DDoS Event Forecasting using Twitter Data*

Zhongqing Wang†,‡ and **Yue Zhang**‡
†Soochow University, Suzhou, China
‡Singapore University of Technology and Design, Singapore
wangzq.antony@gmail.com, yue_zhang@sutd.edu.sg

Abstract

Distributed Denial of Service (DDoS) attacks have been significant threats to the Internet. Traditional research in cyber security focuses on detecting emerging DDoS attacks by tracing network package flow. A characteristic of DDoS defense is that rescue time is limited since the launch of attack. More resilient detection and defence models are typically more costly. We aim at predicting the likelihood of DDoS attacks by monitoring relevant text streams in social media, so that the level of defense can be adjusted dynamically for maximizing cost-effect. To our knowledge, this is a novel yet challenging research question for DDoS rescue. Because the input of this task is a text stream rather than a document, information should be collected both on the textual content of individual posts. We propose a fine-grained hierarchical stream model to capture semantic information over infinitely long history, and reveal burstiness and trends. Empirical evaluation shows that social text streams are indeed informative for DDoS forecasting, and our proposed hierarchical model is more effective compared to strong baseline text stream models and discrete bag-of-words models.

1 Introduction

A Distributed Denial of Service (DDoS) attack employs multiple compromised systems to interrupt or suspend services of a host connected to the Internet [Carl *et al.*, 2006]. Victims are often high-profile web servers such as banks or credit card payment gateways, and therefore a single attack may cause considerable loss [Matthews, 2014]. DDoS attacks are difficult to detect and predict [Bleakley and Vert, 2011]. Traditionally, the aim of a DDoS detection system is to detect and distinguish malicious packet traffic from legitimate traffic [Mirkovic and Reiher, 2004]. Because malicious traffic occurs only after a DDoS attack has begun, there is limited time to prevent damage. In practice, more dynamic defense systems can be deployed for increased resilience to DDoS, with increased cost.

Post	**Target**
The Thai government is about to end their citizens Internet freedom.	Thai Government
soooooooooooo basically they are all leaving soon, sony is mad as hell.	SONY

Table 1: Example tweets with attack targets.

Ideally, if the likelihood of DDoS attacks can be forecasted, it can be used to guide configuration of a DDoS detection and defense system over a certain period of time — when a DDoS is more likely to happen, a more dynamic and costly configuration can be used; but when DDoS is unlikely, the defense level can be reduced to maximize cost effectiveness. This paper investigates the feasibility of forecasting the likelihood of DDoS attacks *before* they happen by monitoring social media stream. Our motivation is that the attacked targets may be mentioned unfavorably or arouse negative sentiments in social media text. Some examples are shown in Table 1. The first tweet describes a future policy of the Thai government, which led to a DDoS attack on the government website. The second is a negative post towards SONY, and bursty negative sentiment in similar posts can suggest possible DDoS attacks. Tweets may not contain hints to all network attacks but we hypothesize that textual information on social media can help network engineers to take necessary actions ahead of time.

Based on the above assumptions, the setting of our research is to monitor tweet streams for a certain target of concern, such as the Thai government and SONY. Our task is to predict whether a DDoS event is likely occur in the next day, given the tweet stream over a historical period related to the monitored target. We choose daily forecast as the time span. Without loss of generality, our model can be used for different time span granularity. This is an ambitious yet challenging task. Since the input is a text stream rather than a document, an ideal model should capture both tweet-level information and stream-level information such as burstiness and sentiment over history.

We propose several deep learning models that induce stream features automatically. Words are represented using distributed vector embeddings injecting sentiment information, and recurrent neural networks are used to cast tweet streams of varying lengths into fixed-size vectors. We compare flat recurrent stream models with a *hierarchial stream*

*This work has been done when the first author worked at SUTD.

model, which arranges infinity long tweets history into a day-week-month hierarchical structure using neural networks to automatically induce streams semantic information. Results show that text streams are informative for predicting DDoS attacks. In addition, the hierarchical stream model that considers long and short term history is more effective compared to fully sentiment-based and flat stream models. Our code and dataset are both available at https://github.com/wangzq870305/ddos_forecast.

2 Related Work

DDoS attacks have become significant threats to the Internet. Effort has been made from both the academia and the industry on their detection and defense [Carl *et al.*, 2006; Zargar *et al.*, 2013]. Typical strategies monitor header information of network packets, deriving an activity profile (which can be the average packet rate of a network flow of consecutive packets with similar fields, such as address, port, and protocol) [Mirkovic and Reiher, 2004]. DDoS can be detected by measuring network activity with all inbound and outbound flows. For example, change-point detection algorithms isolate a change in traffic statistics caused by attacks. These approaches initially filter target traffic data by the address, port, or protocol, storing the resultant flow as a time series, which can show a statistical change when a DDoS flooding attack begins [Bleakley and Vert, 2011].

The key idea of most DDoS detection methods in cyber security is to focus on malicious packet traffic. However, network traffic will change only when a DDoS attack has begun. Hence there is limited time to prevent damages. Our approach predicts the likelihood DDoS attacks from social media text before they begin, and hence can be used to complement and guide network traffic monitor for DDoS defense.

In prior NLP research, the most relevant work is DDoS event mention extraction, which extracts DDoS event mentions from social media. In particular, Ritter *et al.* [2015] exploited expectation regularization to extract DDoS event mentions from large amounts of raw tweets that contain the keyword 'ddos'. Chang *et al.* [2016] proposed a LSTM-based neural model that learns tweet-level features automatically, which improve the accuracies over the method of Ritter *et al.* [2015]. Different from their work, which only considers extracting DDoS related posts, we take a step further, aiming to *predict* the likelihood of DDoS attacks according to tweets. Meanwhile, Ding *et al.* [2015] proposed a deep learning method for event-driven stock market prediction, where two deep convolutional neural networks are used to model short-term and long-term influences of events on stock price movements, respectively. Our work is similar to the above work, yet we use recurrent networks rather than CNNs to model streams of texts, showing that they give better accuracies. To our knowledge, we are the first to use a hierarchical RNN model to render a text stream, and are the first to use text mining for event prediction in the cyber security domain.

3 Basic Networks

We take Convolutional Neural Network (CNN) [Collobert *et al.*, 2011] and Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997] as two basic components in building our neural models. Below we specify the exact *CNN* and *LSTM* variations that are used in this paper.

3.1 Convolutional Neural Network (CNN)

Given a set of input vectors $\{x_1, x_2, ..., x_n\}$, denote:

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n \quad (1)$$

CNN [Collobert *et al.*, 2011] uses convolution functions to compress $x_{1:n}$ into a feature vector. A convolution feature z_i is generated from a window of input $x_{i:i+h-1}$ by:

$$z_i = \tanh(W \cdot x_{i:i+h-1} + b) \quad (2)$$

where the parameters W is a matrix, $b \in R$ is a base term, and $\tanh$ is a non-linear function.

The same convolution formula is applied to $X_{1:n}$ and a max pooling operation is performed on to $Z = [z_1, z_2, ..., z_k]$ to obtain a vector h, where

$$h_j = \max_i Z_{j,i}, 0 < j \leq m \quad (h \in \mathbb{R}^m) \quad (3)$$

For the rest of the paper, we use $\mathrm{CNN}(x)$ to denote the CNN operation above on input $X_{1:n}$ to obtain h.

3.2 Long Short-Term Memory (LSTM)

LSTM [Hochreiter and Schmidhuber, 1997] models a recurrent state transformation sequence from an input sequence $\{x_1, x_2, ..., x_t\}$ to a hidden state sequence $\{h_1, h_2, ..., h_t\}$. At each time step, an input gate, a memory gate and a output gate, denoted as i_t, f_t and o_t respectively, are used to obtain h_t as follows:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{i-1} + b^{(i)}) \quad (4)$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \quad (5)$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \quad (6)$$

$$u_t = \tanh(W^{(u)} + U^{(u)}h_{t-1} + b^{(u)}) \quad (7)$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \quad (8)$$

$$h_t = o_t \odot \tanh(c_t) \quad (9)$$

The above is the standard LSTM variation without coupled input and forget gate or peephole connections. σ denotes the sigmoid function. For the rest of the paper, we use $\mathrm{LSTM}(x_t, h_{t-1})$ to denote the LSTM operation to obtain h_t.

4 Task Definition

The main research question that we investigate is whether tweet streams contain useful information for DDoS defense. Our task is to predict the likelihood that a DDoS event will occur to a *certain target* in day d, given the tweet stream over a history period X related to the monitored target. X is a sequence of N^P days ($X = \{d^{N^P}, ..., d^2, d^1\}$) immediately before d, where d^1 is the day before d and $d^i > d^{i+1}$. N^P can be arbitrary large. We choose to use a *day model* as the basic model instead of a hour model or minute model, because tweets from a certain target on each hour and minute are relatively few, and our DDoS defense system is best reconfigured daily. The set of tweets posted on d^i is denoted as $d^i = \{t^1, t^2, ..., t^{N_i^d}\}$, where N_i^d denotes the number of tweets of the day d^i. Each tweet consists of a sequence of words $t^j = \{w^1, w^2, ..., w^{N_j^t}\}$, where N_j^t denotes the length of the tweet t^j.

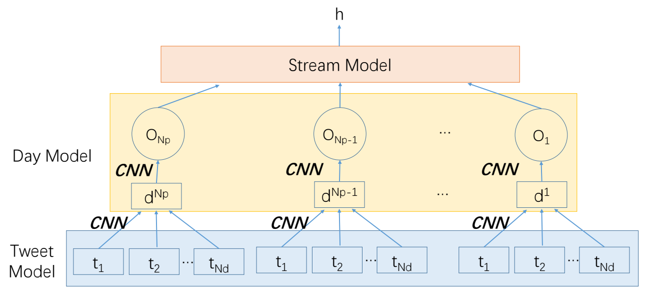

Figure 1: Overview of the proposed framework.

4.1 Distributed Word Embeddings

In the input level, we represent each word w^k with a K-dimensional embedding [Mikolov *et al.*, 2013], thus mapping a tweet t^j into a matrix

$$t^j = \{e(w^1), e(w^2), ..., e(w^{N_j^t})\} \tag{10}$$

Word embeddings capture distributed semantic information, and are thus useful for representing tweet content and bursty events. On the other hand, we find that modelling sentiment information explicitly in the input level can be useful for DDoS forecasting also. We follow Vo and Zhang [2015], using the sentiment-enriched embedding [Tang *et al.*, 2014] of words in sentiment lexicons as a second representation of t^j

$$t^j = \{e_s(w^{s1}), e_s(w^{s2}), ..., e_s(w^{sN_j^t})\} \tag{11}$$

In order to leverage both common word embeddings and explicit sentiment information, we combine the two types of embeddings by simply concatenation the two tweet representations above

$$\begin{aligned} t^j = \{&e(w^1), e(w^2), ..., e(w^{N_j^t}), \\ &e_s(w^{s1}), e_s(w^{s2}), ..., e_s(w^{sN_j^t})\} \end{aligned} \tag{12}$$

4.2 Neural Stream Models

For forecasting DDoS attacks, we use a tweet model to represent text-level features based on the tweet content, a day model to capture daily tweet representations, and a stream model to capture information over the daily stream history, such as changes in bursty topics and sentiments. Hierarchical stream representation is fed to a prediction model for DDoS forecast. The overall architecture of our model is shown on Figure 1.

Tweet Sub Model. We use a CNN to construct the tweet model, representing text-level features for individual tweets. CNN has been shown to give competitive results on sentence modeling [Kim, 2014; Johnson and Zhang, 2015]. The input is the sequence of words of tweet t_i, and the output is a vector representation of the tweet o_j^{tweet}

$$\begin{aligned} o_j^{tweet} &= \text{CNN}(t^j) = \\ \text{CNN}(\langle &e(w^1), e(w^2), ..., e(w^{N_j^t}), \\ &e_s(w^{s1}), e_s(w^{s2}), ..., e_s(w^{sN_j^t})\rangle) \end{aligned} \tag{13}$$

Day-level Sub Model. We treat all relevant tweets in a day as a unit, and use a CNN to extract a unified daily representation vector. In particular, $o_j^{day} = \text{CNN}(d^j) = \text{CNN}(\langle o_1^{tweet}, o_2^{tweet}, ..., o_{N_D}^{tweet}\rangle)$ is used for representing the features of all the tweets in the day d_j.

Stream Sub Model. As shown in Figure 1, daily tweets form a consecutive sequence of vectors $O_{N_p}^{day}, ..., O_2^{day}, O_1^{day}$. We use stream models to capture text stream information on top of the day model. A simple stream model can be a one-layer LSTM on the daily tweet sequence directly. On the other hand, more sophisticated models can be exploited by capturing richer features over a text stream. Without losing generality, we use $h = \text{stream}(o^{day}, O_1^{day})$ to denote the stream model output, which is a fix-sized vector h given $\{O_{N_p}^{day}, ..., O_2^{day}, O_1^{day}\}$ of the day model. We use LSTM as a basic component for streaming modeling. Different from discrete models, a recurrent neural stream models can effectively capture salient features from a daily tweet stream, yet retraining an infinitely long history without losing short or long term data. In this study, we propose and compare three neural network structures to this end. The discussion of stream models will be described in the next section.

Prediction Sub Model. We use a softmax classifier to predict the attack label y based on h, where label probabilities are calculated as:

$$\widehat{p}_\theta(y|X) = \text{softmax}(W^{(s)}h + b^{(s)}) \tag{14}$$

$$\widehat{y} = \arg\max_y \widehat{p}_\theta(y|X) \tag{15}$$

For training, the loss function is the negative log-likelihood of the true class labels $y^{(k)}$ at each node ($k \in \{0, 1\}$):

$$J(\theta) = -\frac{1}{2}\sum_{k=0}^{1} \log \widehat{p}_\theta(y^{(k)}|X^{(k)}) + \frac{\lambda}{2}||\theta||_2^2 \tag{16}$$

where the superscript k indicates the k^{th} labeled node, and λ is an L2 regularization hyperparameter.

We apply online training, optimizing parameters by using Adagrad [Duchi *et al.*, 2011]. In order to avoid over-fitting, dropout [Hinton *et al.*, 2012] is applied to word embeddings with a ratio of 0.2. For LSTM stream models, we empirically set the size of hidden layers to 32. For CNNs, the dimension of the input layer is set to 128, and the output dimension of convolution layers is 32. We train word embeddings using the *Skip-gram* algorithm[1] [Mikolov *et al.*, 2013]. All of the hyperparameters are turned in the development dataset.

5 Stream Model

Now we return to details of stream models. As mentioned in the previous section, given a sequence of daily history tweet vectors $\{o_{N_p}^{day}, ..., o_2^{day}, o_1^{day}\}$, the stream model calculates a fixed sized vector h that encodes salient features from the stream, such as bursty trends.

5.1 Vanilla Stream Model

As a baseline, we model a tweet stream by using an L-STM to recurrently capture daily tweet history. Formally, given $\{o_{N_p}^{day}, ..., o_2^{day}, o_1^{day}\}$ from the day model, we

[1]https://code.google.com/p/word2vec/

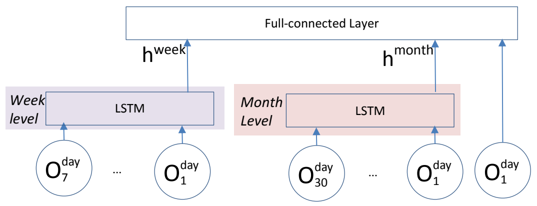

Figure 2: Short- and long-term model.

obtain a corresponding sequence of hidden state vectors $\{h_{N_P}^{day}, ..., h_2^{day}, h_1^{day}\}$, where,

$$h_t^{day} = \text{LSTM}^{DS}(o_t^{day}, h_{t+1}^{day}) \tag{17}$$

From this sequence, the last hidden state vector h_1^{day} is used as the stream representation vector h, and fed to the prediction model for forecasting the likelihood of DDoS attack on day d.

5.2 Short- and Long-Term Stream Model

The vanilla stream model above can capture unbounded history features from a daily tweet stream. However, it does not explicitly model the difference between short and long term histories, which can be useful for two major reasons. First, a contrast between short and long term history can reveal burstiness and trends. Second, the relative importance of longer term history should be smaller compared to that of shorter term history. To verify the above hypothesis, we develop a stream model that captures short-term and long-term histories separately with different LSTMs.

The structure of the proposed model is shown in Figure 2. In particular, a *weekly LSTM model* is used to capture short-term history $\{d_7, d_6, ..., d_1\}$, and a *monthly LSTM model* is used to capture long-term history $\{d_{30}, d_{29}, ..., d_1\}$. Formally, for the weekly model, the hidden state vectors are:

$$h_t^{week} = \text{LSTM}_{week}^{SL}(o_t^{day}, h_{t+1}^{week}), \quad t \in \{7, 6, ...1\} \tag{18}$$

For the monthly model, the hidden state vectors are:

$$h_t^{month} = \text{LSTM}_{month}^{SL}(o_t^{day}, h_{t+1}^{month}), \quad t \in \{30, 29, ...1\} \tag{19}$$

The state vectors of the weekly and monthly models are concatenated with the daily state vector o_1^{day} into a single vector $h = [o_1^{day}, h_1^{week}, h_1^{month}]$, which is fed to the DDoS forecasting layer.

5.3 Hierarchical Stream Model

A drawback of the Short- and Long-Term Model above is that the size of utilizing history is limited to 30 days. For capturing infinitely long history without losing short and long term difference, we propose a fine-grained stacked LSTM model, arranging daily, weekly, and monthly history into a hierarchical structure. Figure 3 shows the model, which consists of three stacked LSTM layers.

Day-level. The first layer is the same as the vanilla sequence model, which maps the daily tweet representation

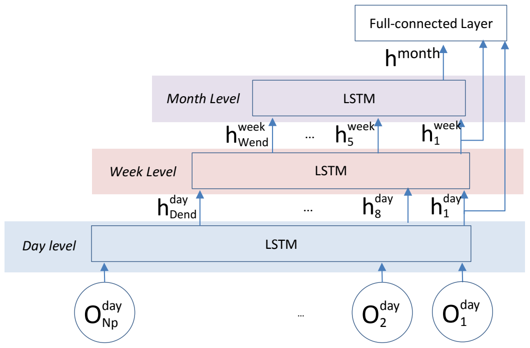

Figure 3: Hierarchical stream model.

sequence $\{o_{N_P}^{day}, ..., o_1^{day}, o_1^{day}\}$ into a hidden state sequence $\{h_{N_P}^{day}, ..., h_2^{day}, h_1^{day}\}$. We call this model the day-level model.

Week-level. The second layer is stacked on top of the day-level model, taking the sequence of hidden state vectors of every 7 days, namely $\{o_{N_W}^{week}, ..., o_2^{week}, o_1^{week}\} = \{h_{D_{end}}^{day}, ..., h_{D_8}^{day}, h_{D_1}^{day}\}$ as input, where N_W is the number of weeks, and $D_{end} = \lfloor N_P/7 \rfloor * 7 + 1$ is the last history day. The weekly hidden state vectors are:

$$h_t^{week} = \text{LSTM}^{WL}(o_t^{week}, h_{t+1}^{week}) \tag{20}$$

Month-level. The third layer is stacked on top of the week-level model, taking the sequence of hidden state vectors of every 4 weeks, $\{o_{N_M}^{month}, ..., o_2^{month}, o_1^{month}\} = \{h_{W_{end}}^{week}, ..., h_{W_5}^{week}, h_{W_1}^{week}\}$ as input, where N_M is the number of months, and $W_{end} = \lfloor N_W/4 \rfloor * 4 + 1$ is the last history week. The monthly hidden state vectors are:

$$h_t^{month} = \text{LSTM}^{ML}(o_t^{month}, h_{t+1}^{month}) \tag{21}$$

The hierarchical state vectors are concatenated into a single vector $h = [h_1^{day}, h_1^{week}, h_1^{month}]$, which is fed to the prediction model.

The hierarchical stream model has several potential advantages. Compared to the flat vanilla model, it explicitly captures history information of various degrees of granularity without constraining maximum history length. Compared to the short- and long-term stream model, back-prorogation training can be performed from the monthly layer to the weekly and daily layer, allowing tight integration of information by joint learning.

6 Experiments

6.1 Experimental Settings

Data. A DDoS event can be defined as a triplet (e, t, d), where e,t,d denote event, target and date, respectively. For building our benchmark dataset, we collect these three types of information from *ddosattacks.net*, a DDoS news website[2]. Most articles from the website correspond to an event e, and we employ one annotator to read through each article to extract the exact date d of the event. We use the *Stanford*

[2]http://www.ddosattacks.net/

Event e	**Target** t	**Date** d
Blizzard Battle.net Hit With Major DDoS Attack	Blizzard	April 15, 2016
Hacker Redirects DDoS Attack to Israeli Intelligence Site	Mossad, Israeli	April 5, 2016
Hackers Target NASA with DDoS Attack	NASA	March 23, 2016

Table 2: Example event triples.

Named Entity Recognizer[3] to extract name entities from the titles and contents of the news articles, and then manually extract the target t of each event e. In the end, we use this semi-automatic process to obtain 170 gold-standard events in the form of (e, t, d). Each event turns out to have a unique target, which demonstrates the sparsity of DDoS attack for each target, and the challenge of our task. Some examples are illustrated in Table 2.

Our goal is to monitor tweet streams for a certain target of concern. Events of a target t are predicted daily using history tweets that mention t. The target names are used as keywords to search and collect the related tweets. History tweet data are collected from August, 2015 to April, 2016, the same span for collecting DDoS news event. For each target, we collect about 200 posts per month[4], obtaining 17760 tweets related to all the 170 targets. Note that we only collect those tweets which mention a target explicitly in order to make sure that the tweets are related to the target.

We use 80 random targets for training, 60 for development, and the remaining 30 for testing. For each target, there is exactly one day in the dataset when a DDoS attack occurred, which is regarded as a positive sample. The remaining days, which do not see attacks, are considered negative samples.

Metric. We use the area under the precision-recall curve (AUC) [Davis and Goadrich, 2006], where *precision* is the fraction of predicted events that are correct, and *recall* is the fraction of gold events that are predicted.

6.2 Development Experiments

Imbalanced data

Our data set is highly imbalanced, with the ratio between positive and negative samples being very small. We investigate four typical strategies to address the issue, which include: 1) *under-sampleing-100*, using 1:100 imbalanced positive and negative samples. 2) *under-sampling-1* [Li *et al.*, 2011], using one sample of negative data for each positive data. 3) *over-sampling*, re-sampling the positive data to make the number of positive samples same as negative sample [Chawla *et al.*, 2002]. 4) *cost-sensitive* training [Sun *et al.*, 2007], where the cost weight for a negative sample is set to 0.01, according to a 1:100 imbalanced ratio between positive and negative samples, and the cost weight for a positive sample is 1. SVM[5] is taken as the basic classify model for all these approaches.

[3]http://nlp.stanford.edu/software/CRF-NER.shtml

[4]We use Google search to collect the history tweets.

[5]SVM^{light} is used as the implementation for the SVM classifier, http://svmlight.joachims.org/

Method	AUC
under-sampling-100	0.099
under-sampling-1	0.162
over-sampling	0.134
cost-sensitive	0.141

Table 3: Performance of different imbalanced data classification approaches.

Table 3 shows the results on the development set. All imbalanced training strategies lead to better AUC compared with the baseline approach. In addition, under-sampling outperforms both over-sampling and cost-sensitive training. which can be due to the fact that over-sampling yields multiple copies of the same features. This is consistent with the observation of Li *et al.* [2011] on imbalanced sentiment classification. Among the performances of under-sampling with different positive:negative ratio, a 1:1 ratio gives the best performance. We thus use under-sampling with balanced samples in the following subsections. Note that the development and test data are still imbalanced.

Correlation between tweets and DDoS events

As a baseline development experiment, we use a set of vanilla stream models to verify the correlation between history tweets and DDoS events. We compare six popular supervised classification models, which include:

- *Neg-Term-count* is the baseline sentiment-based model, we count the negative words from tweets each day, forecasting an attack if the number of negative words is larger than a threshold α, which is the average number of negative words on training data.
- *SVM* is a basic SVM model with bag-of-word features.
- *SVM-emb* is uses word embeddings rather than one-hot vectors for SVM features.
- *SVM-emb-senti* uses both common word embedding and sentiment-enriched embeddings in Section 4.1.
- *LSTM-emb* is the proposed *vanilla stream model* using skip-gram embeddings.
- *LSTM-senti* is the *vanilla stream model* with sentiment-enriched word embeddings.
- *LSTM-emb-senti* is the *vanilla stream model* with both both common word embedding and sentiment-enriched embeddings.

History tweets over a week are used. The results are shown in Table 4.

Is text useful for DDoS forecasting? As can be see from Table 4. A first research question to consider is the feasibility of stream DDoS forecasting. This services as a proof of concept. A random baseline gives very low AUC, given the sparsity of DDoS events. All text-based models outperform the random baseline significantly (p-value $<$ 0.01 using t-test), which demonstrates that text from social media is indeed informative for DDoS forecast.

Useful features. Next we study the sources of information that are useful. Neg-Term-count is a rather strong baseline,

Method	AUC
Neg-Term-count	0.233
SVM	0.164
SVM-emb	0.212
SVM-emb-senti	0.254
LSTM-emb	0.259
LSTM-senti	0.232
LSTM-emb-senti	0.293

Table 4: Correlation between tweets and DDoS events.

Method	AUC
LSTM-*day*	0.256
LSTM-*week*	0.293
LSTM-*month*	0.278

Table 5: Effectiveness of date range.

indicating that sentiment information is related to DDoS attack event forecasting. As a baseline using all words, SVM performs rather badly, showing the disadvantage of sparse models. Embedding features outperform SVM, and give results close to Neg-Term-count. Comparison between *SVM-emb* and *SVM-emb-senti*, and between *LSTM-emb* and *LSTM-emb-senti* shows that sentiment information is highly useful for DDoS event forecasting. Our final LSTM model outperforms other models significantly (p-value < 0.01 using t-test). This indicates the potential of supervised learning for the specific task given a neural model with sufficient representation learning power. In our case, LSTM can leverage non-local semantic information for sentence representation beyond sentiment signals.

Influence of date range

We measure the influence of different history date ranges on the vanilla LSTM stream model (Section 5.1). Shown in Table 5, LSTM-*week* outperforms both LSTM-*day* and LSTM-*month*. Intuitively, if the date range is too small, the stream model cannot capture sufficient historical information for prediction. However, a very large history date range may contain noise and irrelevant information. This suggests the usefulness of combining different history granularities.

Influence of stream models

We compare the different stream models in Section 5. In particular, LSTM^{VS} is the vanilla stream model (Section 5.1), LSTM^{SL} is the LSTM based stream model with short and long term history (Section 5.2), and LSTM^{HS} is the hierarchical LSTM stream model (Section 5.3). From Table 6, we can find that both LSTM^{HS} and LSTM^{SL} outperform the basic LSTM^{VS}, which only considers daily text stream information. This confirms the usefulness of combining differentially historical ranges. In addition, LSTM^{HS}, which considers the hierarchical tweet structure, is more effective than LSTM^{SL}, which only considers tweet sequence over a month. This demonstrates the advantage of LSTM^{HS} by considering unbounded history.

Method	AUC
LSTM^{VS}	0.293
LSTM^{SL}	0.321
LSTM^{HS}	0.346

Table 6: Effectiveness of stream models.

Method	AUC
Neg-Term-Count	0.234
SVM	0.154
SVM-emb-senti	0.254
LSTM^{HS}	0.305

Table 7: Final results on the test dataset.

6.3 Final Results

The final results on the test data set are given in Table 7. The proposed hierarchical structure model LSTM^{HS} brings significant improvements compared to the baseline Neg-Term-count and SVM-emb-senti models (p-value < 0.01 using t-test), which is in line with the development experiments. These results show the usefulness of leveraging social stream text for assisting DDoS detection and defence. On the other hand, the absolute AUC of our models are lower than 0.5, which demonstrates that DDoS forecasting is a highly challenging task. Nevertheless, our work can be regarded as proof-of-concept, since both models significantly outperform a random baseline without textual information.

7 Conclusion

We investigated the novel yet challenging task of DDoS forecasting using text streams from social media, which can be used to augment DDoS attack detection methods in the cyber security domain, by offering useful warnings *before* a DDoS begins. We proposed a fine-grained hierarchial stream model using deep learning, which arranges infinity long tweets history into a day-week-month hierarchy. Evaluation showed that social text streams are indeed informative for DDoS forecast, and our proposed hierarchical stream model is more effective compared to flat baseline text stream models and sentiment-driven models.

Acknowledgments

The corresponding authors is Yue Zhang. We thank our anonymous reviewers for their constructive comments, which helped to improve the paper. This work is supported by the Temasek Lab grant IGDST1403012 at Singapore University of Technology and Design.

References

[Bleakley and Vert, 2011] K. Bleakley and J.-P. Vert. The group fused Lasso for multiple change-point detection. *ArXiv e-prints*, June 2011.

[Carl *et al.*, 2006] Glenn Carl, George Kesidis, Richard R. Brooks, and Suresh Rai. Denial-of-service attack-detection techniques. *IEEE Internet Computing*, 10(1):82–89, January 2006.

[Chang *et al.*, 2016] Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. Expectation-regulated neural model for event mention extraction. In *Proceedings NAACL*, 2016.

[Chawla *et al.*, 2002] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357, 2002.

[Collobert *et al.*, 2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[Davis and Goadrich, 2006] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of ICML*, ICML '06, New York, NY, USA, 2006. ACM.

[Ding *et al.*, 2015] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *Proceedings of IJCAI*, pages 2327–2333, 2015.

[Duchi *et al.*, 2011] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[Hinton *et al.*, 2012] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Johnson and Zhang, 2015] Rie Johnson and Tong Zhang. Semi-supervised convolutional neural networks for text categorization via region embedding. In *Proceedings of NIPS*, pages 919–927, 2015.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of EMNLP*, pages 1746–1751, 2014.

[Li *et al.*, 2011] Shoushan Li, Guodong Zhou, Zhongqing Wang, Sophia Yat Mei Lee, and Rangyang Wang. Imbalanced sentiment classification. In *Proceedings of CIKM*, pages 2469–2472, 2011.

[Matthews, 2014] Tim Matthews. Incapsula survey : What ddos attacks really cost businesses. *Incapsula, Inc.*, 2014.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119, 2013.

[Mirkovic and Reiher, 2004] Jelena Mirkovic and Peter Reiher. A taxonomy of ddos attack and ddos defense mechanisms. *SIGCOMM Comput. Commun. Rev.*, 34(2):39–53, April 2004.

[Ritter *et al.*, 2015] Alan Ritter, Evan Wright, William Casey, and Tom M. Mitchell. Weakly supervised extraction of computer security events from twitter. In *Proceedings of WWW*, pages 896–905, 2015.

[Sun *et al.*, 2007] Yanmin Sun, Mohamed S. Kamel, Andrew K. C. Wong, and Yang Wang. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378, 2007.

[Tang *et al.*, 2014] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of ACL*, pages 1555–1565, 2014.

[Vo and Zhang, 2015] Duy-Tin Vo and Yue Zhang. Target-dependent twitter sentiment classification with rich automatic features. In *Proceedings of IJCAI*, pages 1347–1353, 2015.

[Zargar *et al.*, 2013] Saman Taghavi Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys and Tutorials*, 15(4):2046–2069, 2013.