flat} &= \mathcal{F}latten(\mathcal{X}_m) \\ \mathcal{X}_o &= \mathcal{E}_{lt}(\mathcal{X}_{flat}) \end{aligned} \tag{3.5}$$

Here, $\mathcal{X}_o$ is the final latent space representation of the original time series $\mathcal{R}_{s_{1,n}}$.

Using Equations 3.4 and 3.5, we define two encoders $\mathcal{E}_h(.)$ and $\mathcal{E}_f(.)$. We encode two segments of each reply thread, $\mathcal{R}_{s_{(1,t_h)}}$ and $\mathcal{R}_{s_{(t_h+1,t_f)}}$ that are termed as history and future representations respectively, as follows:

$$\begin{aligned}\mathcal{X}_h &= \mathcal{E}_h(\mathcal{R}_{s_{(1,t_h)}})\\ \mathcal{X}_f &= \mathcal{E}_f(\mathcal{R}_{s_{(t_h+1,t_f)}})\end{aligned} \tag{3.6}$$

Here, $\mathcal{X}_h$ and $\mathcal{X}_f$ form the history and future latent representations, which we will use finally to decode to the complete hate trend. Note $\mathcal{X}_h \in \mathbb{R}^{s\times N_{X_h}}$ and $\mathcal{X}_h \in \mathbb{R}^{s\times N_{X_f}}$, where N_{X_h} and N_{X_f} represent the length of low dimensional vectors X_h and X_f respectively.

Decoder. Although we use dual encoders to convert the hate intensity of each reply thread into two latent representations, $\mathcal{X}_h$ and $\mathcal{X}_f$, we use a single decoder $\mathcal{D}(.)$ to convert the latent representation back to the original input. For this, we concatenate the two latent representations and train the decoder to recreate the original hate intensity profile per reply thread. The functioning of the decoder can be denoted as follows:

$$\mathcal{R}^*{}_{s_{(1,n)}} = \mathcal{D}(\mathcal{X}^*) \tag{3.7}$$

We measure the precision of predictions from the decoder by comparing $\mathcal{R}^*{}_{s_{(1,n)}}$ to $\mathcal{R}_{s_{(1,n)}}$.

3.2.3 Fuzzy Associations

As stated in Section 3.2.1, the hate intensity profiles of reply threads in our dataset are noisy and volatile in nature, and have no evident pattern. Our objective is to use low dimensional latent representations of the data procured via autoencoder (as discussed in Section 3.2.2) to group similar profiles together. Recent research that align with this approach are an attestation to the validity of performance of the deep learning based models in learning hidden features from time-series data for different tasks [6, 57]. We use a clustering approach over the latent representations to group heterogeneous hate intensity profiles into (near-)homogeneous clusters. In this unsupervised setting, the task of finding meaningful associations in data is unfairly dependent on the number of clusters j and the cluster centres. Therefore, we adopt a fuzzy clustering approach and use the membership probabilities as a feature embedding, rather than confining each profile to a single cluster.

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

$$\mathcal{X} = \mathcal{X}_h \oplus \mathcal{X}_f \tag{3.8}$$

Now, in place of employing a hard clustering approach, i.e., fixing the association of each profile to the closest cluster, we make use of cluster membership probabilities from a fuzzy clustering. The membership probability vector represents the associative probabilities of each cluster with the given thread, denoted by $\mathcal{P}(C_{c1}, C_{c2}, .., C_{cj})$, where C_{ci} denotes the cluster centre of the i^{th} cluster.

Fuzzy Clustering. Since our objective is to find associations of each hate intensity profile to the homogeneous clusters, we perform clustering on the combined latent representation $\mathcal{X}$. We adopt a state-of-the-art fuzzy clustering model [40], denoted by $\mathcal{GM}(.)$ to detect the clusters C_c which is the set of cluster centres as follows:

$$C_c = \mathcal{GM}(\mathcal{X}) = (C_{c1}, C_{c2}, .., C_{cj}) \tag{3.9}$$

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

3.2.4 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 centres, where the weights correspond to the cluster membership probabilities for the new thread, denoted by $\mathcal{P}^*(C_{c1}, C_{c2}, .., C_{cj})$. We define prior knowledge as follows:

$$\mathcal{X}^c = \sum_{i \in (0 \leq i \leq j)} C_{ci} \cdot \mathcal{P}^*(C_{ci}) \tag{3.10}$$

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 thread, we cannot use the fuzzy clustering model $\mathcal{GM}(.)$ directly. Rather, we construct a prior model $\mathcal{PR}(.)$ with the aim of predicting the membership probabilities for new threads using only the latent representation of the history $\mathcal{X}_h$ and sentiment feature $\mathcal{S}_{s_{(1,t_h)}}$.

$$\mathcal{PR}(\mathcal{E}_h(\mathcal{R}_{s_{(1,t_h)}}), \mathcal{S}_{s_{(1,t_h)}}) = \mathcal{P}^*(C_{c1}, C_{c2}, .., C_{cj}) \tag{3.11}$$

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

3.2.5 Estimating Latent Representation of Upcoming Reply Threads

Since our ultimate objective is to predict the complete hate intensity profile trend given the history $\mathcal{R}_{s_{(1,t_h)}}$, we need to estimate the latent representation of upcoming future hate intensity profile $\mathcal{X}^*{}_f$. This will finally be fed into the decoder along with the latent representation of the history. For this, we utilize the prior knowledge (as explained in Section 3.2.4), and the latent representation of the history $\mathcal{X}^c$.

To avoid the estimation task from being overly governed by the prior knowledge, we design the predictor in two stages. The first stage involves the creation of vector i.e., $\mathcal{X}_h^c$ that constitutes the information required to predict $\mathcal{X}^*{}_f$. Since we are only given $\mathcal{X}_h$, we measure the deviation of the prior from the expected only in the latent space where we encode the initial history of hate diffusion, i.e., $\mathcal{R}_{s_{(1,t_h)}}$. We calculate deviation by first applying the difference operator on the expected ($\mathcal{X}_h$) and the estimated ($\mathcal{X}_h^c$) priors of the reply thread history. We employ a single-layer perceptron, $\mathcal{FP}_d(.)$ to formulate a vector that encodes the dissimilarity between the prior knowledge from history $\mathcal{X}_h^c$ and $\mathcal{X}_h$, which is denoted as $\mathcal{X}_d$.

$$\mathcal{X}_s = \mathcal{X}_h \ominus \mathcal{X}_h^c; \qquad \mathcal{X}_d = \mathcal{FP}_d(\mathcal{X}_s) \tag{3.12}$$

We finally obtain the $\mathcal{X}_h c$ vector by concatenating the provided input $\mathcal{X}_h$, the pre-processed prior $\mathcal{X}_d$ and $\mathcal{X}_f^c$ as, $\mathcal{X}_{hc} = \mathcal{X}_h \oplus \mathcal{X}_d \oplus \mathcal{X}_f^c$.

The second stage is the deep linear transformation model $\mathcal{FP}_p(.)$ that predicts the upcoming hate intensity in the latent space $\mathcal{X}_f^*$ as follows:

$$\mathcal{X}_f^* = \mathcal{FP}_p(\mathcal{X}_{hc}) \tag{3.13}$$

3.2.6 Decoding Latent Representation

As mentioned in Section 3.2.5, $\mathcal{X}_f^*$ is the upcoming hate intensity in the latent space. We regress this low-dimensional representation to hate intensity profiles of length n, i.e., $\mathcal{R}^*{}_{s_{(1,n)}}$. To serve this purpose, we concatenate the primal versions of the history $\mathcal{X}_h$ and the predicted future $\mathcal{X}_f^*$ of reply threads.

$$\mathcal{X}^* = \mathcal{X}_h \oplus \mathcal{X}_f^* \tag{3.14}$$

where $\mathcal{X}^*$ is the predicted hate intensity profile of the upcoming reply thread in the latent space.

We contrive the decoder as a mirror of the encoder. We define the decoder as,

$$\mathcal{R}^*{}_{s_{(1,n)}} = \mathcal{D}(\mathcal{X}^*) \tag{3.15}$$

where $\mathcal{R}^*{}_{s_{(1,n)}}$ is the final predicted hate intensity profile of the reply thread. We measure the accuracy of the prediction with respect to $\mathcal{R}^*{}_{s_{(1,n)}}$.

3.2.7 Implementation Details

We train our proposed autoencoder by backpropagating L2 loss to learn low-dimensional representations for the hate intensity profiles. Note that the decoder $\mathcal{D}(.)$ has a transpose architecture w.r.t. the encoder $\mathcal{E}_h(.)$, which is why the decoder also takes special indices as input for the decoding process. These special indices returned by the Inception module of the encoder are used for the *unmax-pooling operation*. We apply the Gaussian Mixture model for fuzzy clustering on the concatenated latent representations.

$\mathcal{PR}(.)$ is a 3-layered deep neural network unit, where the last layer employs Sigmoid activation. We consider the output of $\mathcal{PR}(.)$ to be the estimated weights $\mathcal{P}^*(C_{c1}, C_{c2}, .., C_{cj})$, as explained in detail in Section 3.2.4. $\mathcal{FP}_p(.)$ is another 3-layered feed forward neural network that performs linear transformation and estimates the future latent space $\mathcal{X}_f^*$.

3.3 Experimental Results and Analysis

3.3.1 Experiment Setup

We consider the following hyper-parameters as default: $w = 0.6$, $\delta = 10$, $t_h = 35$, $t_f = 284$, $n = 300$, $j = 15$, $N_{X_h} = 32$, $N_{X_f} = 128$ and `DRAGNET` with Inception-Time module [14] in the autoencoder for transformation step with kernel sizes $[5, 7, 9]$. Davidson's model [9] is considered as the default hate speech detection model (see Section 3.1). A Gaussian Mixture model with covariance type **full** is used for fuzzy clustering. We use 80-20 split for training and testing. In Section 3.3.2, we will show how the model responds to the change in the major hyper-parameters.

Model	r	RMSE ↓	MFE ↓
LSTM	0.145	0.611	0.500
CNN	0.105	0.644	0.509
DeepAR	0.310	0.484	0.065
TFT	*0.469*	*0.437*	*0.076*
N-Beats	0.380	0.544	0.085
ForGAN	0.240	0.603	0.360
DRAGNET w/o Sentiment	0.515	0.286	0.018
DRAGNET	**0.563**	**0.247**	**0.010**

TABLE 3.2: Overall performance (↓: lower value is better). Second last row indicates the performace of DRAGNET without the sentiment feature. The best baseline is italicized.

For the purpose of evaluation, we use three metrics - **Pearson Correlation coefficient** (r), **Root Mean Square Error** (RMSE) and **Mean Forecast Error** (MFE)[3]. For the former metric, higher value is better, whereas for the latter two, lower value is better.

3.3.2 Results and Analysis

In this section, we perform a detailed comparative analysis of the performance of the competing models. We also show how DRAGNET and the best baseline respond with respect to changes in the – hate detection model used to measure the hate intensity, demography and topic of the root tweet, types of users, etc. In addition to this, we also show how DRAGNET reacts to changes in the model parameters.

Comparative Analysis

Table 4.1 shows the overall performance. In general, DRAGNET outperforms all the baselines by a significant margin[4] across all three comparison metrics. TFT ends up as the best performing baseline. However, DRAGNET beats TFT by 0.094 points in r, 0.190 points in RMSE and 0.066 points in MFE. Among other baselines, LSTM and CNN turn out to be the worst baselines, while DeepAR and N-Beats have performances comparable to TFT. Surprisingly, ForGAN also seems to be one of the worst performers. This may be due to the extreme dissimilarity in the hate intensity profiles present in our dataset. We also show that DRAGNET abetted with the sentiment feature is more effective than the other baselines. However, with sentiment as an additional feature, the performance of DRAGNET improves further.

Detailed Introspection

We further dig deeper into the performance of DRAGNET to better understand its superiority and limitations. Throughout this study, we compare DRAGNET with TFT, the best baseline.

- **Hate speech classifier.** As explained in Section 3.1, we use a hate speech detection model to measure the hate intensity score. We are curious to know how the model performs if we change the hate speech classifier. To this end, we consider the following hate speech detection models – Davidson et al. [9] (default), Founta

[3]Given two sequences a and b of length n, $MFE(a,b) = \frac{\sum_{i=0}^{n}(a[i]-b[i])}{n}$.
[4]All results are significant with $p < 0.001$.

et al.[16] and Waseem et al.[50]. Figure 3.3(a) shows a consistent precedence of DRAGNET over TFT across all three hate detection models.

- **Weight w in the hate intensity score.** In Section 3.1, our hate intensity score uses w to facilitate the trade-off between hate detection model and the hate lexicon component. As shown in Figure 3.3(b), DRAGNET clearly outperforms TFT for all three values of w (i.e., 0.45, 0.6, 0.75).
- **Demography of root tweet.** Our curated dataset consists of reply threads with root tweets originating in various countries – the US, the UK and India & others (Brazil, Australia, Argentina). DRAGNET shows consistency across all these geographical locations as illustrated in Figure 3.3(c).
- **Topic-wise data.** Figure 3.3(d) illustrates the performance of both DRAGNET and TFT across four topics (detailed in Section 2.1). DRAGNET beats TFT across all topics.
- **Type of root tweet.** We randomly select a set of 1830 root tweets and manually label them as 'controversial', 'fake', 'hate' and 'others'. Figure 3.3(e) again illustrates that DRAGNET shows consistency across these labels.
- **Type of root users.** We also analyse whether the popularity of the user who posted the root tweet affects the performance of the models. To check this, we consider the follower count of a user as a proxy to popularity. We then divide the root tweets equally into various bins based on the follower count of root users and measure the performance across bins as shown in Figure 3.3(f) – Bin 1 (Bin 4) represents the set of the least (most) popular root users. DRAGNET once again remains consistent across bins, indicating its stable performance irrespective of the popularity of the root users.

3.3.3 Ablation Study

Furthermore, we study how various model parameters affect the performance of DRAGNET and TFT.

- **Change in δ.** The hyper-parameter δ represents the window size in the hate intensity profiles. Figure 3.4(a) shows that both DRAGNET and TFT improve in performance with an increase in the value of δ.
- **Change in t_h.** t_h represents the size of the initial history that it requires to predict the complete hate intensity profile for a new reply thread. As expected, Figure 3.4(b) illustrates that the prediction accuracy of DRAGNET and TFT increases as t_h increases.
- **Varying the number of clusters.** One of the crucial hyper-parameters of DRAGNET is the number of clusters, j in the fuzzy clustering step. Figure 3.4(c) illustrates that DRAGNET's performance is the best for $j = 15$ (which is the default value).
- **Varying the clustering algorithm.** We use Gaussian Mixture (GM) model as the default clustering method. We further check the performance of DRAGNET with other clustering methods – C-Means (a hard clustering method) and Bayesian Gaussian Mixture (BGM, a variant GM). Figure 3.4(d) confirms that GM performs the best out of the three, with BGM showing comparable results, This also supports our decision of choosing a fuzzy clustering approach.

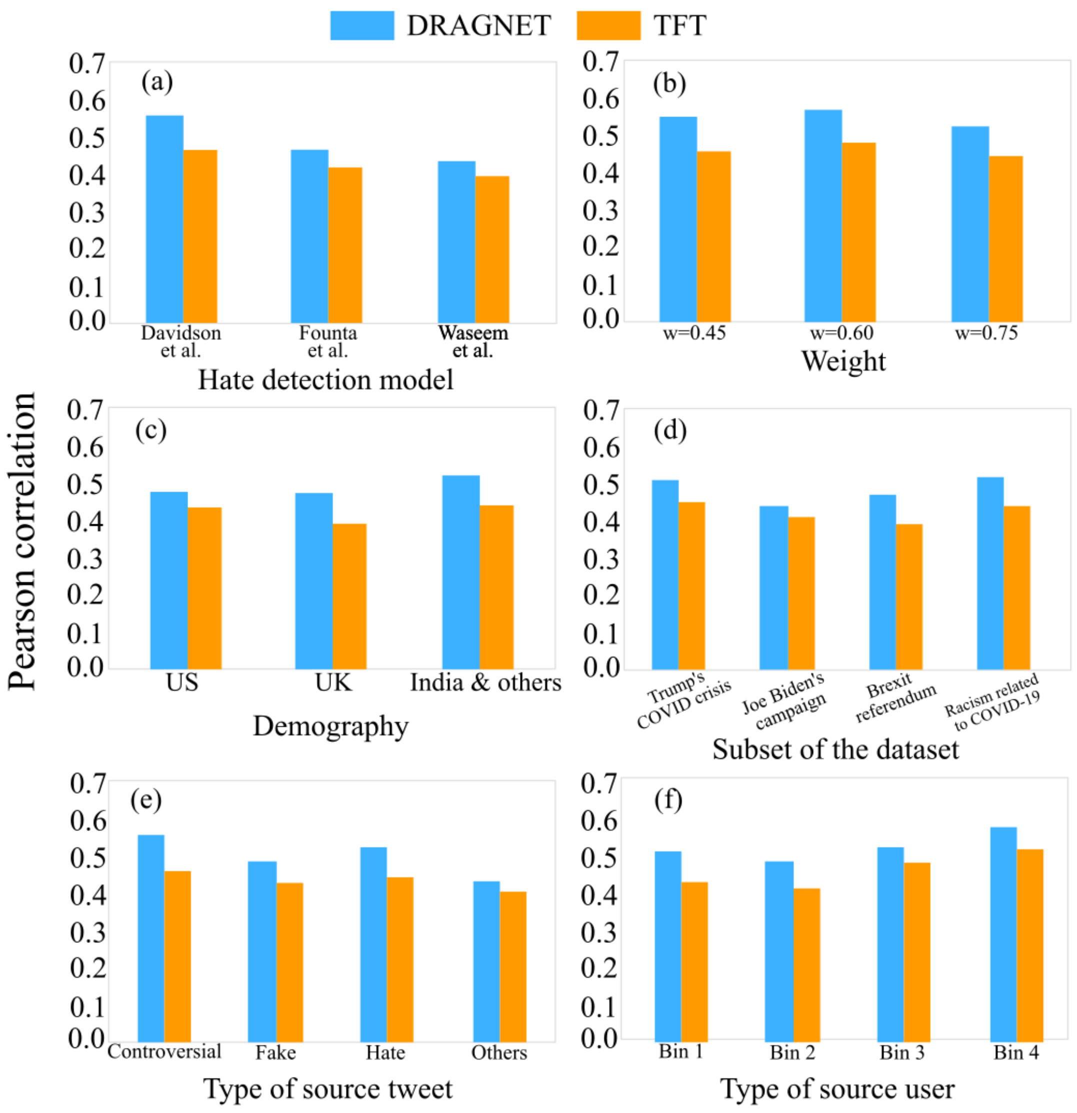


FIGURE 3.3: Comparison in performance of DRAGNET and TFT using the Pearson correlation w.r.t (a) hate speech detection model used, (b) weight w in the hate intensity function, (c) demography of the root tweet, (d) topics, (e) type of root tweet, and (f) type of source user.

3.4 Real-world Deployment

DRAGNET was deployed by a fact-checking startup in its advanced AI platform to monitor real-world harmful content in heterogeneous social media data. It was tested by being integrated into a proprietary pipeline that ranks and scores individual content pieces as well as aggregated semantic content clusters (narratives) for insights about misinformation and deleterious content. In particular, DRAGNET was used as an alternative way to profile dynamic streams of content and their aggregated equivalents. The aim here was to use DRAGNET to model the hate profile of narratives and also to link different narratives with similar profiles. The hate profile information of narratives is used along with other proprietary metrics to come up with insights about the potential risk and damage posed by a harmful narrative. The workflow of the application of DRAGNET in the AI Platform is illustrated in Figure 3.5.

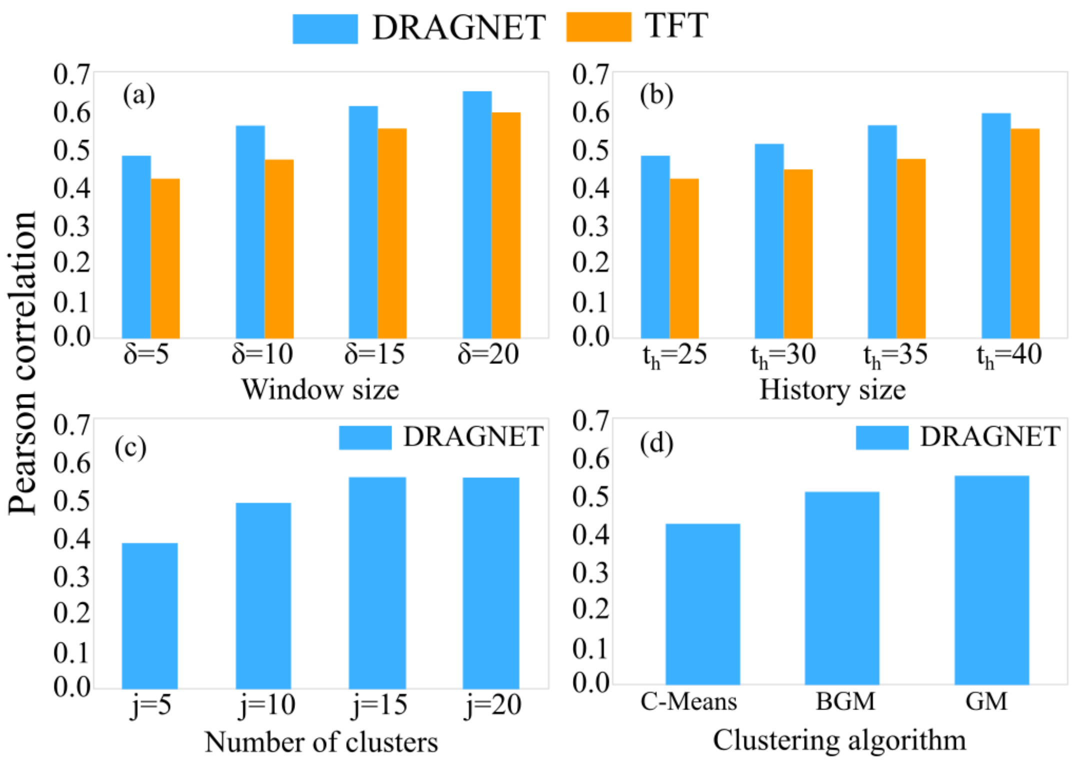


FIGURE 3.4: Performance of DRAGNET and TFT w.r.t the change in (a) window size, and (b) history. We also show the performance of DRAGNET with the change in (c) the number of clusters, and (d) the clustering method.

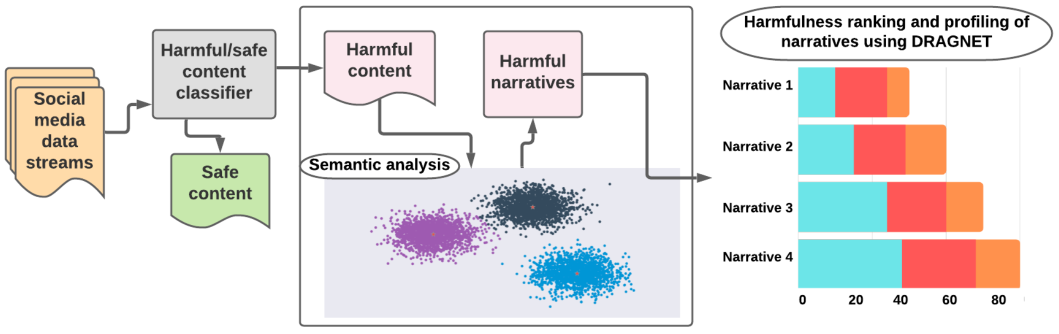

Social media data streams
Harmful/safe content classifier
Safe content
Harmful content
Harmful narratives
Semantic analysis
Harmfulness ranking and profiling of narratives using DRAGNET
Narrative 1
Narrative 2
Narrative 3
Narrative 4
0
20
40
60
80

FIGURE 3.5: Visual workflow of the application of DRAGNET in an advanced AI Platform for hate intensity profiling and ranking of narratives (aggregated semantic content analysis).

We observe that DRAGNET offers complementary insights and has significant potential to improve the overall accuracy of the pipelines in profiling and ranking content for hate speech and real-world online harmfulness. In particular, the hate intensity scores from DRAGNET offer additional knowledge to the proprietary pipelines for ranking and prioritization of high-risk online malevolence for enforcement of countermeasures to minimize their impact and damage.

We plan to further test DRAGNET extensively on multiple heterogeneous social media data streams ingested by the platform to comprehend DRAGNET's abilities to generalise accurately and to reliably detecting hate speech patterns.

Chapter 4

On the Fly Model Training

4.1 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, ModelOnely 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 [19]. 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 4.1

4.1.1 Preliminaries

In this chapter we follow the same definition of hate intensity as in Section 3.1.Since the underline model uses sliding window for training on the fly we define window as defined in Section 3.1 to be chunk, further define chunk size as Δ previously denoted as δ to avoid confusion. In the training set, from a given chunk $\mathcal{T}^{\varphi}_{k,k+\Delta}$, we measure the hate intensity score of its constituent replies, i.e., $\mathcal{H}(c^{\varphi}_{k})$, $\mathcal{H}(c^{\varphi}_{k+1})$, …, $\mathcal{H}(c^{\varphi}_{k+\Delta})$. From this, we extract five quantities, namely:
(i) sum of hate intensities, $\mathcal{H}(\mathcal{T}^{\varphi}_{k,k+\Delta})$,
(ii) hate intensity of the first reply for the chunk, $\mathcal{H}(c^{\varphi}_{k})$,
(iii) hate intensity of the last reply for the chunk, $\mathcal{H}(c^{\varphi}_{k+\Delta})$,
(iv) max hate intensity, $\mathcal{H}\left(\mathcal{T}^{\varphi}_{k,k+\Delta}\right)_{\max} = \max_{c\in\mathcal{T}^{\varphi}_{k,k+\Delta}} \mathcal{H}(c)$,
(v) min hate intensity, $\mathcal{H}\left(\mathcal{T}^{\varphi}_{k,i+\Delta}\right)_{\min} = \min_{c\in\mathcal{T}^{\varphi}_{k,k+\Delta}} \mathcal{H}(c)$.

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

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

4.1.2 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 , 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,

$$\begin{cases} \mathbf{z}_k = \mathbf{A}\mathbf{z}_{k-1} + \mathbf{v}_{1,k}, \\ \mathbf{x}_k = \mathbf{H}\mathbf{z}_k + \mathbf{v}_{2,k}, \end{cases} \tag{4.1}$$

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 [47]. 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 $k \in \{1, \ldots, K\}$,

$$\begin{cases} \mathbf{z}_k = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{z}_{k-1} + \mathbf{v}_{1,k}, \\ \mathbf{x}_k = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{z}_k + \mathbf{v}_{2,k}, \end{cases} \tag{4.2}$$

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 4.2 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 $\mathbf{A}_0 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{A}_1 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{A}_2 \in \mathbb{T}^{N_z\times N_z}$, $\mathbf{H}_0 \in \mathbb{R}^{N_x\times N_z}$, $\mathbf{H}_1 \in \mathbb{R}^{N_z\times N_z}$, $\mathbf{H}_2 \in \mathbb{R}^{N_z\times N_z}$ 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 [45]. 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.

4.1.3 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 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

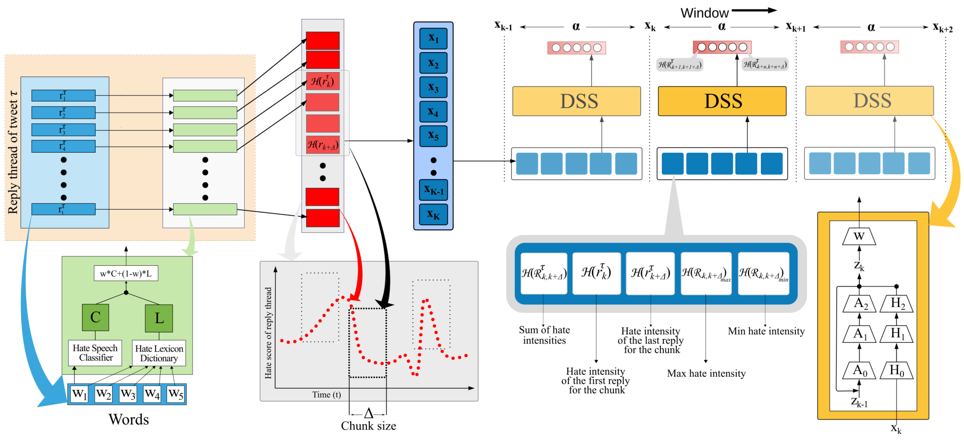


FIGURE 4.1: Schematic diagram of DESSERT where $\mathbf{x}_k$ represents the input (ordered sequence of vectors), $\mathbf{z}_{k-1}$ represents the hidden state-space vector (i.e. feature) for previous timestamp. $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ are the learnable parameters, and $\mathbf{z}_k$ represents the hidden state-space vector for current timestamp.

$\mathbb{R}^{N_z \times N_z}$ and $\bar{\mathbf{z}}_0 \in \mathbb{R}$. Then, Equation 4.2 reads as a first-order Markovian multi-linear Gaussian model, where $(\mathbf{z}_k)_{1\leq k\leq K}$ 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:

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}) = \mathcal{N}(\mathbf{z}_k; \bar{\mathbf{z}}_k, \mathbf{P}_k). \tag{4.3}$$

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 $k \in \{1, \ldots, K\}$, $\bar{\mathbf{z}}_k$ and $\mathbf{P}_k$ can be computed by means of the Kalman filter recursions as follows: For $k = 1, \ldots, K$
Predict state:

$$\begin{cases} \mathbf{z}_k^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\bar{\mathbf{z}}_{k-1}, \\ \mathbf{P}_k^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{P}_{k-1}(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top + \mathbf{Q}. \end{cases} \tag{4.4}$$

Update state:

$$\begin{cases} \mathbf{y}_k = \mathbf{x}_k - \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2 z_k^-,\ \mathbf{S}_k = \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{P}_k^-(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top + \mathbf{R}, \\ \mathbf{K}_k = \mathbf{P}_k^-(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top\mathbf{S}_k^{-1}, \\ \bar{\mathbf{z}}_k = \mathbf{z}_k^- + \mathbf{K}_k\mathbf{y}_k, \\ \mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \end{cases} \tag{4.5}$$

The Rauch-Tung-Striebel (RTS) smoother [45] makes a backward recursion on the data which makes use of the filtering distributions computed by the Kalman filter to obtain the smoothing distribution $p(\mathbf{z}_k|\mathbf{x}_{1\ldots K})$. Below, we summarize the RTS recursions:
For $k = K, \ldots, 1$
Backward Recursion (*Bayesian Smoothing*):

$$\begin{cases} \mathbf{z}_{k+1}^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\bar{\mathbf{z}}_k,\ \mathbf{P}_{k+1}^- = \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{P}_k(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top + \mathbf{Q}, \\ \mathbf{G}_k = \mathbf{P}_k(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top(\mathbf{P}_{k+1}^-)^{-1}, \\ \mathbf{z}_k^s = \mathbf{z}_k + \mathbf{G}_k[\mathbf{z}_{k+1}^s - \mathbf{z}_{k+1}^-], \\ \mathbf{P}_k^s = \mathbf{P}_k + \mathbf{G}_k(\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-)\mathbf{G}_k^\top. \end{cases} \tag{4.6}$$

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

4.1.4 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) [45]. 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 4.1.3, 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:

$$\begin{aligned}
\mathbf{\Sigma}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^s + \mathbf{z}_k^s(\mathbf{z}_k^s)^\top; \ \mathbf{\Phi}^{[i]} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_{k-1}^s + \mathbf{z}_{k-1}^s(\mathbf{z}_{k-1}^s)^\top, \\
\mathbf{B}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k(\mathbf{z}_k^s)^\top; \ \mathbf{D}^{[i]} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^s\mathbf{G}_{k-1}^\top + \mathbf{z}_k^s(\mathbf{z}_{k-1}^s)^\top, \\
\mathbf{\Gamma}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k\mathbf{x}_k^\top.
\end{aligned} \tag{4.7}$$

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 A.2 for detailed derivations):

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

And,

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

where $\mathrm{tr}(\cdot)$ 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:

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

And,

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

4.1.5 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.

4.1.6 Forecasting

As explained in Section 4.1.2, 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,

$$\widehat{\mathbf{x}}_{k+\alpha+1} = \mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_2 \mathbf{z}_{k+\alpha}^{-}, \tag{4.8}$$

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 $\mathcal{H}(\mathcal{T}_{k,k+\Delta}^{\varphi})$.

4.1.7 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:

$$p(\widehat{\mathbf{x}}_k | \mathbf{x}_{1:k-1}) = \mathcal{N}\left(\widehat{\mathbf{x}}_k; \mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_2 \mathbf{z}_k^{-}, \mathbf{S}_k\right), \tag{4.9}$$

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 4.1.3 (see [12] 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 $\mathbf{S}_k$, denoted by $\mathbf{S}_k[0,0]$, 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 $[\mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_2 \mathbf{z}_k^{-}][0] \pm \mathbf{S}_k[0,0]$. We can even go further in the analysis. For instance, we can define the confidence score about an increase of the hate intensity as:

$$\widehat{p}_k = \int_{\widehat{\mathbf{x}}_k[0]}^{+\infty} \mathcal{N}\left(y; \left[\mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_2 \mathbf{z}_k^{-}\right][0], \mathbf{S}_k[0,0]\right) dy \tag{4.10}$$

$$= 1 - \mathrm{CDF}(\widehat{\mathbf{x}}_k[0] | \left[\mathbf{H}_0 \mathbf{H}_1 \mathbf{H}_2 \mathbf{z}_k^{-}\right][0], \mathbf{S}_k[0,0]), \tag{4.11}$$

where CDF denotes the cumulative distribution function for the multivariate Gaussian model. Equation 4.10 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

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 4.1: Overall performance (↓: lower value is better).

validate the method by using the standard cross-entropy loss as:

$$\text{log-loss} = \frac{1}{K}\sum_{k=1}^{K} - \left(L_k[i]\log(\widehat{p}_k)\right), \tag{4.12}$$

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.

4.1.8 Time Complexity

$\triangleright$ **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 [33], which leads to a complexity of $\mathcal{O}(\alpha N_z^{2.376})$. $\triangleright$ **Testing.** The testing phase just amounts to evaluating the multi-linear equation (Equation 4.8). This has complexity of $\mathcal{O}(N_x N_z^2)$ for each chunk. It can be reduced to $\mathcal{O}(N_z^2)$ if we want to forecast only one feature (which is the case in our setting, i.e., only the overall hate intensity).

4.2 Experimental Results and Analysis

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).

4.2.1 Experiment Setup

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 $\mathcal{C}$ by [16]. 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.

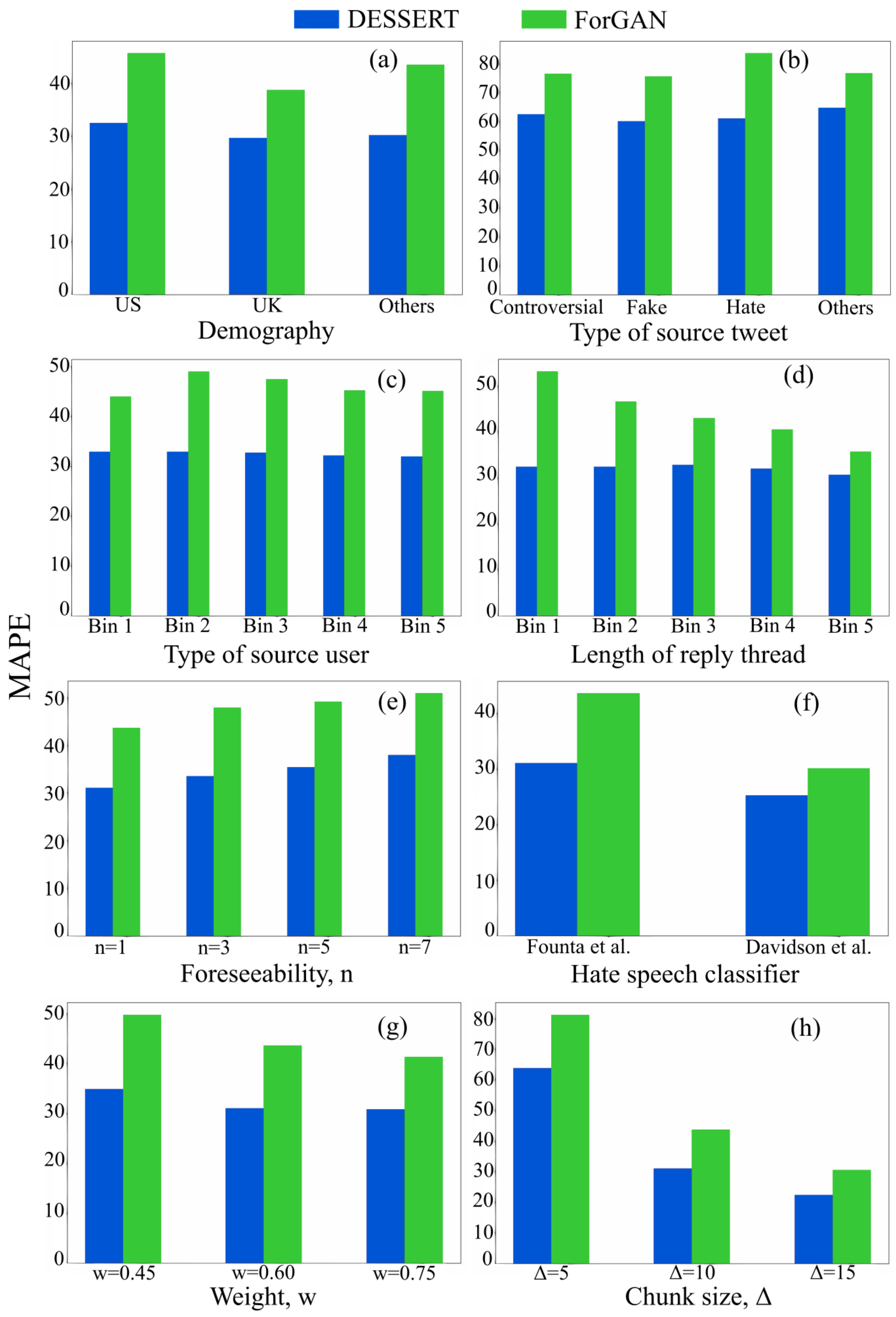


FIGURE 4.2: Performance (MAPE) of DESSERT and ForGAN w.r.t (a) demography, (b) types of source tweets, (c) types of source users, (d) length of reply threads, (e) foreseeability, (f) hate speech classifiers ([16] and [8]), (g) weight w, and (h) chunk size Δ. See the Appendix for the correlation-based (Pearson's r) performance.

4.2.2 Overall Performance

Table 4.1 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 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 4.1.7, 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 4.12). 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 4.2.3) 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.

4.2.3 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 4.2(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 4.2(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 4.2(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 4.2(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., $\mathcal{T}^{\varphi}_{t,t+\Delta}$. 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 $\mathcal{T}^{\varphi}_{t+(n-1)\Delta,t+n\Delta}$, with $n = 1, 3, 5, 7$. Also, it captures how early our model can predict. Figure 4.2(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 $\mathcal{C}$ in Equation 3.1. Here, we check how the models respond if we change $\mathcal{C}$. We

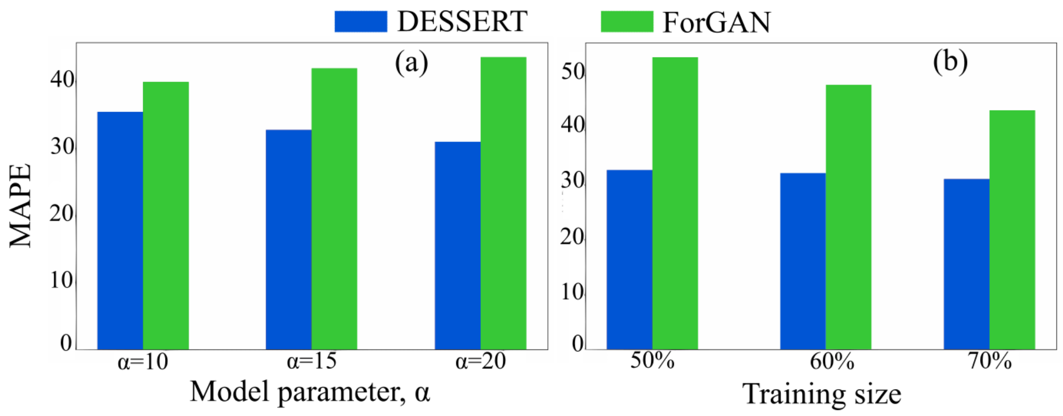


FIGURE 4.3: Model ablation w.r.t. (a) α and (b) training size.

replace our default $\mathcal{C}$ [17] by the hate speech detection method proposed by **?** and observe that DESSERT still outperforms ForGAN (Figure 4.2(f)).
Varying w. In Eq. 3.1, the weight w balances the effect of the hate speech classifier and the lexicon to measure hate intensity. Figure 4.2(g) shows that DESSERT's accuracy does not depend much on w.
Varying Δ. Figure 4.2(h) shows that with the increase of chunk size Δ, both ForGAN and DESSERT improve.

4.2.4 Ablation Study

Varying layers. Table 4.1 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 4.3(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. 4.3(b) shows that unlike ForGAN, DESSERT remains effective with varying training size.

4.3 Real-world Deployment

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 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 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 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 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.

Chapter 5

Conclusion

In this work, we studied a novel problem - hate intensity prediction of Twitter reply threads. We started off with curating a large-scale dataset of $\sim 4.5k$ complete reply threads related to four controversial topics from Twitter. We then proposed DRAGNET, a novel deep stratified learning model to address the problem. DRAGNET is highly efficient, outperforming six baselines.In addition, DRAGNET has been deployed in an advanced AI platform for monitoring detrimental content on the web to profile and ranking content clusters in order to track the high risk threats and thereby to recommend countermeasures to minimise their reach and to reduce the damage. We further proposed model that is an online model catering to the volatile nature of twitter i.e. 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. DESSERT has also 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.

Chapter 6

Publications

- Dahiya, Snehil, Shalini Sharma, Dhruv Sahnan, Vasu Goel, Emilie Chouzenoux, Víctor Elvira, Angshul Majumdar, Anil Bandhakavi, and Tanmoy 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, pp. 2732-2742. 2021. [7]
- Sahnan, Dhruv, Snehil Dahiya, Vasu Goel, Anil Bandhakavi, and Tanmoy 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), pp. 549-558. IEEE, 2021. [43]

Appendix A

Appendix

A.1 Additional Results

Figure A.2 shows the distribution of the reply thread length and the number of unique users per reply thread. Figure A.1 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 A.3 shows the detailed analysis of the results in terms of Pearson's r. Figure A.4 shows the ablation results in terms of Pearson's r.

A.2 Expectation-Minimization

The training of requires the estimation of the matrices $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ jointly with the state. This problem can be solved efficiently using an expectation-maximization (EM) algorithm [45], 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 model in Eq. 4.2, the EM consists in searching for $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2$ and $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$ maximizing $p(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2 | \mathbf{x}_{1:K})$ or, equivalently, maximizing $\varphi_K(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2)$ $= \log p(\mathbf{x}_{1:K} | \mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2)$. 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 $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2$, and $\mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2$, $\varphi_K(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2)$ $\geq \mathcal{Q}(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2; \mathbf{\Theta}^{[i]})$, where $\mathbf{\Theta}^{[i]}$ gathers the outputs of the RTS smoother at previous EM iterate i, listed in Equation 4.7. The application of [45, Theo.12.4] to our model in Equation 4.2 and the cancellation of constant terms, leads to the function

$$
\begin{aligned}
&\mathcal{Q}(\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_2, \mathbf{H}_0, \mathbf{H}_1, \mathbf{H}_2, \mathbf{\Theta}^{[i]}) = \\
&\qquad -\frac{K}{2}\mathrm{tr}\Big(\mathbf{Q}^{-1}\mathbf{\Sigma}^{[i]} - \mathbf{D}^{[i]}(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top - \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2(\mathbf{D}^{[i]})^\top \\
&\qquad\qquad + \mathbf{A}_0\mathbf{A}_1\mathbf{A}_2\mathbf{\Phi}^{[i]}(\mathbf{A}_0\mathbf{A}_1\mathbf{A}_2)^\top\Big) \\
&\qquad -\frac{K}{2}\mathrm{tr}\Big(\mathbf{R}^{-1}\mathbf{\Gamma}^{[i]} - \mathbf{B}^{[i]}(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top - \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2(\mathbf{B}^{[i]})^\top \\
&\qquad\qquad + \mathbf{H}_0\mathbf{H}_1\mathbf{H}_2\mathbf{\Sigma}^{[i]}(\mathbf{H}_0\mathbf{H}_1\mathbf{H}_2)^\top\Big).
\end{aligned}
$$

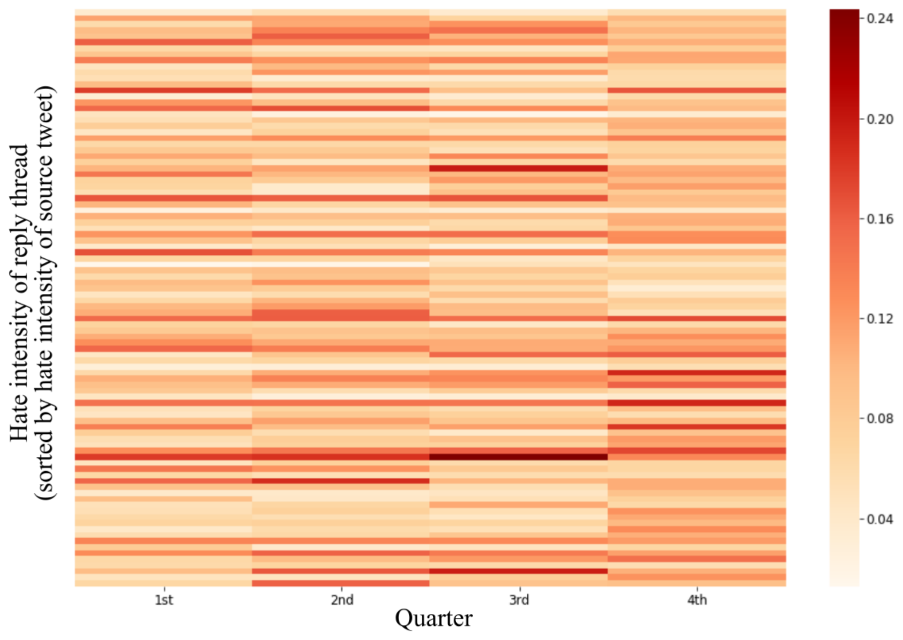


FIGURE A.1: Heat map showing the hate intensity per quarter of the reply threads (sorted by the hate intensity of the source tweets). We divide each reply thread into four equal quarters and measure hate intensity per quarter. It indicates that there is no single quarter which is always more hateful than the others across reply threads.

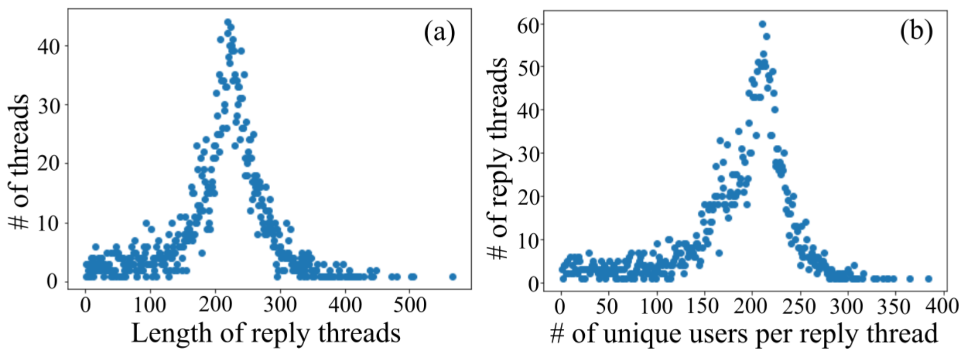


FIGURE A.2: Distribution of (a) the length of reply threads and (b) the number of unique users involved in reply threads.

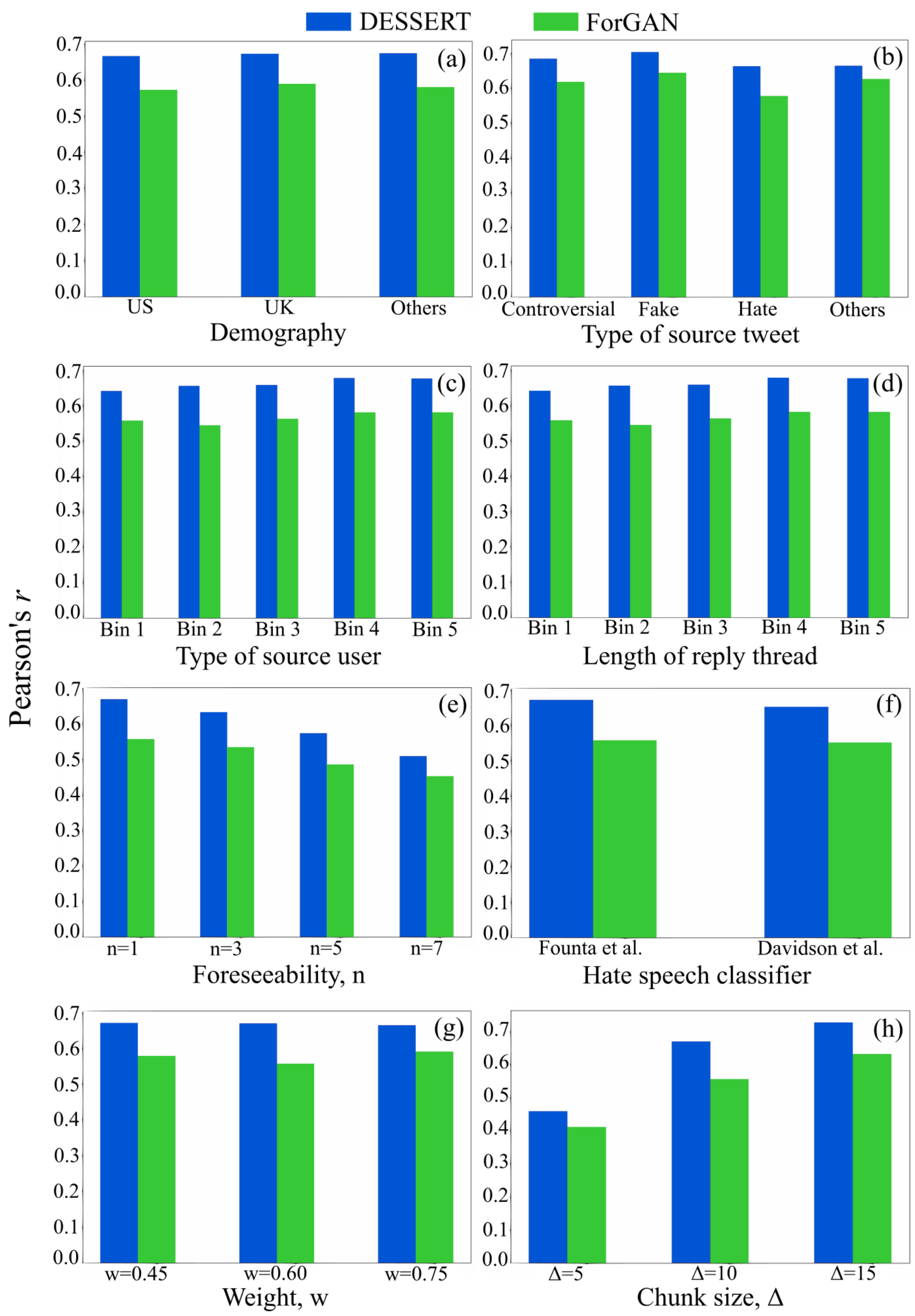


FIGURE A.3: Performance (Pearson's r) of and ForGAN w.r.t (a) demography, (b) types of source tweets, (c) types of source users, (d) length of reply threads, (e) foreseeability, (f) hate speech classifiers (**?** and **?**), (g) weight w, and (h) chunk size Δ.

The proposed update in Section 4.1.4 amounts to maximizing $\mathcal{Q}(\cdot, \boldsymbol{\Theta}^{[i]})$, for $i = 1, 2, \ldots$, using a coordinate descent algorithm, i.e.,

$$\begin{aligned}
\mathbf{A}_0^{[i+1]} &= \text{argmin}_{\mathbf{A}_0} - \mathcal{Q}(\mathbf{A}_0, \mathbf{A}_1^{[i]}, \mathbf{A}_2^{[i]}, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}, \boldsymbol{\Theta}^{[i]}) \\
\mathbf{A}_1^{[i+1]} &= \text{argmin}_{\mathbf{A}_1} - \mathcal{Q}(\mathbf{A}_0^{[i+1]}, \mathbf{A}_1, \mathbf{A}_2^{[i]}, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}, \boldsymbol{\Theta}^{[i]}) \\
\mathbf{A}_2^{[i+1]} &= \text{argmin}_{\mathbf{A}_2} - \mathcal{Q}(\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2, \mathbf{H}_0^{[i]}, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}, \boldsymbol{\Theta}^{[i]}) \\
\mathbf{H}_0^{[i+1]} &= \text{argmin}_{\mathbf{H}_0} - \mathcal{Q}(\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2^{[i+1]}, \mathbf{H}_0, \mathbf{H}_1^{[i]}, \mathbf{H}_2^{[i]}, \boldsymbol{\Theta}^{[i]}) \\
\mathbf{H}_1^{[i+1]} &= \text{argmin}_{\mathbf{H}_1} - \mathcal{Q}(\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2^{[i+1]}, \mathbf{H}_0^{[i+1]}, \mathbf{H}_1, \mathbf{H}_2^{[i]}, \boldsymbol{\Theta}^{[i]}) \\
\mathbf{H}_2^{[i+1]} &= \text{argmin}_{\mathbf{H}_2} - \mathcal{Q}(\mathbf{A}_0^{[i+1]}, \mathbf{A}_1^{[i+1]}, \mathbf{A}_2^{[i+1]}, \mathbf{H}_0^{[i+1]}, \mathbf{H}_1^{[i+1]}, \mathbf{H}_2, \boldsymbol{\Theta}^{[i]}),
\end{aligned}$$

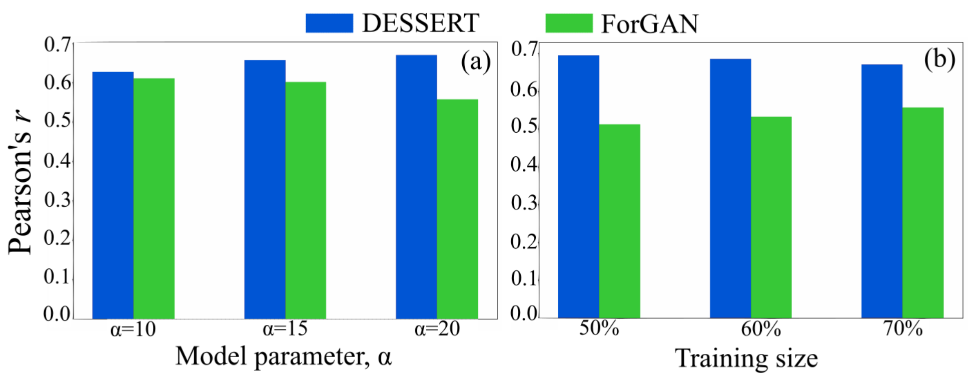


FIGURE A.4: Model ablation w.r.t. (a) α and (b) training size.

which leads to the six sub-problems provided in Section 4.1.4. This procedure is theoretically sound (see [4, 45] 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.

Bibliography

[1] Pinkesh Badjatiya et al. "Deep Learning for Hate Speech Detection in Tweets". In: *WWW*. 2017, 759–760.

[2] Mohit Bhardwaj et al. "Hostility detection dataset in Hindi". In: *arXiv preprint arXiv:2011.03588* (2020).

[3] Tanmoy Chakraborty et al. *Combating Online Hostile Posts in Regional Languages During Emergency Situation: First International Workshop, CONSTRAINT 2021, Collocated with AAAI 2021, Virtual Event, February 8, 2021, Revised Selected Papers*. Vol. 1402. Springer Nature, 2021.

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

[5] Emilie Chouzenoux and Jean-Christophe Pesquet. "A Stochastic Majorize-Minimize Subspace Algorithm for Online Penalized Least Squares Estimation". In: *IEEE Transactions on Signal Processing* 65.18 (2017), pp. 4770–4783.

[6] Zhicheng Cui, Wenlin Chen, and Yixin Chen. "Multi-scale convolutional neural networks for time series classification". In: *arXiv preprint arXiv:1603.06995* (2016).

[7] Snehil Dahiya et al. "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*. 2021, pp. 2732–2742.

[8] Thomas Davidson et al. "Automated Hate Speech Detection and the Problem of Offensive Language". In: *ICWSM*. 2017, pp. 512–515.

[9] Thomas Davidson et al. "Automated hate speech detection and the problem of offensive language". In: *ICWSM*. Vol. 11. 1. 2017.

[10] P Deepak, Tanmoy Chakraborty, Cheng Long, et al. *Data Science for Fake News: Surveys and Perspectives*. Vol. 42. Springer Nature, 2021.

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

[12] V. Elvira, J. Míguez, and P. .M. Djurić. "Adapting the Number of Particles in Sequential Monte Carlo Methods through an Online Scheme for Convergence Assessment". In: *IEEE Transactions on Signal Processing* 65.7 (2017), pp. 1781–1794.

[13] Chenyou Fan et al. "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.

[14] Hassan Ismail Fawaz et al. "Inceptiontime: Finding alexnet for time series classification". In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[15] Paula Fortuna and Sérgio Nunes. “A survey on automatic detection of hate speech in text”. In: *ACM Computing Surveys* 51.4 (2018), pp. 1–30.

[16] Antigoni Maria Founta et al. “A Unified Deep Learning Architecture for Abuse Detection”. In: *WebSci*. 2019, 105–114.

[17] Antigoni-Maria Founta et al. “Large Scale Crowdsourcing and Characterization of Twitter Abusive Behavior”. In: *ICWSM*. 2018, pp. 491–500.

[18] Arijit Ghosh Chowdhury et al. “ARHNet - Leveraging Community Interaction for Detection of Religious Hate Speech in Arabic”. In: *ACL Student Research Workshop*. Florence, Italy, July 2019, pp. 273–280.

[19] Barbara Hammer. “On the approximation capability of recurrent neural networks”. In: *Neurocomputing* 31.1-4 (2000), pp. 107–123.

[20] Peter Hastings et al. “Stratified learning for reducing training set size”. In: *International Conference on Intelligent Tutoring Systems*. Springer. 2016, pp. 341–346.

[21] Md. Rezaul Karim, Sumon Dey, and Bharathi Raja Chakravarthi. “DeepHateExplainer: Explainable Hate Speech Detection in Under-resourced Bengali Language”. In: *ArXiv* abs/2012.14353 (2020).

[22] Douwe Kiela et al. “The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes”. In: *NIPS*. 2020, pp. 1–14.

[23] Alireza Koochali et al. “Probabilistic forecasting of sensory data with generative adversarial networks–forgan”. In: *IEEE Access* 7 (2019), pp. 63868–63880.

[24] Shiyang Li et al. “Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting”. In: *arXiv preprint arXiv:1907.00235* (2019).

[25] Bryan Lim et al. “Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting”. In: *CoRR* abs/1912.09363 (2019). arXiv: `1912.09363`.

[26] Ping Liu et al. “Forecasting the presence and intensity of hostility on Instagram using linguistic and social features”. In: *ICWSM*. Vol. 12. 1. 2018.

[27] Son T Luu, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. “A Large-scale Dataset for Hate Speech Detection on Vietnamese Social Media Texts”. In: *arXiv preprint arXiv:2103.11528* (2021).

[28] Srishti Rai2 Manoj Kumar Pathak1. “Mob Lynching: A New Form of Hate Crime”. In: *Medico Legal Update* 20.3 (2020), pp. 122–128.

[29] Zelda Mariet and Vitaly Kuznetsov. “Foundations of sequence-to-sequence modeling for time series”. In: *AISTATS*. 2019, pp. 408–417.

[30] Sarah Masud et al. “Hate is the New Infodemic: A Topic-aware Modeling of Hate Speech Diffusion on Twitter”. In: *arXiv preprint arXiv:2010.04377* (2020).

[31] Binny Mathew et al. “Spread of Hate Speech in Online Social Media”. In: *WebSci*. 2019, 173–182.

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

[33] Corey Montella. *The Kalman filter and related algorithms: A literature review*. Tech. rep. `https://www.researchgate.net/publication/236897001_The_Kalman_Filter_and_Related_Algorithms_A_Literature_Review`. 2011.

[34] Vit Niennattrakul and Chotirat Ann Ratanamahatana. “On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping”. In: *MUE*. 2007, pp. 733–738. DOI: 10.1109/MUE.2007.165.

[35] Boris N Oreshkin et al. “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting”. In: *arXiv preprint arXiv:1905.10437* (2019).

[36] Parth Patwa et al. “Fighting an infodemic: Covid-19 fake news dataset”. In: *International Workshop on Combating Online Hostile Posts in Regional Languages during Emergency Situation*. Springer. 2021, pp. 21–29.

[37] Parth Patwa et al. “Overview of CONSTRAINT 2021 Shared Tasks: Detecting English COVID-19 Fake News and Hindi Hostile Posts”. In: *Combating Online Hostile Posts in Regional Languages during Emergency Situation*. Ed. by Tanmoy Chakraborty et al. Cham: Springer International Publishing, 2021, pp. 42–53. ISBN: 978-3-030-73696-5.

[38] Fabio Poletto et al. “Resources and benchmark corpora for hate speech detection: a systematic review”. In: *Lang Resources & Evaluation (2020)* (2020).

[39] Jing Qian et al. “Lifelong Learning of Hate Speech Classification on Social Media”. In: *NAACL*. 2021, pp. 2304–2314.

[40] Douglas Reynolds. “Gaussian Mixture Models”. In: *Encyclopedia of Biometrics*. Ed. by Stan Z. Li and Anil Jain. Boston, MA: Springer US, 2009, pp. 659–663. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_196. URL: https://doi.org/10.1007/978-0-387-73003-5_196.

[41] Manoel Ribeiro et al. “"Like Sheep Among Wolves": Characterizing Hateful Users on Twitter”. In: *MIS2 Workshop at WSDM’2018*. Dec. 2017.

[42] Ignacio Rojas et al. “Soft-computing techniques and ARMA model for time series prediction”. In: *Neurocomputing* 71.4-6 (2008), pp. 519–537.

[43] Dhruv Sahnan et al. “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.

[44] David Salinas et al. “DeepAR: Probabilistic forecasting with autoregressive recurrent networks”. In: *International Journal of Forecasting* 36.3 (2020), pp. 1181–1191.

[45] Simo Särkkä. *Bayesian filtering and smoothing*. 3. Cambridge University Press, 2013.

[46] Rajat Sen, Hsiang-Fu Yu, and Inderjit Dhillon. “Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting”. In: *arXiv preprint arXiv:1905.03806* (2019).

[47] Shalini Sharma et al. “Blind Kalman Filtering for Short-term Load Forecasting”. In: *IEEE Transactions on Power Systems* 35.6 (2020), pp. 4916–4919.

[48] Jianhua Sun et al. “Three Steps to Multimodal Trajectory Prediction: Modality Clustering, Classification and Synthesis”. In: *arXiv preprint arXiv:2103.07854* (2021).

[49] Milo Trujillo et al. “What is BitChute? Characterizing the "Free Speech" Alternative to YouTube”. In: *ACM Hypertext*. New York, NY, USA: Association for Computing Machinery, 2020, 139–140. ISBN: 9781450370981.

[50] Zeerak Waseem and Dirk Hovy. "Hateful symbols or hateful people? predictive features for hate speech detection on twitter". In: *NAACL student research workshop*. 2016, pp. 88–93.

[51] Xin Wei et al. "Machine learning for pore-water pressure time-series prediction: application of recurrent neural networks". In: *Geoscience Frontiers* 12.1 (2021), pp. 453–467.

[52] Michael Wiegand et al. "Inducing a Lexicon of Abusive Words – a Feature-Based Approach". In: *NAACL*. 2018, pp. 1046–1056.

[53] Cuili Yang et al. "Design of polynomial echo state networks for time series prediction". In: *Neurocomputing* 290 (2018), pp. 148–160.

[54] Zhilin Yang et al. "Xlnet: Generalized autoregressive pretraining for language understanding". In: *arXiv preprint arXiv:1906.08237* (2019).

[55] Ye Yuan and Kris Kitani. "Diverse trajectory forecasting with determinantal point processes". In: *arXiv preprint arXiv:1907.04967* (2019).

[56] Savvas Zannettou et al. "What is Gab: A Bastion of Free Speech or an Alt-Right Echo Chamber". In: *WWW*. 2018, 1007–1014.

[57] Ali Ziat et al. "Spatio-temporal neural networks for space-time series forecasting and relations discovery". In: *ICDE*. IEEE. 2017, pp. 705–714.