# Detecting Denial-of-Service Attacks from Social Media Text: Applying NLP to Computer Security

**Nathanael Chambers** and **Ben Fry** and **James McMasters**
Department of Computer Science
United States Naval Academy
nchamber@usna.edu

## Abstract

This paper describes a novel application of NLP models to detect denial of service attacks using only social media as evidence. Individual networks are often slow in reporting attacks, so a detection system from public data could better assist a response to a broad attack across multiple services. We explore NLP methods to use social media as an *indirect* measure of network service status. We describe two learning frameworks for this task: a feed-forward neural network and a partially labeled LDA model. Both models outperform previous work by significant margins (20% F1 score). We further show that the topic-based model enables the first fine-grained analysis of how the public reacts to ongoing network attacks, discovering multiple "stages" of observation. This is the first model that both detects network attacks (with best performance) and provides an analysis of when and how the public interprets service outages. We describe the models, present experiments on the largest twitter DDoS corpus to date, and conclude with an analysis of public reactions based on the learned model's output.

## 1 Introduction

Distributed Denial of Service (DDoS) attacks have become more frequent and more severe in their impact. Coordinated attacks across several services are now common, yet there are fewer methods to detect multi-network events. Research into detecting and preventing single attacks focuses on *direct* evidence based on characteristics of a network itself, such as monitoring abnormal traffic. This paper instead investigates an aytpical source for multiple attacks with *indirect* evidence: social media text. Do users of attacked systems post on social media? What can be learned from comments? Can NLP learning models extract enough information from user posts to detect attacks? Previous work on attack detection with social media is sparse, and focused on detecting trending words. This paper is the first to learn models of language without 'attack' dictionaries and seed words. The goal is the real-time detection of attacks without network data. Our secondary goal is to illustrate NLP applications to computer security topics.

Research on information extraction from social media has shown that many types of events in the world can be reliably detected from the language that users post. Several approaches have been shown effective in identifying events like earthquakes (Sakaki et al., 2010), concerts and product releases (Ritter et al., 2012), and other natural disasters (Neubig et al., 2011). Detecting DDoS attacks is not too dissimilar from these goals. An attack is a real event in the world, and it takes a community by surprise. This paper thus adopts ideas from NLP, but applies them to the unique application of DDoS detection.

Social media is obviously not the only way (nor the most direct) to monitor network services and attacks. There are several commercial services that directly measure outages, such as norsecorp[1]. These perform *direct* monitoring of network response. We do not propose social media as a better alternative, but rather as an alternative that enhances direct monitoring. Social media also brings its own unique benefits. For instance, social media does not require a priori knowledge of which networks should be monitored. It can also help detect "soft" outages like slowdowns and account blockages, things that direct monitoring cannot always detect. Therefore, this paper is not suggesting a replacement, but rather a new source of valuable information. It is a monitoring architecture that is not constrained by a predefined list of services.

Our goal in using social media is driven by the

[1] http://map.norsecorp.com



*Proceedings of NAACL-HLT 2018*, pages 1626–1635
New Orleans, Louisiana, June 1 - 6, 2018. 

hypothesis that as a network attack unfolds, its users go through a series of observational stages that can be automatically learned and detected. The first stage is a state of confusion and basic symptom observation, as seen in the following real tweets from Twitter:

> *hey linode what's happening? I can't login, my servers are down and you don't reply on mails?*
>
> *is xbox live experiencing some issues?*

These tweets don't discuss an attack even though that is what was occurring. Later stages then develop into direct commentary as the community coelesces to a belief that an attack is the cause:

> *Breaking: Band of America website rumored to be under DDoS attack.*
>
> *Citi Bank & BofA under Massive DDoS attack*

We show that our proposed LDA-based model can effectively identify these stages. There is very little previous work in this area, and that which exists focuses entirely on the second stage. Ritter et al. (2015) proposed models that include hard-coded keywords like 'DDoS' and phrases like '<entity> is down'. Their work helped identify attacks with social media, but these identifications tend to be *after* the news has already reported it. Early symptoms that are discussed don't use words like 'DDoS' because the conclusion has not yet been drawn. We thus propose the first learning models that identify *early* attack discussions: the first is a neural network, and the second is a broader topic model that provides better insight into the evolution of an attack.

Finally, our last goal is to model the themes and topics that users notice during network attacks. This is a somewhat subjective analysis, but it is backed by an empirical model trained from real-world data. Not only do we empirically produce state-of-the-art results, 25% gains over previous work, we also show that our detection system learns topics of discussion previously uninvestigated in the security field.

The core contributions of this paper are as follows: (1) a *25% improvement* over previous work on attack detection, (2) we present the first neural network results on detecting network attacks from social media, (3) we present a partially labeled LDA model for detecting network attacks with state-of-the-art results, (4) the PLDA enables the first analysis of the evolution of an attack as seen through its users, and (5) we make available the largest list of historical DDoS attacks to date.

## 2 Previous Work

The most relevant line of research to this paper is *event extraction* from social media. Space prohibits describing all work; the major approaches vary in levels of supervision. Ritter et al. (2012) used a Latent Dirichlet Allocation model to identify events in text *without* labeled data. They showed you can cluster and extract events like concerts, movies, and performances into a calendar. General event detection from social media has continued in several threads (Benson et al., 2011; Popescu et al., 2011; Anantharam et al., 2015; Wei, 2016; Zhou et al., 2017). Guo et al. (2013) link tweets to news stories using an annotated dataset. Sakaki et al. (2010) detect earthquake events by monitoring tweets with keywords like 'earthquake'. This is similar in goal to our paper, but different in approach and brittle in its application. We crucially do *not assume* that users use known keywords and phrases.

We take inspiration from the thread of work on flu detection (Lamb et al., 2013; Broniatowski et al., 2013). Their work leverages mentions of an event ('caught', 'sick', 'flu'), and then uses human annotators to label these mentions as relevant to the desired event (flu). We also identify mentions of an event, but we crucially differ by *not knowing* event words a priori. We believe a typical user does not know what a DDoS attack is, so we cannot assume certain language will be used. A major contribution of this work is the first analysis of how the (perhaps uninformed) public perceives DDoS attacks as they occur.

The first work (to our knowledge) on attack detection from Twitter was Motoyama et al. (2010). They tracked a single phrase "X is down" and experimented with whether outages could be detected from its counts. They use a trend detection formula to notice increases of this one phrase to trigger an alert. We compare against this strong baseline later. The work by Kergl et al. (2016) uses social media to identify users who discuss zero-day exploits. While not directly related to the work in this paper, its success reinforces the hypothesis that social media contains useful data for computer security monitoring.

The main thread in this area is the learning model from Ritter et al. (Ritter et al., 2015) and follow-on work (Kergl, 2015; Chang et al., 2016). They proposed a weakly supervised learner to identify cybersecurity events from Twitter. They

| **Attacked Services** (dd-mm-yy) | | | |
|---|---|---|---|
| Ancestry.com | 16-06-14 | Lib. Congress | 18-07-16 |
| BBC Website | 14-03-15 | Newsweek | 29-09-16 |
| Call of Duty | 20-09-14 | Planned Parent. | 29-07-15 |
| DNS | 21-10-16 | Reddit | 19-04-13 |
| Github | 27-03-15 | Spamhaus | 18-03-13 |

Table 1: A sample of 10 DDoS events in our dataset. A 20 day span is collected around each attack date.

collected tweets that contain the word *'DDoS'*, and then collected a set of known network attack days. The known days provided a training set from which they trained this weakly-supervised classifier on the 'DDoS' tweets. An important constraint in their approach, similar to flu research, is the need to use a seed word(s). Seed words enable the collection of a very relevant training set, but it limits the system because it depends on social media posts to use these words, and more importantly, to actually know that a DDoS is happening. We instead hypothesize that attacks are preceded by users who first observe symptoms of the attack, and don't directly discuss a DDoS or use related attack words. Our analysis shows we match several orders of magnitude more tweets.

## 3 Datasets

We manually created a dataset of historical DDoS attacks that include the entity attacked and the date of attack. Most past attacks are difficult to identify hour ranges, so we used a full 24-hour day as our granularity. We included 6 attacks with sufficient volume from previous work (Ritter et al., 2015), but we grew this set to 50 attacks based on our own investigations into recent years, mostly through web search results for 'DDoS attacks'. Table 1 lists *some* of these for illustration of its diversity. The full list is at `www.usna.edu/Users/cs/nchamber/data/ddos/`

For each of these known attacks, we collected tweets that contained the attacked entity's name in a 20-day period: 17 days prior to the attack, 1 day on the attack, and 2 days following. We wanted a sufficient lead up to the attack to include previous work's trending model (Motoyama et al., 2010), and to provide non-attack days for evaluation. The days surrounding the known attack date are labeled NOT-ATTACK, and the attack day itself as ATTACK. Sometimes an attack lasted longer than a single day, in which case the days following were also labeled as ATTACK, as appropriate.

The historical attacks span the years 2012-2016. We split the data so that years 2012, 2015, and 2016 comprise the training set, 2013 is the development set, and the year 2014 is the test set only used for computing final experiment numbers. Splitting on years (rather than months or entities) guards against test set pollution into our training set. The evaluation on the 2014 test set is thus an unbiased experiment because nothing from the entire year is included in training. For the experiments, we use the union of training and development to train the final models that are then used to evaluate on the test year 2014. There are 200 test days in 2014, 50 in dev, and ˜500 in training.

The full dataset consists of 50 attack days over approximately 800 days and 2 million tweets. The only previous work on this area used seed words to pull out around 9-10 thousand tweets. Our dataset is *more than 2 orders of magnitude larger*. The reason is due to the larger number of attacks we collected, but notably our tweets are more diverse and varied because we don't require hard-coded target words and phrases to match.

Formally, the dataset is 800 labeled datums:

$$d_i = (Entity, Date, Tweets, Label) \quad (1)$$

where $d_i \in D$ and $D$ is the set of all days. $Entity$ is the attacked network service, $Date$ is the calendar date, and $Tweets$ are all tweets on that date mentioning that entity. $Label$ is a binary variable: ATTACK or NOT-ATTACK. Even though the day following an attack often includes attack discussion, it is still labeled NOT-ATTACK. Only if the attack was ongoing is the next day labeled ATTACK.

## 4 Models

Two primary goals motivate the models we propose and evaluate. The first goal is the automatic classification of attack and non-attack events. We propose the first neural network for this task, and move on to a generative model based on topic models. We evaluate their relative performance and compare against baselines from prior work.

The second goal is a model that enables analysis of user behavior during the evolution of an attack. What do people notice? What do people focus on? These are important questions for the security community that NLP models can help answer. We present a brief subjective study using the generative model, show how learned topics change over time, and discuss the data's implications.

### 4.1 Task Formulation

As discussed in Section 3, our input is labeled datums: $d_i = (Entity, Date, Tweets, Label)$. Each datum in the training set has a known label of ATTACK or NOTATTACK based on our historical knowledge of which entities were attacked on which days. We thus formulate the task as a binary classification over 24 hour days. We train models with the labeled training set, and report final numbers on test. In order to tune parameters, we use the development set to run grid searches over the models' parameters. The test set was always excluded from these until the final experiments.

### 4.2 Logistic Regression

Our first baseline model is logistic regression with word-based features. The following were used:

**Unigrams**. All words in the tweets were lowercased and punctuation stripped.
**Bigrams**. All bigrams are included & lowercased. Start and stop symbols are used for tweet boundaries, and punctuation included as separate tokens.
**Bigram/Trigram Patterns**. Since we know the entity, we parameterize the entity's mention in each tweet, and build bigrams and trigrams around them. For instance, the phrase "reddit is slow" is included as a trigram feature "X is slow". This allows learning across instances, so "spamhaus is slow" is included as the same feature.

We use the Stanford CoreNLP toolkit with default settings to train the model. We removed all features that occurred only once. This model is referred to as **LogisticReg** below.

### 4.3 Neural Network Models

Neural networks have made significant advancements in many NLP areas. Two of the main reasons for this are (1) improved representation of the features, and (2) stacking of hidden layers provides a better data fit.

We experimented with two feed-forward neural networks using word embeddings. We first trained a simple one-layer neural network that is similar to logistic regression, but with embeddings as input (instead of frequency counts). This is the **Neural-1** model. We then trained a two-layer network with hidden layer $h$ of size $m$, and a softmax output layer to the binary label task. This is the **Neural-2** model.

The input to both of these models is as a Continuous Bag of Words (CBOW) model (Mikolov et al., 2013). Unlike logistic regression, the only features input to the network are unigrams (a tweet's individual tokens). Each unigram $u$ has a word embedding $x_u$ of length $n$, and they are all input as a weighted average. The reader is referred to Mikolov (2013) for more CBOW background.

We do not use pre-trained word embeddings, but instead learn them from our data. The embedding values are initialized randomly $[0, 1]$ from the uniform distribution. We used DyNet as our modeling toolkit (Neubig et al., 2017).

Overfitting is often a problem with neural networks, and we quickly found our models doing so. We thus applied $0.5$ dropout for regularization (Srivastava et al., 2014). We experimented with other dropout values but did not see reliable gains or losses, so kept it at the typical $0.5$ value.

We trained other networks without word embeddings, but instead "one-hot vectors" where the vector is the size of the vocabulary. This model did not perform as well and required more memory, so we do not report its results. Additional hidden layers did not improve either, as expected from the observed overfitting.

### 4.4 Constrained Topic Modeling

While the neural models above improve over previous work and baselines, they are difficult to interpret what is actually learned. One of the applications of this paper is to analyze what people discuss during network attacks. The hidden layers and word embeddings are opaque and difficult from which to draw conclusions.

In contrast, a generative model that represents words explicitly as probability distributions allows for easier post-analysis. It also may generalize better to this task because training data is more sparse and noisy. While we have 2 million tweets, orders of magnitude more than previous work, this is still modest in size with 800 days. To make matters worse, the dataset is biased toward NOTATTACK. $95\%$ of the training set is NOTATTACK, leaving few training instances that are actually labeled as ATTACK. As shown in the next section, the neural models tend to overfit to these small signals. Further, we observed that online discussions go through different stages (Section 5.4), and the neural model merges stages to its detriment.

We thus propose a model inspired by Latent Dirichlet Allocation (LDA) (Blei et al., 2003), but a model carefully designed to the unique applica-

tion at hand. For readers unfamiliar with LDA, the model can be thought of as a clustering algorithm, and an overview of LDA and its variants can be found in Blei's survey (Blei, 2012).

#### 4.4.1 LDA for Security Events

A traditional LDA model can learn general topics on our dataset with the hope that attack topics bubble up. Our initial experiments found this to be insufficient and the non-attack days were full of distracting topics.

For the goal of analyzing attack discussion, we need to encourage the LDA model to learn attack-specific topics. We draw heavily from Labeled LDA (Ramage et al., 2009). Each word is assigned a topic as in standard LDA, but topics can have a known label from the document. This is relevant to this paper because we know which days are attacks (in training). Thus, when a tweet is on an attack day, we assign the tweet a label ATTACK, and bias the Labeled LDA learner to assign its words to an attack-related topic.

What labels do we have in our data? ATTACK and NOTATTACK labels are first, but we also know which entities are mentioned in tweets, providing labels to learn entity-specific topics. We can label a tweet about reddit as REDDIT, and bias the Labeled LDA algorithm to assign a reddit-specific topic. The following tweet is an example:

| reddit isn't responding maybe DNS is wrong |
|---|

This tweet mentions two entities (reddit and dns), and it occurs on a known attack day for reddit in training. This tweet thus has 3 labels (attack, reddit, dns). The tweet's tokens can draw from 1 of 3 topics, which is good but a bit constraining. One of the premises of this paper is that people discuss attacks on social media in a variety of ways (not just one topic). They might discuss hackers, the DDoS attack itself, or just general downtime. The vanilla Labeled LDA (Ramage et al., 2009) is then too strict, but there is a multi-topic extension in the Partially Labeled Dirichlet Allocation (PLDA) (Ramage et al., 2011). PLDA is a version that instead of having one topic per label, it learns $N_l$ topics for each label $l$. For our example tweet, tokens can now be labeled with 1 of $\sum_l N_l$ topics. We use $N_{attack} = 5$ and $N_{reddit} = N_{dns} = 5$ in our experiments[2], so this tweet would sample from 15 topics.

[2]This is about reddit, but each company has its own 5 topics. Experiments have 40 companies for total 200 topics.

Formally, let a tweet be defined as a document $d$ with words $w \in W_d$. Each document has a set of labels $\Lambda$. This set $\Lambda$ always contains the BACKGROUND label to capture general twitter conversations. Further, if a network or company is mentioned in the document, $\Lambda$ also contains the company's label (e.g., MICROSOFT). Finally, the label ATTACK is added to $\Lambda$ if $d$ is an attack day and $W_d$ includes the attacked network's name. Each word $w \in d$ has a latent label $l$ and a latent topic $z_l$. For readers familiar with plate diagrams, this diagram is shown in Figure 1. Readers will notice its similarity to Ramage et al. (2009) with the addition of a new $\beta$ parameter and the important change that the attack label is *observed* in training, but *unobserved* in test. When observed (in training), we favor assigning words to attack topics. When unobserved, we want to dissuade but still allow for it when the text strongly favors attack topics. To this end, our PLDA differs from standard use in that $\psi$ is generated from a non-symmetric dirichlet with hyperparameters $\boldsymbol{v}$ (a vector of length $\sum_l N_l$) defined as:

$$v_i = \begin{cases} \alpha, & \text{if } i \notin AttackTopics \\ \beta, & \text{if } i \in AttackTopics \ \ \& \ \ attack \notin \Lambda \\ \beta * 10, & \text{if } i \in AttackTopics \ \ \& \ \ attack \in \Lambda \end{cases}$$

This is a non-symmetric dirichlet prior that enables attack labels to be chosen ($\beta$) without an observed attack day. Every tweet must be able to sample from attack topics because we need to label future unknown (unlabeled) attacks. The PLDA in the literature assumes full labeling at all times, but our task is more difficult. When ATTACK is observed, its smoothing parameter's value is $\beta * 10$ because of our heightened certainty, rather than simply $\beta$ when unobserved.

The number of attack topics $N_a$ and background topics $N_b$ was chosen empirically from dev set performance. For simplicity and to avoid overfitting, we chose a single number $M = 5$ of company topics $\forall_c \, N_c = M$ that is the same across all companies and also $N_{attack} = M$. Only $N_b$ was varied in our parameter tuning stage to discover how many background topics were necessary.

Space prohibits a full mathematical description of PLDA, so we direct the reader to Ramage et al. (2011) for details. Those unfamiliar with the above formalities can think of it as a soft clustering of words that is accomplished through sampling.

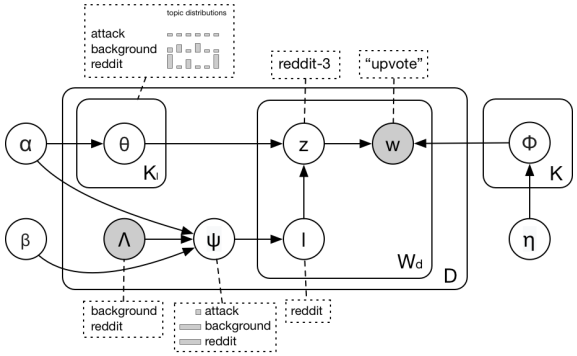


Figure 1: Plate diagram of the partially labeled topic model (PLDAttack). Dotted boxes are example values of a non-attack tweet discussing Reddit.

#### 4.4.2 Inference and Classification

Inference in this model is performed with collapsed Gibbs sampling, sampling $l$ and $z_l$ in turn while holding all other variables constant. A single iteration requires looping over the entire dataset and assigning labels (topics) to each token on each day. We repeat this process until convergence of the joint probability of the model. After convergence, we hold distributions $\theta, \psi$ constant and run 20 more sampling iterations. Each word is then assigned the topic that was sampled the most in the 20 iterations.

Once sampling completes, this PLDAttack model provides us two very useful tools. First, the assigned topics enables us to use it as a classifier for our target task: DDoS detection from social media. Second, the topics themselves allow us to create timelines of discussions about DDoS attacks. This provides a higher-level analysis of what people say (Learned Topics).

With all words labeled, we want the model to make a prediction about an entity $e$ on a given day $d$. Was the entity attacked[3]? We compute the probability of an attack using the labels themselves without any modification:

$$P(attack|d) = \frac{\sum_{w \in W_d} 1\{z_w \in Attack\}}{|W_d|} \quad (2)$$

where $|W_d|$ is the number of words in all tweets on day $d$ and $Attack$ is the set of attack topics. $1\{x\}$ is the indicator function. If this probability is greater than a threshold, the entity/day is labeled as an attack. Otherwise, it is not an attack.

The cutoff threshold depends on a typical probability that is assigned to tweets, and how frequent an entity is actually mentioned on a given day. We use the development set to identify the optimal cutoff to maximize our F1 score.

[3]Or, is the entity currently under attack? This paper is an early detection attempt, leaving live-tracking to future work.

## 5 Evaluation and Results

All experiments are conducted on the dataset described in Datasets. The task is a binary classification of ATTACK or NOTATTACK given a day of tweets. All parameters are optimized on the development set: we treat attack days as known on training days, but hidden from the development and test days. We calculate F1 score on the development attack days, and optimize parameters using a basic grid search. For the final reported results, we combine train+dev into one observed training set, and the test set is now included in sampling, but with unobserved attack days. Since the PLDAttack model is probabilistic, all reported numbers are an average of 10 independent runs.

We use ATTACK F1 as the main evaluation target; the harmonic mean between precision and recall. Applications overly concerned with missing attacks would optimize to recall $R$. We chose $F1$ as a happy balance between a quality classifier (good precision $P$) and a useful classifier (good recall $R$). We report all three scores for both the ATTACK and NOTATTACK labels, but optimize to F1 during parameter search on the development set.

### 5.1 Trending Baselines

**Entity Trending**: This baseline follows the hypothesis that a website under attack is mentioned more than usual, and language analysis is not required. There is credence to this idea. Much of our data includes a spike in discussion on the attack day (however, some non-attack days show similar frequency spikes). We model frequency trending with an exponential decay function similar to that in Motoyama et al. (2010). It uses an Exponentially Weighted Moving Average:

$$A_t = \alpha * n_t + (1 - \alpha) * A_{t-1} \quad (3)$$

where $A_t$ is the EWMA of day $t$, $n_t$ is the number of tweets on day $t$, and $\alpha$ determines how the current day's count affects the moving average. We then need a threshold $T_t$ to determine when $n_t$ is trending. This is based on a moving deviation $\sigma^2$:

$$D_t = n_t - A_{t-1} \quad (4)$$

| | Non-Attack | | | Attack | | |
|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 |
| Freq Baseline | .99 | .87 | .92 | .29 | **.83** | .43 |
| Motoyama'10 | .97 | .94 | .95 | .35 | .58 | .44 |
| LogisticReg | .97 | .75 | .85 | .14 | .67 | .24 |
| Neural-1 | .96 | .97 | **.96** | .54 | .47 | .49 |
| Neural-2 | .97 | .96 | **.96** | .55 | .53 | .53 |
| PLDAttack | .96 | .96 | **.96** | **.61** | .52 | **.55** |

Table 2: Results on the held-out test set of 200 test datums. Motoyama'10 is a trending phrase baseline.

$$\sigma_t^2 = \beta * D_t^2 + (1 - \beta) * \sigma_{n-1}^2 \quad (5)$$

Given this deviation, the threshold is then:

$$T_t = M_{t-1} + \epsilon * \sigma_{t-1} \quad (6)$$

If $n_t > T_t$ for a day, we signal an ATTACK.

**Pattern Trending**: This modified baseline exactly duplicates Motoyama et al. (2010). Their approach looks for trending mentions that match the pattern, *'X is down'*. The X is substituted with the company's name. We use the same equation 6, but frequency $n_t$ is defined as how many tweets contain the pattern (instead of just 'X').

### 5.2 Experiment Results

The test set results for baselines and models are shown in Table 2. All improvements are statistically significant as indicated using McNemar's two-tailed test. The trending baselines have high recall. When an attack is happening, the network does indeed trend on social media. Precision is low, however, because non-security events also cause discussions. The neural models outperform the baselines, and a hidden layer (Neural-2) is definitely needed for increased detection. The training set of 500 documents is still small for neural training, though. Neural models have many parameters, and they overfit to our training set despite regularization with dropout, reducing dimensions, and removing hidden layers. Even still, we improved over the Motoyama baseline by 20% relative F1.

PLDA (PLDAttack) showed the highest precision when classifying an ATTACK. Since its recall was similar to the neural models, it produced the best F1 score. This is a 25% relative improvement over previous work. PLDAttack generalizes to the dataset slightly better than the neural models.

| Topic 1 | Topic 2 | Topic 3 | Topic 4 | Topic 5 |
|---|---|---|---|---|
| issues | after | attack | down | down |
| working | service | ddos | servers | anonymous |
| having | #news | under | site | megaupload |
| issue | hit | twitter | hacked | takes |
| email | attack | dns | website | sites |
| update | website | attacks | goes | music |
| services | hackers | massive | taking | claims |
| problem | #tech | services | web | doj |

Table 3: The top words in each of 5 attack topics learned from the entire train/test dataset.

We now have two good approaches for detecting attacks: neural models and topic modeling. The remaining question is to analyze what people are actually discussing, and it is here that topic modeling further shines.

### 5.3 Learned Topics

The generative model is attractive because we can use its learned distributions to produce insight into what people discuss. It is more precise in our experiments, and the neural models are simply too difficult to analyze.

Table 3 shows some of the attack topics that were learned on one of our model's runs (results are an average of training runs). As can be seen, though topics are similar, they capture subtle differences in what people discuss during an attack. The first topic represents tweets about news surrounding the event. These often contain links, and show up after the attack is made known to news agencies. In contrast, topic 3 is more general about servers down and specific services such as email. The fourth topic captures discussion about Anonymous and the claims that the group makes about taking sites down. This was obviously learned as an artifact of our data which contains several Anonymous-related events.

One of the most useful analyses we can do with this type of model is track topic evolution over time. Figure 2 illustrates one attack day and the dramatic jump of the attack topics. For simplicity, we plot the 5 attack topics, and hide the others as they are generally flatter across the bottom. This shows that social media became aware on the 9th hour, and only took one more hour to reach peak intensity. What is perhaps most useful with a timeline is understanding the impact of an attack on its users. There is a fair bit of chatter the day following the event, showing that people do not easily forget such attacks and depending on the entity, this could have effects on how people engage. We

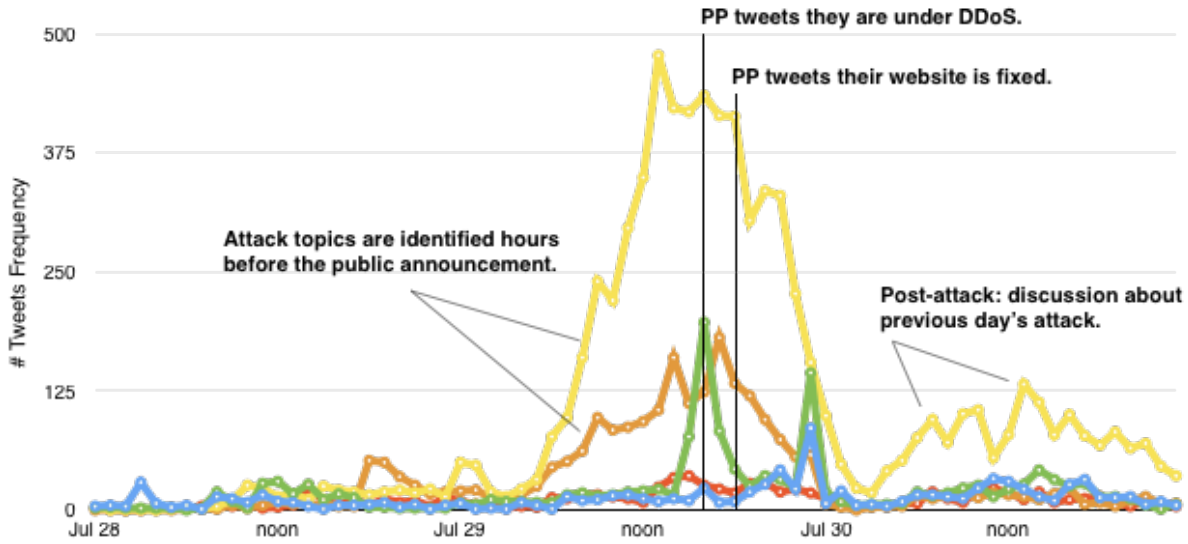


Figure 2: Attack topics during a 3-day period around a DDoS on the Planned Parenthood (PP) website. The attack was announced at 5pm on July 29. The green topic spikes with PP's first tweet about the DDoS. The yellow & brown topics rise hours **before** the announcement. They decrease after the attack, similar to the red 'news' topic.

also see that Planned Parenthood (in this example) delayed in announcing the attack. Whether this is tactical or simply how long it took to realize the event, PLDAttack offers a natural way to discover just how soon patrons (or the public in general) became aware of the issue without an announcement. Decision making around these events might be guided by helpful NLP tools such as this.

Finally, we note previous work tracked tweets with 'DDoS' in it. There were ˜50 such tweets, but our models instead matched *tens of thousands*. Previous seed-based work cannot produce this type of analysis.

### 5.4 Attack Stages in Online Discussions

To analyze the PLDAttack model's strengths, we split attack days into "spiking" chunks to identify common stages of online chatter. We use topic frequency spikes for Reddit to provide diversity in analysis from the Planned Parenthood Figure 2. Reddit is a community based website attacked on April 19, 2013. We identified four distinct stages of a DDoS attack on social media: (1) Symptom, (2) Inference, (3) Confirmation, and (4) Resumption. Figure 3 shows examples from each stage.

The **Symptom Stage** is the earliest sign of a problem with user observations of the network service. These aren't comments about malicious attacks, but statements about authentication problems and unresponsive websites. This is the most difficult stage for a learner (**false positives**). Services can have trouble for a variety of reasons, not necessarily DDoS attacks. Some of our evaluation data includes inoccuous problems, and these caused a decrease in precision.

The **Inference Stage** includes guesses about the cause of the previous stage's symptoms. These can and do intermix with the Symptom Stage. As seen in the reddit examples in Figure 3, some of the users wonder if they broke reddit rather than a malicious act occurring. We also see an example of someone guessing that it is a DDoS attack, but without actual knowledge of it.

The **Confirmation Stage** occurs when the website publicly announces an attack. Not all attacks have a public announcement. Our error analysis revealed this to be the cause of several **false negatives**. When the public is not directly informed, the learning algorithms must rely on symptoms and inferences only. Previous work largely isolated itself to attacks with a Confirmation Stage, for instance, relying on the 'DDoS' keyword to be present (Ritter et al., 2015).

Finally, the **Resumption Stage** is when the network service is restored. The reddit examples show people commenting on the resumption, and making jokes about the previous situation. Similar to the Symptom Stage, this stage contributes to **false positives** because it also occurs with normal routine network problems, not just malicious acts.

### 5.5 Error Analysis

Identifying the four stages above led to a natural method of studying errors in our model. We

**Symptom Stage**
Please come back Reddit! I'm bored.
Reddit won't authenticate me. #lifeisover
Reddit is calling me a robot and won't let me use it

**Inference Stage**
There is a DDoS attack on Reddit right now?
i broke reddit? wat? I think we crashed Reddit #wow

**Confirmation Stage**
wow turns out reddit is being DDoS attacked right now
Reddit is experiencing a malicious DDoS attack
Reddit's reward for the Boston bombing? DDOS attacks.

**Resumption Stage**
@JpDeathBlade Reddit is back and in full force.
Reddit may be returning.
It's ok, Reddit is back up. Go home, nothing to see here.

Figure 3: Examples from the four stages of a social media response to DDoS attacks.

chose a sample of false positives and false negatives, and manually looked at these incorrect decisions to align common mistakes with how they related to the 4 stages. Looking at the false positives, the majority are from the Symptom and Inference Stages. Looking at false negatives, we found attacks where the network did **not** make a public statement, so the Confirmation Stage was missing.

These stages of course do not account for all of the mistakes that are made. Precision is at 61% in our best model, leaving room for improvement. Other reasons for errors included distractor events. For example, the Boston Bombing occurred near the Reddit DDoS. The preceding days included thousands of tweets talking about the *attack* in Boston. This is obviously a different type of attack, and the machine learners were led astray.

### 5.6 Robustness

A danger in many stochastic processes is finding one good run and only reporting on those results. We thus compare our our model across runs and found the topics to be somewhat robust and steady. We chose five random runs of the best performing model (the one from Figure 2) and focused on the largest attack topic. Is this topic learned in all runs? Not only was the same topic subjectively learned in each run, we graphed the observed frequency of this largest attack topic from 5 of the 10 runs. Not only did it maintain the same frequency, but also the same general shape across the runs. Space prohibits more illustration, but the graph can be found on our data website: `www.usna.edu/Users/cs/nchamber/data/ddos/`

## 6 Discussion

The core conclusion from our experiments is that social media does indeed contain signals to identify DDoS attacks. Our proposed neural network outperformed previous work (Motoyama et al., 2010) by 20% F1, a very large margin. Even though online users are an *indirect* source of evidence, the 53% F1 from the neural network shows that useful information can be extracted from text.

We further improved results with the generative PLDAttack model based on topic modeling, achieving a smaller 4% increase over the neural net but 25% over the prior trending approach. Although neural networks have significant advantages over LDA-based models, PLDAttack offers advantages by enabling deeper analysis of what people say, what topics are discussed, and how attack discussions evolve over time on Twitter. For instance, it enabled Figure 2 to illustrate the different topics that people discuss during such an event.

Can these results be used in a DDoS detection framework? We believe it can. PLDAttack recall may not be as high as desired, but it can be increased by adjusting the prediction cutoff probability $\lambda$. We empirically set the cutoff based on dev set performance to optimize F1. However, a detection system may desire to optimize recall at the expense of precision, thus choosing a lower $\lambda$ and forcing the system to predict attacks more often. This would increase false positives, but with a human in the loop, it is manageable to monitor.

This paper thus proposed two NLP models for learning to identify DDoS attacks from social media without network data. They leverage *indirect* evidence described by users when they post online about service availability. By identifying the *early topics* before public announcements, we see this as an important step toward a broad-scale monitoring system not dependent on individual network reporting. We hope our datasets and models encourage further efforts in NLP and Computer Security. Models and data are available online: *www.usna.edu/Users/cs/nchamber/data/ddos/*

## 7 Acknowledgments

This work was supported in part by a grant from the Office of Naval Research. We also thank the support of the DoD HPC Modernization Office for enabling our undergraduate education and research. Finally, thanks to EdinburghNLP for hosting me while wrapping up this work. Slàinte!

## References


Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):43.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 389–398. Association for Computational Linguistics.

David M Blei. 2012. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

David A Broniatowski, Michael J Paul, and Mark Dredze. 2013. National and local influenza surveillance through twitter: an analysis of the 2012-2013 influenza epidemic. *PloS one*, 8(12):e83672.

Ching-Yun Chang, Zhiyang Teng, and Yue Zhang. 2016. Expectation-regulated neural model for event mention extraction. In *Proceedings of NAACL-HLT*, pages 400–410.

Weiwei Guo, Hao Li, Heng Ji, and Mona T Diab. 2013. Linking tweets to news: A framework to enrich short text data in social media. In *ACL (1)*, pages 239–249. Citeseer.

Dennis Kergl. 2015. Enhancing network security by software vulnerability detection using social media analysis extended abstract. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on*, pages 1532–1533. IEEE.

Dennis Kergl, Robert Roedler, and Gabi Dreo Rodosek. 2016. Detection of zero day exploits using real-time social media streams. In *Advances in Nature and Biologically Inspired Computing*, pages 405–416. Springer.

Alex Lamb, Michael J Paul, and Mark Dredze. 2013. Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of HLT-NAACL*, pages 789–795.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Marti Motoyama, Brendan Meeder, Kirill Levchenko, Geoffrey M Voelker, and Stefan Savage. 2010. Measuring online service availability using twitter. *WOSN*, 10:13–13.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Graham Neubig, Yuichiroh Matsubayashi, Masato Hagiwara, and Koji Murakami. 2011. Safety information mining-what can nlp do in a disaster-. In *IJCNLP*, volume 11, pages 965–973.

Ana-Maria Popescu, Marco Pennacchiotti, and Deepa Paranjpe. 2011. Extracting events and event descriptions from twitter. In *Proceedings of the 20th international conference companion on World wide web*, pages 105–106. ACM.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. 2009. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 248–256. Association for Computational Linguistics.

Daniel Ramage, Christopher D. Manning, and Susan Dumais. 2011. Partially labeled topic models for interpretable text mining. In *Proceedings of KDD*.

Alan Ritter, Oren Etzioni, Sam Clark, et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM.

Alan Ritter, Evan Wright, William Casey, and Tom Mitchell. 2015. Weakly supervised extraction of computer security events from twitter. In *Proceedings of the 24th International World Wide Web Conference (WWW)*.

Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Wei Wei. 2016. *Probabilistic Models of Topics and Social Events*. Ph.D. thesis, Arizona State University.

Deyu Zhou, Xuan Zhang, and Yulan He. 2017. Event extraction from twitter using non-parametric bayesian mixture model with word embeddings. In *European Association for Computational Linguistics (EACL)*.