REST: A Thread Embedding Approach for Identifying and Classifying User-Specified Information in Security Forums

Joobin Gharibshah, Evangelos E. Papalexakis, Michalis Faloutsos
University of California - Riverside, CA
900 University Ave, Riverside, California 92557
jghar002,epapalex,michalis@cs.ucr.edu

Abstract

How can we extract useful information from a security forum? We focus on identifying threads of interest to a security professional: (a) alerts of worrisome events, such as attacks, (b) offering of malicious services and products, (c) hacking information to perform malicious acts, and (d) useful security-related experiences. The analysis of security forums is in its infancy despite several promising recent works. Novel approaches are needed to address the challenges in this domain: (a) the difficulty in specifying the "topics" of interest efficiently, and (b) the unstructured and informal nature of the text. We propose, REST, a systematic methodology to: (a) identify threads of interest based on a, possibly incomplete, bag of words, and (b) classify them into one of the four classes above. The key novelty of the work is a multi-step weighted embedding approach: we project words, threads and classes in appropriate embedding spaces and establish relevance and similarity there. We evaluate our method with real data from three security forums with a total of 164k posts and 21K threads. First, REST robustness to initial keyword selection can extend the user-provided keyword set and thus, it can recover from missing keywords. Second, REST categorizes the threads into the classes of interest with superior accuracy compared to five other methods: REST exhibits an accuracy between 63.3-76.9%. We see our approach as a first step for harnessing the wealth of information of online forums in a user-friendly way, since the user can loosely specify her keywords of interest.

Keywords: Embedding, Classification, Security Forums

Introduction

Security forums hide a wealth of information, but mining it requires novel methods and tools. The problem is driven by practical forces: there is useful information that could help improve security, but the volume of the data requires an automated method. The challenge is that there is a lot of "noise", there is lack of structure, and an abundance of informal and hastily written text. At the same time, security analysts need receive focused and categorized information, which can help their task of shifting through it further. We define the problem more specifically below.

Given a security forum, we want to extract threads of interest to a security analyst. We consider two associated problems that together provide a complete solution. First, the input is all the data of a forum, and the user specifies its interest by providing one or more bag-of-words of interest. Arguably, providing keywords is a relatively easy task for the user. The goal is to return all the threads that are of interest to the user, and we use the term **relevant** to indicate such threads. A key challenge here is how to create a robust solution that is not overly sensitive to the omission of potentially important keywords. We use the term **identification** to refer to this problem.

Second, we add one more layer of complexity to the problem. To further facilitate the user, we want to group the relevant threads into classes. Again, the user defines these classes by providing keywords for each class. We refer to this step as the **classification** problem. Note that the user can specify the classes of interest fairly arbitrarily, as long as there is training data for the supervised-learning classification.

There is relatively limited work on extracting information from security forums, and even less work on using embedding techniques in analyzing online forum data. We can group prior work in the following categories. First, there is work that analyzes security forums to identify malicious activity (Portnoff et al. 2017; Tavabi et al. 2018; Gharibshah, Papalexakis, and Faloutsos 2018). Moreover, there are some efforts to detect malicious users (Li and Chen 2014; Marin, Shakarian, and Shakarian 2018) and emerging threats on forums and other social networks (Sapienza et al. 2017; 2018). Second, there are several studies on analyzing online forums without a security focus (Zhang, Culbertson, and Paritosh 2017; Cong et al. 2008). Third, there is a large body of work in embedding techniques for: (a) analyzing text in general (Mikolov et al. 2013; Joulin et al. 2017), and (b) improving text classification (Devlin et al. 2018; Wang et al. 2018; Tang, Qu, and Mei 2015). Also, note that there exist techniques that can do transfer learning between forums and thus, eliminate the need to have training data for every forum (Gharibshah, Papalexakis, and Faloutsos 2018).

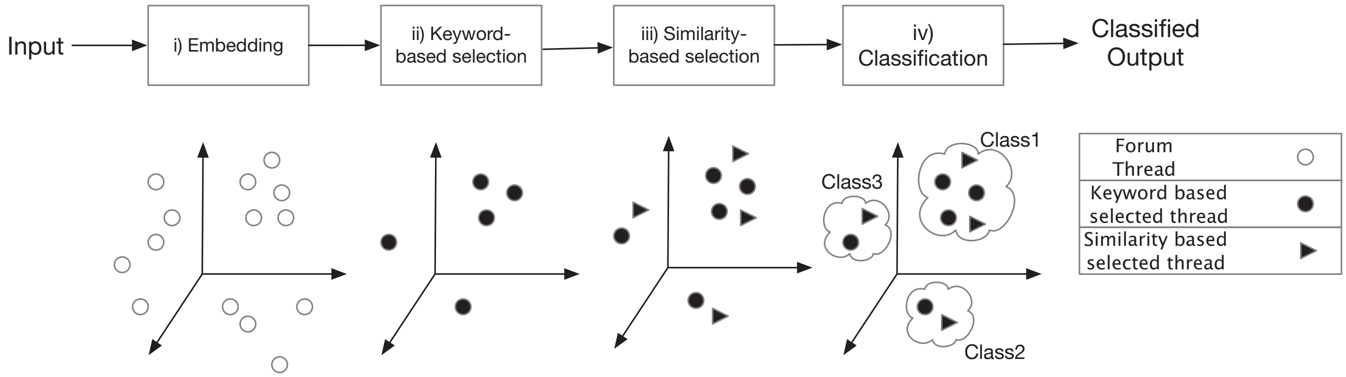

Input
i) Embedding
ii) Keyword-based selection
iii) Similarity-based selection
iv) Classification
Classified Output
Class1
Class2
Class3
Forum Thread ○
Keyword based selected thread ●
Similarity based selected thread ▶

Figure 1: An overly-simplified overview of analyzing a forum using the REST approach: i) **project** all threads to embedding space, ii) **select** relevant threads using keyword-based selection, iii) **expand** by adding similar threads, iv) **classify** the threads into classes using supervised learning. We illustrate the embedding space as a three dimensional space.

We discuss related work in more detail in our related work section.

We propose a systematic approach to identify and classify threads of interest based on a multi-step weighted embedding approach. Our approach consists of two parts: (a) we propose a similarity-based approach with thread embedding to extract relevant threads reliably, and b) we propose a weighted-embedding based classification method to group relevant threads into user-defined classes.

The key technical foundation of our approach relies on: (a) building on a word embedding to define thread embedding, and (b) conducting similarity and classification at the thread embedding level. Figure 1 depicts a high-level visualization of the key steps of our approach: (a) we start with a word embedding space and we define a thread embedding where we project the threads of the forum, (b) we identify relevant threads to the user-provided keywords, (c) we expand this initial set of relevant threads using thread similarity in the thread embedding, (d) we develop a novel weighted embedding approach to classify threads into the four classes of interest using ensemble learning. In particular, we use similarity between each word in the the forums and representing keywords of each class in order to up-weight the word embedding vectors. Then we use weighted embeddings to train an ensemble classifier using supervised learning.

We evaluate the proposed method with three security forums with 163k posts and 21k unique threads. The users in these forums seem to have a wide range of goals and intentions. For the evaluation, we created a labelled dataset of 1350 labeled threads across three forums, which we intend to make available to the research community. We provide more information on our datasets in the next section.

Our results can be summarized into the following points:

a. Providing robustness to initial keyword selection. We show that our similarity-based expansion of the user-defined keywords provides significant improvement and stability compared to simple keyword-based matching. First, the effect of the initial keyword set is minimized: by going from 240 to 300 keywords, the keyword-based method identifies 25% more threads, while the similarity based method increases by only 7%. Second, our approach increases the number of relevant threads by 73-309% depending on the number of keywords. This suggests that our approach is less sensitive to omissions of important keywords.

b. The relevant threads are 22-25% of the total threads. Our approach reduces the amount of threads to 22-25% of the initial threads. Clearly, these results will vary depending on the keywords given by the user and the type of the forum.

c. Improved classification accuracy. Our approach classifies threads of interest in four different classes with an accuracy of 63.3-76.9% and weighted average F1 score 76.8% and 74.9% consistently outperforming five other approaches.

Our work in perspective. Our work is building block towards a systematic, easy to use, and effective mining tool for online forums in general. Although here we focused on security forums, it could easily apply to other forums, and provide the users with the ability to define topics of interest by providing one or more set of keywords. We argue that our approach is easy to use since it is robust and forgiving w.r.t. the initial keyword set.

Definitions and Datasets

	OffensComm.	HackThisSite	EthicalHackers
Posts	25538	84,745	54176
Users	5549	5904	2970
Threads	3542	8504	8745

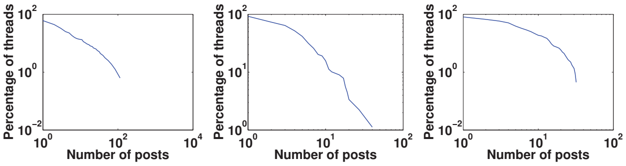
Table 1: The basic statistics of our forums.

We have collected data from three different forums: OffensiveCommunity, HackThisSite and EthicalHackers. These forums seem to bring together a wide range of users: system administrators, white-hat hackers, black-hat hackers, and users with variable skills, goals and intentions. We briefly describe our three forums below.

a. OffensiveCommunity (OC): This forum seems to be on the fringes of legality. As the name suggests, the forum focuses on “offensive security”, namely, breaking into systems. Indeed, many posts provide step by step instructions on how to compromise systems, and advertise hacking tools and services.

b. HackThisSite (HT): As the name suggests, this forum has also an attacking orientation. There are threads that describe how to break into websites and systems, but there are also more general discussions about the users’ experiences in cyber-security.

EthicalHackers (EH): This forum seems to consist mostly of “white hat” hackers, as its name suggests. Many threads are about making systems more secure. However, there are many discussions with malicious intents are going on in this forum. Moreover, there are some notification discussions to alert about emerging threats.



(a)OffensComm. (b)HackThisSite (c)EthicalHackers

Figure 2: CCDF of the number of Post per thread (log-log scale).

Basic forum statistics. We present basic statistics of our forums in Table 1. We also study some of their properties and make the following two observations.

Observation 1: More than half of the threads have one post! In Figure 2, we plot the complementary cumulative distribution function of the number of post per thread for our forums. We observe the skewed distribution that describes the behavior of large systems. In addition, the distribution shows that more than half of thread has one single post in the threads and 73% of the threads has one or two posts in threads.

Observation 2: The first post defines the thread. Prior research (Zhang, Culbertson, and Paritosh 2017) seems to confirm something that we intuitively expect: the first post of the thread pretty much defines the thread. Intrigued, we sampled and manually verified that this seems to be the case. Specifically, we inspected a random sample of 10% of the relevant threads (found by our approach), and we found that more than 97% of the follow up posts fall in line with the topic of the thread: while a majority of them, express appreciation, agreement etc. For example, the follow up posts to a malicious tutorial in OffensiveCommunity were: “Great Tut”, “Thank you for sharing”, “Nice post”, “Work[s] great for me!”

Defining the classes of interest. As we explained in the introduction, we want to further help a security analyst by giving them the ability to define classes of interest among the threads of interest. These are user-defined classes. To ground our study, we focus on the following classes, which we argue could be of interest to a security analyst.

a. Alerts: These are threads where users are reporting about being attacked by a hackers or notifying about exploits and vulnerabilities. An example from EthicalHackers is a thread with the title “Worm Hits Unsecured Space Station Laptops” and the first line of the first post is “NASA spokesman Kelly Humphries said in a statement that this was not the first time that the ISS had been affected by malware, merely calling it a “nuisance.””

	OffensComm.		HackThisSite		EthicalHackers	
Labeled	450		450		450	
	#	%	#	%	#	%
Hacks	202	45%	31	7%	42	9%
Services	204	45%	286	64%	166	37%
Alerts	27	6%	20	4%	78	18%
Experiences	17	4%	113	25%	164	36%

Table 2: Our groundtruth data for each forum and the breakdown per class.

b. Services: These are threads where users are offering or requesting malicious hacking services or products. An example from OffensiveCommunity is a thread with the title “Need hacking services” and this first line “Im new to this website. Im not a hacker. Would like to hire hacking services to hack email account, Facebook account and if possible iPhone.”

c. Hacks: These are threads where users post detailed instructions for performing malicious activities. The difference with the above category is that the information is offered for free here. An example from OffensiveCommunity is a thread titled “Hack admin account in XP, Vista, Windows 7 and Mac - Complete beginners guide!!” with a first line: “Hack administrator account in XP OS – Just by using command prompt is one of the easiest ways (without installation of any programs).....”. As expected, these posts are often lengthy as they convey detailed information.

d. Experiences: These are threads where users share their experience related to general security topics. Often users provide a personal story, a review or an article on a cybersecurity concept or event. For example, in HackThisSite a thread titled “Stupid people stories”, the author explains cyber-security mistakes that he made.

The sets of keywords which “define” each class are shown in Table 5. Clearly, these sets will be provided by the user depending on classes of interest. Note that these keywords are also provided to our annotators as hints for labeling process.

Establishing the Groundtruth

For validating our classification method, we need groundtruth to do both the training and the validation. We randomly selected 450 threads among the relevant threads from each forum as selected by the identification part. The labelling involves five annotators that manually label each thread to a category based on the definitions and examples of the four classes which we listed above. The annotators were selected from a educated and technically savvy group of individuals to improve the quality of the labels. We then combine the “votes”, and assign the class selected by the majority.

We assess the annotators’ agreement based on the Fleiss-Kappa coefficient and we show the results in Table 3. We

Label	Hacks	Services	Alerts	Experiences
OffensComm.	0.778	0.702	0.816	0.732
HackThisSite	0.953	0.966	0.793	0.875
EthicalHackers	0.682	0.733	0.766	0.620

Table 3: Assessing the annotator agreement using the Fleiss-Kappa coefficient for each class for our three datasets.

see that there is a high annotator agreement across all forums as the Fleiss-Kappa coefficient is 78.6. 92.6, 70.3 for OffensiveCommunity, HackThisSite and EthicalHackers respectively.

With this process, we labelled 1350 posts in three forums and we present our labeled data in Table 2. We make our groundtruth available to the researchers in the community in order to foster follow up research. [1]

Challenges of simple keyword-based filtering

Given a set of keywords, the most straightforward approach in identifying relevant documents (or threads here) is to count the combined frequency with which these keywords appear in the document. A user needs to identify the keywords that best describe the topics and concepts of interest, which can be challenging for non-trivial scenarios (Wang et al. 2016). We outline some of the challenges below.

- The user may not be able to provide all keywords of interest. In some cases, the user is not aware of a term, and in some cases, this not even possible: consider the case where we want to find the name of a new malware that has not yet emerged.
- Stemming, variations and compound words is a concern. The root of a word can appear in many different versions: e.g. hackers, hacking, hacked hackable, etc. There exist partial solutions for stem but challenges still remain (Jivani 2011).
- Spelling errors and modifications and linguistic variations. Especially for an international forum, different languages and backgrounds can add noise.

The above challenges motivated us to consider a new approach that uses a small number of indicative keywords to create a seed set of threads, and then use similarity in the embedding space to find more similar threads, as we describe in the next section.

Identifying threads of interest

We present our approach for selecting relevant threads starting from sets of keywords provided by the user. Our approach consists of the following phases: (a) a keyword matching step, where we use the user-defined keywords to identify relevant threads that contain these keywords, and (b) a similarity-based phase, where we identify threads that are "similar" to the ones identified above. The similarity is established at the word embedding space as we describe later.

Symbol	Description
v_i	Word i in a forum
$\vec{v_i}$	Embedded vector for word i
$v_{i,k}$	Value of dimension k in embedded vector for word i
t_r	Thread r
m	The dimensions of the word embedding space
n	Number of words in a thread
d	Number of words in a forum
$W(t_r)$	Set of words in thread r
D	Set of words in a forum
$\vec{P}(e)$	Embedding projection of entity e (word, thread etc)
$Sim(w, c)$	Similarity of vectors w and c
w_k	The "center of gravity" word for class k
$\vec{\beta}_k$	Affinity vector of class k
$\vec{\beta}_k[i]$	Value of Affinity vector of class k at index i
WS_l	Keyword set l for identifying relevant threads
T_{key}	Keyword threshold in identifying relevant threads
T_{sim}	Similarity threshold in identifying relevant threads

Table 4: The symbol table with the key notations.

Phase 1: Keyword-based selection

Given a set or sets of keywords, we identify the threads where these keywords appear. A simple text matching approach can distinguish all occurrence of such keywords in the forum threads. In more detail, we follow the steps below:

Step 1: The user provide a set or sets of keywords WS_l, which capture the user's topics of interest. Having sets of keywords enables the user to specify combinations of concepts. For example, in our case we use, the following sets: (a) hacking related, (b) exhibiting concern and agitation, and (c) searching and questioning.

Step 2: We count the frequency of each keyword in all the threads. This can be done easily with elastic search or any other straightforward implementation.

Step 3: We identify the relevant threads, as the threads that contain a sufficient number of keywords from each set of keywords WS_l. This can be defined by a threshold, T_{key_l}, for each set of keywords.

Going beyond simple thresholds in this space, we envision a flexible system, where the user can specify complex queries that involve combinations of several different keyword sets WS_l. For example, the user may want to find threads with: (a) at least 5 keywords from set WS_1 and 3 keywords from WS_2, or (b) at least 10 keywords from WS_3.

Phase 2: Similarity-based selection

We propose an approach to extract additional relevant threads based on their similarity to existing relevant threads. Our approach is inspired by and combines elements from earlier approachs (Mikolov et al. 2013; Shen et al. 2018), which we discuss and contrast with our work in the related work section.

Overview of our approach. Our approach follows the steps below, which are also depicted visually in Figure 1. The input is a forum, a set of keywords, and set of relevant threads, as identified by the keyword-based phase above.

Step 1. Determining the embedding space. We project every word as a point in a m-dimensional space using a word

[1] Data is provided at the following link: https://github.com/icwsmREST2019/RESTDATA.

embedding approach. Therefore, every word is represented by a vector of m dimensions.

Step 2. Projecting threads. We project all the threads in an appropriately constructed multi-dimensional space: both the relevant threads selected from the keyword-based selection and the non-selected ones. The thread projection is derived from the projections of its words, as we describe below.

Step 3. Identifying relevant threads. We identify more relevant threads among the non-selected threads that are "sufficiently-close" to the relevant threads in the thread embedding space.

The advantage of using similarity at the level of threads is that thread similarity can detect high-order levels of similarity, beyond keyword-matching. Thus, we can identify threads that do not necessary exhibit the keywords, but use other words for the same "concept". We show examples of that in Tables 9 and 10.

Our similarity-base selection in depth. We provide some details in several aspects of our approach.

Step 1: in depth. We train a skip-gram word embedding model to project every word as a vector in a multi-dimensional space (Mikolov et al. 2013). Note that we could not use pre-trained embedding models, since there are many words in our corpus that do not exist in the dictionary of previous models.

The number of dimensions of the word embedding can be specified by the user: NLP studies usually opt for a few hundred dimensions. We discuss how we selected our dimensions in our experiments section.

At the end of this step, every word v_i is projected to $\vec{P}(v_i)$ or $\vec{v_i}$, a real-value m-dimensional vector, $(v_i[1], v_i[2], ..., v_i[m])$. A good embedding ensures that two words are similar, if they are close in the embedding space.

Step 2: in depth. We project the threads in an $2m$-space, by "doubling" the m-dimensional space that we used for words as we will show below. The thread projection is a function of the vectors of its words and captures both the average and the maximum values of the vectors of its words.

a. Capturing the average: $P_{avg}(t_r)$. Here, we want to capture the average "values" of the vectors of the words in the thread. For thread t_r, the average projection, $P_{avg}(t_r)$ is calculated as follows for each dimension l in the m-dimensional word space:

$$\vec{P}_{avg}(t_r)[l] = \frac{1}{|W(t_r)|} \cdot \sum_{v_i \in W(t_r)} \vec{v_i}[l], \tag{1}$$

Recall that $W(t_r)$ is the set of words of the thread. For simplicity, we refer to projection of word v_i as $\vec{v_i}$ instead of the more complex $\vec{P}(v_i)$.

b. Capturing the high values: $P_{max}(t_r)$. Averaging can fail to represent adequately the "outlier" values, and to overcome this, we calculate a vector of maximum values, $\vec{P}_{max}(t_r)$, for each thread. For each dimension l in the word embedding, $P_{max}[l]$ is the maximum value of that dimension over all existing l-dimension values among all the words in the thread, which we can state more formally below:

$$\vec{P}_{max}(t_r)[l] = \max_{v_i \in W(t_r)} \vec{v_i}[l] \tag{2}$$

Finally, we create the projection of thread t_r by using both these vectors, $\vec{P}_{avg}(t_r)$ and $\vec{P}_{max}(t_r)$, as this combination has been shown to provide good results (Shen et al. 2018). Specifically, we concatenate the vectors and we create the thread representations in an $2m$-dimensional space.

$$\vec{P}(t_r) = (\vec{P}_{avg}(t_r), \vec{P}_{max}(t_r)) \tag{3}$$

Step 3: in depth. We identify similar threads at the $2m$-space-dimensional space of thread embedding from step 2. We propose to use the cosine-similarity determine the similarity among threads, which seems to give good results in practice. Most importantly, we can control what constitutes a *sufficiently-similar* thread using a threshold T_{sim}. The threshold needs to strike a balance between being too selective and too loose in its definition of similarity. Furthermore, note the right threshold value also depends on the nature of the problem and the user preferences. For example, a user may want to be very selective, if the resources for further analyzing relevant threads is limited or if the starting number of threads is large.

Classifying threads of interest

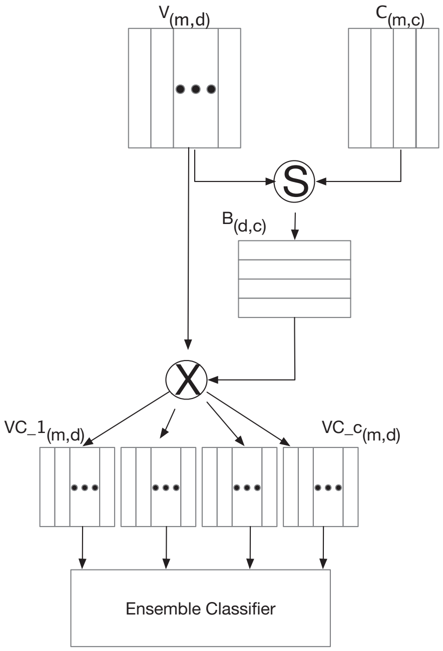


Figure 3: A visual overview of our classifier

We present our approach for classifying relevant threads into user defined classes. To ground the discussion, we presented four classes on which we focus here, but our approach can be applied for any number and type of classes as long as there is training data for the supervised learning.

Defining Affinity. We use the term **affinity**, $\vec{\beta}_k[i]$, to refer to the "contribution" of word v_i in a thread towards our belief that the thread belongs to class k.

Recall also that each class k is characterized by a group of words that we denote as $WordClass_k$. These sets of words are an input to our algorithm and in practice they will be provided by the user.

High-level overview creating our classifier. Our approach consists of the following steps, which are visually represented in Figure 3.

Step 1. We create a representation of every class k into the word embedding space by using the words that define the class, $WordClass_k$.

Step 2. For all the words in the forum, we calculate the affinity of the word v_i for each class k, $\vec{\beta}_k[i]$.

Step 3. For each class, we create a weighted embedding by using the affinity to adjust the embedding projection of each word for each class.

Step 4. We use weighted embedding to train an ensemble classifier using supervised learning.

Using the classifier. Given a thread, we calculate its projection in the embedding space, and then we pass it to the classifier to determine its class.

Our algorithm in more detail. In the remainder of this section, we provide a more in depth description of the algorithm.

Step 1: in depth. For each class k, we use the set of words $WordClass(k)$, and to define a representation, $\vec{w}_k$, for that class in the word embedding space. We project each word in $WordClass(k)$ to the embedding space by using the same word embedding model, which we trained in the previous section. Then, we define the class vector w_k to be the average of the word embeddings of the words in $WordClass(k)$ similarly to equation 1. Note that these class embedding vectors correspond to each column of the matrix $C_{(m,c)}$ in Figure 3, where m in the dimension of the embedding and c is the number of classes.

Step 2: in depth. The affinity of each word v_i in the forum for each class is calculated by the similarity of the word v_i to $\vec{w}_k$, which represents the class in this space. We calculate the proximity using the cosine similarity, as follows:

$$Sim(\vec{v_i}, \vec{w}_k) = \frac{\vec{v_i} \cdot \vec{w}_k}{||\vec{v_i}|| \cdot ||\vec{w}_k||} \tag{4}$$

Then, for each class k, we create vector $\vec{\beta_k}$ whose element $[i]$ corresponds to the affinity of word v_i of the forum D. Specifically, we normalize the values by using *Softmax* of the similarity vector $Sim(v_i, w_k)$ as follows:

$$\vec{\beta_k}[i] = \frac{exp(Sim(\vec{v_i}, \vec{w}_k))}{\sum_{y_j \in D} exp(Sim(\vec{y_j}, \vec{w}_k))}, \tag{5}$$

where $y_j \in D$ iterates through all the words in the forum. Note that $\vec{\beta_k}$ corresponds to a row k in matrix $B_{d,c}$ in figure 3, where c is the number of classes and d is the total number of words in the forum.

Step 3: in depth. For each class k, we create a "custom" word embedding, $VC_k(m, d)$ in Figure 3. Each such matrix that is focused on detecting threads of k and it will be used in our ensemble classification.

Hacks	Services	Alerts	Experiences
Tutorial	tool	announced	article
guide	price	reported	story
steps	pay	hacked	challenge

Table 5: $WordClass$, the set of words which ”define” each class.

For each class, we create, $VC_k(m, d)$, a class-specific word embedding by modifying the word projections, $\vec{v_i}$ using the affinity of the word $\vec{\beta}_k[i]$ for class. Formally, we calculate VC_k by calculating column $VC_k[*, i]$ as follows:

$$VC_k[*, i] = \vec{\beta}_k[i] \cdot \vec{v_i} \tag{6}$$

where $\vec{\beta}_k[i]$ is the affinity value of word v_i for class k.

For each thread t_r, we calculate the projection of the thread by calculating $\vec{P}_{avg}(t_r)$ and $\vec{P}_{max}(t_r)$ using the modified word projections, $\vec{\beta_k}[i] \cdot \vec{v_i}$, captured in the $VC_k(m, d)$ matrix and using equations 1 and 2. Finally, we create the projection of each thread in the $2m$-space, using equation 3.

Step 4: in depth: We use weighted embeddings of threads to train an ensemble classifier using supervised learning.

For each class k, we train the classifier by using the weighted representation vector in a supervised learning. Each VC_k in Figure 3 becomes the basis for a classifier with weighted penalty in favor of class k. The ensemble classifier combines the classification results from each VC_k classifier using the max-voting approach (Kittler et al. 1998).

Using contextual features. Apart from the words in the forum, we can also consider other types of features, which we refer to as contextual features of the threads. One could think of various such features, but here we list the features that we use in our evaluation: (1) number of newlines, (2) length of the text, (3) number of replies in the thread (following posts after the first post), (4) average number of newlines in replies, (5) average length of replies, and (6) the aggregated frequency of the words of each bag-of-words set provided by the user.

These features capture contextual properties of the posts in the threads, and provide additional information not necessarily captured by the words in the thread. Empirically, we find that these features improve the classification accuracy significantly. The inspiration to introduce such features came from manually inspection of posts and threads. For example, we observed that Hacks and Experiences usually have longer posts than other. Moreover, Hacks threads contain a larger number of newline characters. An interesting question is to assess the value of such metrics when used in conjuction with word-based features.

Experimental Results

We present our experimental results and evaluation of our approach.

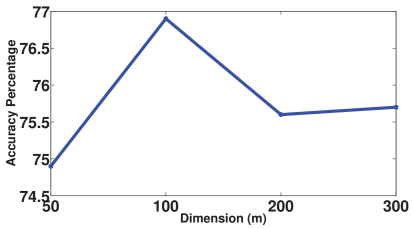


Figure 4: Selecting the number of dimensions of word embedding: the accuracy of REST for different dimensions in OffensiveCommunity.

Conducting our study

We use the three forums that presented in Table 1 and the groundtruth, which we created as we explained in section definitions.

Keywords sets: We considered three keyword sets to capture relevant threads. These keywords set are: (a) hacking related, (b) exhibiting concern and agitation, and (c) searching and questioning. We collected a set of more than 290 keywords in three sets. We started with a small core group of keywords, which we expanded by adding their synonyms using thesaurus.com and Google's dictionary. We ended up with 68, 207 and 17 keywords for the three groups respectively.

These keyword sets are used in extracting relevant threads with the keyword-based selection. We select a thread, if it contains at least one word from each keyword set: $T_{key_1}, T_{key_2}, T_{key_3} >= 1$. As we discussed earlier, there are many different ways to perform this selection in the presence of multiple groups of words and depending on the needs of the problem.

Pre-processing text: As with any NLP method, we do several pre-processing steps in order to extract an appropriate set of words from the target document. First we tokenize the documents in to bigrams, then we remove the stopwords, numbers and IP addresses based on a recent work (Gharibshah, Papalexakis, and Faloutsos 2018). In addition, here we opt to focus on the title and the first post of a threads instead of using all the posts. Our rationale is based on the two observations regarding the nature of the threads: (a) most of them have one post anyway, and (b) the title and the post typically define their essence. In the future, we will examine the effect of using the text from more posts from each thread.

Identification: Implementation choices. The identification algorithm requires some implementation choices, which we describe below.

Embedding parameters: We set the window size to 10 and we tried several different values as the dimension of the embedding between 50-300, and we found that $m = 100$ with the highest accuracy as depicted in Figure 4 and m is in the range of choice of other studies in this space.

Similarity threshold: $T_{sim} = 0.96$**.** The similarity threshold T_{sim} determines the "selectiveness" in identifying similar threads, as we described in a previous section. We find that a value of 0.96 worked best among all the different values we tried. It strikes the balance between being: sufficiently selective to filter out non-relevant threads, but sufficiently flexible to identify similar threads.

Classification: Implementation choices. We present the implementation choices for our classification study.

Evaluation Metrics: We used the accuracy of the classification along with the average weighted F1 score, which is designed to take into consideration the size of the different classes in the data.

Our classifier. We use random forest as our classification engine, which performed better than several others that we examined, including SVM, Neural Networks, and K-nearest-neighbors. Results are not shown due to space limitations.

Class defining words: The set of keywords we have used for each class are as shown in Table 5.

Baseline methods. We evaluate our approach against five other state of the arts methods, which we briefly describe below.

- **Bag of Words (BOW)**: This methods uses the word frequency (more accurately the TFIDF value) as its main feature (McCallum and Nigam 1998; Gharibshah et al. 2017; Jin et al. 2016).
- **Non-negative Matrix Factorization (NMF)**: This method uses linear-algebra to represent high-dimensional data into low-dimensional space, in an effort to capture latent features of the data (Lee and Seung 1999).
- **Simple Word Embedding Method (SWEM)**: There is a family of methods that use the word2vec as their basis, and use a recently proposed method (Shen et al. 2018).
- **FastText (FT)**: Similar to NMF and SWEM, FastText represents words and text in a low dimensional space (Joulin et al. 2017).
- **Label Embedding Attentive Model (LEAM)**: This is the most recent approach (Wang et al. 2018) claims to outperform other state of art methods including PTE (Tang, Qu, and Mei 2015). We used their provided linear implementation of their attentive model.
- **Bidirectional Encoder Representations from Transformers (BERT)** : This is a new pre-trained Deep Bidirectional Transformer for Language Understanding introduced by Google (Devlin et al. 2018). BERT provides contextual representation for text, which can be used for a wide range of NLP tasks. As we discuss later, BERT did not provide good results initially, and we created a tuned version to make the comparison more meaningful.

Results 1: Thread Identification

We present the results from the identification part of our approach.

Our similarity-based method is robust to the number of initial keywords. We want to evaluate the impact of the number of keywords to the similarity based method. In Figure 5, we show the robustness of each identification methods to the initial set of keywords for OffensiveCommunity. By adding 60 keywords, from 240 to 300, the keyword-based method identifies 25% more threads, while the similarity

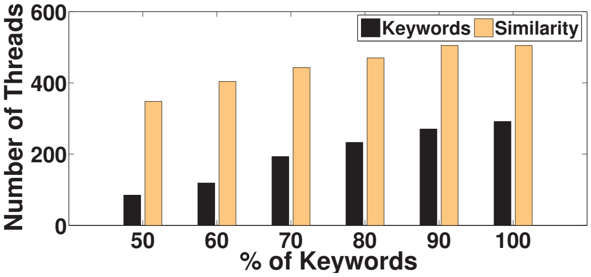


Figure 5: The robustness of the similarity approach to the initial keywords: number of relevant threads as a function of the number of keywords for OffensiveCommunity.

Relev. Threads	OffensComm.	HackThisSite	EthicHack
Keyword	291	840	893
Similarity	505	1121	1360
Total	796	1961	2753
Total(%)	22%	23%	25%

Table 6: The relevant threads and their identification method: keywords and similarity. The total percentage refers to the selected threads over all the threads in the forum.

based method has only 7% increment. Similarly, doubling the initial size of the keywords results in 242% increase for the keyword-based method but only 45% in the similarity-based method.

We argue that our approach is more robust to the initial number of keywords. First, with less number of keywords, we retrieve more threads. Second, an increase in the number of keywords has less relative increase in the number of threads. This is an initial indication that our approach can achieve good results, even with a small initial set of keywords.

Evaluation of our approach: High precision and reasonable recall. We show that our approach is effective in identifying relevant threads. Evaluating precision and recall would have been easy if all the threads in a forum were labelled. Instead, we use an indirect method to gauge recall and precision as we describe below.

Indirect estimation of recall. We consider as "groundtruth" the relevant threads that we find with set of keywords in keyword-based selection method and report how many of those threads that our method finds with only 50% of the keywords in similarity-based selection. The experiment is shown in Figure 5. We use only 50% of the keywords to extract the relevant threads with the similarity selection approach, and then compare it with the relevant threads identified with larger set of keywords [60-100]%. We show in Table 7 that with 50% of the keywords we can identify more than 60-70% of the relevant threads, which we identify if we have more keywords available.

Estimating precision. To evaluate precision, we want to identify what percentage of the retrieved threads are relevant. To this end, we resort to manual evaluation. We have labeled 300 threads from each dataset retrieved with 50% of the keywords and we get our annotators to identify if they are relevant. We show the results in Table 8. We understand that on average more than 97.5% of the threads identified with the similarity based method are relevant with an inter-annotator agreement Fleiss-Kappa coefficient of 0.952.

keywords %	60	70	80	90	100	Avg.
OffensComm.	78.2	76.9	72.9	70.8	70.1	70.94
HackThisSite	74.82	72.01	70.68	69.92	69.74	71.43
EthicHack	68.41	60.4	60.8	57.2	56.51	60.67

Table 7: Identification: Indirect "gauge" of Recall: We report how many threads our method finds with 50% keywords compared to the keyword based selection with larger sets of keywords [60-100]% .

	OffensComm.	HackThisSite	EthicHack	Avg.
Precision	98.2	97.5	97.0	97.5

Table 8: Identification Precision: the precision of the identified thread of interest with the similarity-based method.

The power of the embedding in determining similarity. We find that the similarity step identifies threads that are deemed relevant to a human reader, but are not "obviously similar", if you examine the threads word for word. We provide a few examples of threads that were identified by the keyword-based selection, and the related similar threads that our approach identified. Table 9 and 10 illustrate how the retrieved thread are similar to the target thread conceptually, without matching linguistically.

A four-fold thread reduction. Our approach reduces the amount of threads to only 22-25% of the initial threads as shown in Table 6. Figure 6 depicts the same data visually. Clearly, these results will vary depending on the keywords given by the user and the type of the forum.

Results 2: Thread Classification

We present the results of our classification study.

REST compared to the state-of-the-art. Our approach compares favourably against the competition. Table 11 summarizes the results of the baseline methods and our REST for three forums. REST outperforms other baseline methods with at least 1.4 percentage point in accuracy and 0.7 percentage point in F1 score, except BERT. First, using BERT "right out of the box" did not give good results initially. However, we fine-tuned BERT for this domain. BERT performs poorly on two sites, HackThisSite and EthicalHackers, while it performs well for OffensiveCommunity. We attribute this to the limited training data in terms of text size and also the nature of the language users use in such forums. For example, we found that the titles of two misclassified threads contained typos and used unconventional slang and writing structure " Hw 2 gt st4rtd with r3v3r53 3ngin33ring 4 n00bs!!", "metaXploit 3xplained fa b3ginners!!!". We intend to investigate BERT and how it can be tuned further in future work. Note that methods BOW and NMF did not assign any instances to the minority classes correctly, therefore the value of F1 score in Table 11 is reported as NA.

Selection Method	Title	Post
Keyword selected	[ULTIMATE] How to SPREAD your viruses successfully [TUTORIAL]	Educational Purposes NOT MINE In this tutorial I will show you how to spread your trojans/viruses etc. I will show you many methods, and later you choose which one ....
Similarity selected	Botnet QA!	Just something I compiled quickly. Im also posting my bot setup guide soon. If you want any questions or links added to the Q&A, please ask and Ill add them.
	The COMPLETE beginners guide to hacking	another great guide i found :D Sections: 1) Introduction 2) The hacker manifesto 3) What is hacking? 4) Choosing your path 5) Where should I start? 6) Basic terminology 7) Keylogging...
	[TUT]DDoS Attack - A life lesson	Introduction I know their are a lot more ways to DoS than are shown here, but ill let you figure them out yourself. If you find any mistake in this tutorial please tell me‘What is D́DoS?́

Table 9: Examples of similar threads for class Hacks: threads offering hacking tutorials.

Selection Method	Title	Post
Keyword selected	Blackmailed! How to hack twitter?	Hey, everyone. Im new on this website and I need help. Im trying to hack a twitter account because theyve been harassing me and reporting isnt helping at all.
Similariry selected	Need hacking services	IIm new to this website. Im not a hacker. Would like to hire hacking services to hack email account, Facebook account and if possible iPhone. Drop me a pm if you can do it. Fee is negotiable. Thanks
	Hello hacker members	My name is XXXX and im looking for someone to help me crack a WordPress password from a site that has stolen all our copyrighted content. Weve reported to google but is taking forever. I have the username of the site, just need help to crack the password so i can remove our content. Please message me with details if you can help
	finding a person with his email	Hello guys! I need to find out how I can find a person b́ehindán email! Let me explain please ...
	Hi	hello everyone im new here and i want to learn how to hack an account any account in fact fb twitter even credit card hope you code help me out who knows maybe i can help you in the future right give and take

Table 10: Examples of similar threads for class Services: threads looking for hacking services.

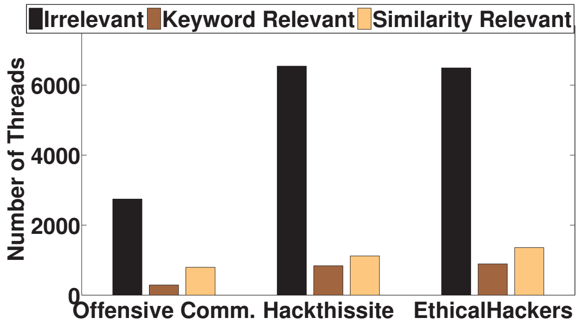


Figure 6: Number of relevant thread in each forums identified by our approach: (a) irrelevant (not selected), (b) selected via keyword matching and (c) selected via similarity.

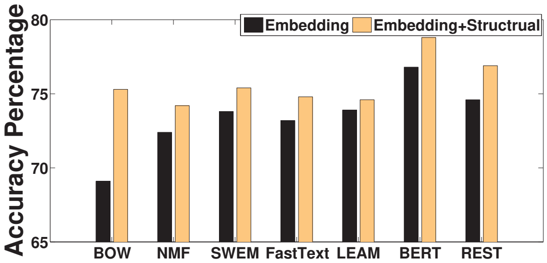


Figure 7: Classification accuracy for two different features sets in 10-fold cross validation in OffensiveCommunity forum.

The contextual features improves classification for all approaches. We briefly discussed contextual features in our classification section. We conduced experiments with and without these features for all six algorithms and we show the results in Figure 7 for OffensiveCommunity. Including the contextual features in our classification improves the accuracy for all approaches (on average by 2.4%). The greatest beneficiary is the Bag-of-Words method whose accuracy improves by roughly 6%.

Datasets	Metrics	BOW	NMF	SWEM	FastText	LEAM	BERT	**REST**
OffensComm.	Accuracy	75.33±0.1	74.31±0.1	75.55±0.21	74.64±0.15	74.88±0.22	**78.58± 0.08**	77.1±0.18
	F1 Score	NA	NA	74.15±0.23	72.5±0.15	72.91±0.18	**78.47±0.01**	75.10±0.14
HackThisSite	Accuracy	65.3±0.41	69.46±0.12	73.27±0.10	69.92±0.08	74.6±0.04	68.99±0.4	**76.8± 0.1**
	F1 Score	NA	70.23±0.13	71.89±0.14	65.81±0.4	71.41±0.09	63.61±0.41	**74.47±0.24**
EthicalHackers	Accuracy	59.74± 0.21	58.3± 0.15	61.3± 0.17	59.73± 0.21	61.80 ±0.13	54.91± 0.32	**63.3± 0.09**
	F1 Score	NA	57.83±0.16	59.6±0.23	59.5±0.13	60.9±0.17	51.78±0.15	**61.7±0.21**

Table 11: Classification: the performance of the five different methods in classifying threads in 10-fold cross validation.

Related Work

We summarize related work group into areas of relevance.

a. Identifying entities of interest in security forums. Recently there have been a few efforts focused on extracting entities of interest in security forums. A very interesting study focuses on the dynamics of the black-market of hacking goods and services and their pricing (Portnoff et al. 2017), which for us is one of the categories of interest. Some other recent efforts focus on identifying malicious IP addresses that are reported in the forum (Gharibshah, Papalexakis, and Faloutsos 2018; Gharibshah et al. 2017), which is relatively different task, as there, the entity of interest has a well-defined format. Another interesting work (Tavabi et al. 2018) uses a word embedding technique focusing identifying vulnerabilities and exploits.

b. Identifying malicious users and events. Several studies focus on identifying key actors and malicious users in security forums by utilizing their social and linguistics behavior. (Li and Chen 2014; Marin, Shakarian, and Shakarian 2018; Abbasi et al. 2014). Another work (Gharibshah et al. 2019; Sapienza et al. 2017) identifies emerging threats by monitoring threads activities and the behavior of malicious users and correlating it with information from security experts on Twitter. Users' behaviors studied (Gharibshah et al. 2020; Aghapour and Farrell 2016) to identify abnormal users in interaction with each others. Another study (Sapienza et al. 2018) detects emerging security concerns by monitoring the keywords used in forums and other online platforms, such as blogs.

c. Analyzing other online forums. Researchers have analyzed a wide range of online forums such as blogs, commenting platforms, reddit etc. Indicatively, we refer to a few recent studies. Google (Zhang, Culbertson, and Paritosh 2017) analyzed *question-answer* type of forums and they also published the large dataset that they collected. Another study focusing on detecting question-answer threads within a discussion forum using linguistic features (Cong et al. 2008).

Despite many common algorithmic approaches, we argue that each type of forum and different focus questions necessitate novel algorithms.

d. NLP, Bag-of-Words, and Word Embedding techniques. Natural Language Processing is a vast field, and even the more recent approaches, such as query transformation and word embedding have benefited from significant numbers of studies (Scells and Zuccon 2018; McCallum and Nigam 1998; Mikolov et al. 2013; Le and Mikolov 2014; Li and Chen 2014; Jin et al. 2016; Wang et al. 2018; Zamani and Croft 2017; Shen et al. 2018; Lee and Seung 1999). Most recently, several methods focus on combining word embedding and deep learning approaches for text classification (Wang et al. 2018; Zamani and Croft 2017; Shen et al. 2018; Devlin et al. 2018).

We now discuss the most relevant previous efforts. These efforts use word embedding representation and they use it for classification for text, but: (a) neither of those focuses on forums, (b) there are some other technical differences with our work. The first work, predictive text embedding (PTE) (Tang, Qu, and Mei 2015), uses a network-based approach, where each thread is described by a network of interconnected entities (title, body, author etc). The second study, LEAM (Wang et al. 2018), uses a word embedding and a Neural Network classifier to create a thread embedding. LEAM argues that it outperforms PTE, and as we show here, we outperform LEAM. Recently Google introduced BERT (Devlin et al. 2018), a deep pre-trained bidirectional transformers for language understanding which uses a pre-trained unsupervised language model on large corpus of data. Although the power of large data set for training is indisputable, at the same time, we saw first hand the need for some customization for each domain space.

Finally, there are some efforts that use Doc2Vec to identify the embedding of a document (equivalently threads in our case). However, these techniques would not work well here due to the small size of the datasets (Le and Mikolov 2014). This technique could be applied in much larger forums, and we will consider it in such a scenario in the future.

Conclusion

There is a wealth of information in security forums, but still, the analysis of security forums is in its infancy, despite several promising recent works.

We propose a novel approach to identify and classify threads of interest based on a multi-step weighted word embedding approach. As we saw, our approach consists of two parts: (a) a similarity-based approach to extract relevant threads reliably, and b) weighted embedding-based classification method to classify threads of interest into user-defined classes. The key novelty of the work is a multi-step weighted embedding approach: we project words, threads and classes in the embedding space and establish relevance and similarity there.

Our work is a first step towards developing an easy-to-use methodology that can harness some of the information in security forums. The easy-of-use stems from the ability of our method to operate with an initial set of bag-of-words, which our system uses to seeds to identify threads that the user is interested in.

Acknowledgments

This work is supported by DHS ST Cyber Security (DDoSD) HSHQDC-14-R-B00017 grant, NSF NeTS 1518878 and UC-NL-CRT LFR 18548554.

References

Abbasi, A.; Li, W.; Benjamin, V.; Hu, S.; and Chen, H. 2014. Descriptive analytics: Examining expert hackers in web forums. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, 56–63.

Aghapour, E., and Farrell, J. A. 2016. Human action prediction for human robot interaction. In *2016 American Control Conference (ACC)*, 5407–5412.

Cong, G.; Wang, L.; Lin, C.-Y.; Song, Y.-I.; and Sun, Y. 2008. Finding question-answer pairs from online forums. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, 467–474. New York, NY, USA: ACM.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Gharibshah, J.; Li, T. C.; Vanrell, M. S.; Castro, A.; Pelechrinis, K.; Papalexakis, E. E.; and Faloutsos, M. 2017. Inferip: Extracting actionable information from security discussion forums. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, 301–304. New York, NY, USA: ACM.

Gharibshah, J.; Gharibshah, Z.; E. Papalexakis, E.; and Faloutsos, M. 2019. An empirical study of malicious threads in security forums. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, 176–182. New York, NY, USA: ACM.

Gharibshah, Z.; Zhu, X.; Hainline, A.; and Conway, M. 2020. Deep Learning for User Interest and Response Prediction in Online Display Advertising. *Data Science and Engineering* 5(1):12–26.

Gharibshah, J.; Papalexakis, E. E.; and Faloutsos, M. 2018. Ripex: Extracting malicious ip addresses from security forums using cross-forum learning. In *Advances in Knowledge Discovery and Data Mining*, 517–529.

Jin, P.; Yue, Z.; Chen, X.; and Xia, Y. 2016. Bag-of-embeddings for text classification. *IJCAI International Joint Conference on Artificial Intelligence* 2016-Janua:2824–2830.

Jivani, A. 2011. A comparative study of stemming algorithms. *Int. J. Comp. Tech. Appl.* 2:1930–1938.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2017. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 427–431. Association for Computational Linguistics.

Kittler, J.; Hatef, M.; Duin, R. P. W.; and Matas, J. 1998. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20(3):226–239.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–1188–II–1196. JMLR.org.

Lee, D. D., and Seung, H. S. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788.

Li, W., and Chen, H. 2014. Identifying top sellers in underground economy using deep learning-based sentiment analysis. In *2014 IEEE Joint Intelligence and Security Informatics Conference*, 64–67.

Marin, E.; Shakarian, J.; and Shakarian, P. 2018. Mining key-hackers on darkweb forums. In *2018 1st International Conference on Data Intelligence and Security (ICDIS)*, 73–80.

McCallum, A., and Nigam, K. 1998. A comparison of event models for naive bayes text classification.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781.

Portnoff, R. S.; Afroz, S.; Durrett, G.; Kummerfeld, J. K.; Berg-Kirkpatrick, T.; McCoy, D.; Levchenko, K.; and Paxson, V. 2017. Tools for Automated Analysis of Cybercriminal Markets. *Proceedings of the 26th International Conference on World Wide Web - WWW '17* 657–666.

Sapienza, A.; Bessi, A.; Damodaran, S.; Shakarian, P.; Lerman, K.; and Ferrara, E. 2017. Early warnings of cyber threats in online discussions. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, 667–674.

Sapienza, A.; Ernala, S. K.; Bessi, A.; Lerman, K.; and Ferrara, E. 2018. Discover: Mining online chatter for emerging cyber threats. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, 983–990.

Scells, H., and Zuccon, G. 2018. Generating better queries for systematic reviews. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '18, 475–484. New York, NY, USA: ACM.

Shen, D.; Wang, G.; Wang, W.; Renqiang Min, M.; Su, Q.; Zhang, Y.; Li, C.; Henao, R.; and Carin, L. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 440–450. Association for Computational Linguistics.

Tang, J.; Qu, M.; and Mei, Q. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 1165–1174. New York, NY, USA: ACM.

Tavabi, N.; Goyal, P.; Almukaynizi, M.; Shakarian, P.; and Lerman, K. 2018. Darkembed: Exploit prediction with neural language models. In *AAAI*, 7849–7854. AAAI Press.

Wang, S.; Chen, Z.; Liu, B.; and Emery, S. 2016. Identifying Search Keywords for Finding Relevant Social Media Posts. *Proceedings of the Thirthieth AAAI Conference on Artificial Intelligence (AAAI-16)* 3052–3058.

Wang, G.; Li, C.; Wang, W.; Zhang, Y.; Shen, D.; Zhang, X.; Henao, R.; and Carin, L. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2321–2331.

Zamani, H., and Croft, W. B. 2017. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, 505–514. New York, NY, USA: ACM.

Zhang, A. X.; Culbertson, B.; and Paritosh, P. 2017. Characterizing Online Discussion Using Coarse Discourse Sequences. *Proceedings of the International Conference on Weblogs and Social Media* 357–366.