Multi-class and Multi-label classification of Darkweb Data

by

Revanth Patil

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved March 2018 by the
Graduate Supervisory Committee:

Paulo Shakarian, Chair
Adam Doupe
Hasan Davulcu

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

In this research, I try to solve multi-class multi-label classification problem, where the goal is to automatically assign one or more labels(tags) to discussion topics seen in darkweb. I observed natural hierarchy in our dataset, and I used different techniques to ensure hierarchical integrity constraint on the predicted tag list. To solve 'class imbalance' and 'scarcity of labeled data' problems, I developed semi-supervised model based on elastic search(ES) document relevance score. I evaluate our models using standard K-fold cross validation method. Ensuring hierarchical integrity constraints improved F1 score by 11.9% over standard supervised learning, while our ES based semi-supervised learning model out-performed other models in terms of precision(78.4%) score while maintaining comparable recall(21%) score.

To my family, roommates, friends and colleagues.

ACKNOWLEDGMENTS

Foremost, I would like to express my sincere gratitude to advisor Dr. Paulo Shakarian for the continuous support of my MS study and research, and for his guidance, patience, and knowledge. His guidance provided right direction for the research and writing of this thesis. I am confident that learnings from this research and the CySIS lab are going to be strong foundations in my career.

I would like to thank Dr. Hasan Davulcu and Dr Adam Doupe for being on my committee and for their support to my research and dissertation. Their cooperation and response facilitated a smooth conduct of my dissertation process.

My sincere gratitudes to Jana Shakarian for all the support and expert knowledge in the field of Darkweb data. I thank Ph.D student Ashkan Aleali, for the stimulating discussions and guidance, and Ph.D student Ericsson Marin, for reviewing and editing all my work. I also thank all the other CySIS lab members for their support.

This research is supported by the Office of Naval Research(ONR) Neptune program, the ASU Global Security Initiative(GSI) and the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory (AFRL). The U.S Government is authorized to reprodcue and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of ONR, IARPA, AFRL or the U.S. Government.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

The ability to traverse the internet with complete anonymity provides online platforms for illegal activities such as credit card fraud, identity theft, leaks of sensitive information and sharing hacking information [Chertoff and Simon (2015)]. One of the most prevalent cyber environments that emerged in the last decade and contributed to the achievement of those criminal tasks are darkweb forums [Abbasi *et al.* (2014)], since they include encryption technology to prevent monitoring and also provide protection from unauthorized users. Table 1.1 provides the details of thirteen broad categories of data seen in web based hidden services on darkweb [Moore and Rid (2016)], having discussion forums as the main supplier platform for the spread of criminal activities.

Considering the enormity of the data in those environments, there is an impending need to go beyond this broad categorization, providing to security researchers a more granular, structural and interdependent classification of the available information. The deepweb data can be classified into different domains such as hacking, whistleblowing, financial-fraud, drugs, counterfeit, books, porn, etc.

In this work, we aim to provide an intelligent system capable of classifying the information extracted from darkweb forums in a hierarchical structure of tags. Technically, we aim to address a multi-class and multi-label classification problem, automatically assigning one or more tags to the topics of the forums analyzed. Such classification would help us to proactively identify cyber threats in very early stages, allowing for a preemptive response by distributing the relevant data to interested groups - government, cybersecurity professionals, financial institutions, etc.

Category	Percentage
Violence	0.3
Arms	0.8
Social	1.2
Hacking	1.8
Illegimtimate pornography	2.3
Nexus	2.3
Extremism	2.7
Unknown	3.0
Other illicit	3.8
Finance	6.3
Drugs	8.1
Other	19.6
None	47.7

Table 1.1: Web Based Hidden Services in February 2016Moore and Rid (2016).

In order to accomplish our goal, we addressed two problems – scarcity of labeled data, and imbalanced classes in the ground truth. Standard classifiers require a large number of labeled training examples to learn accurately. Currently, we rely on the field experts on labeling the training set, but this is a time-consuming process and often doesn't scale considering the fact that our dataset is constantly increasing. The class imbalance problem typically occurs when, in a classification problem, there are many more instance of some classes than the others.

Specific contribution of this paper include, 1) We observed that there was natural

hierarchy structure in our tags. We use multiple approaches to ensure that the tag hierarchy is respected in prediction list. Imposing hierarchical integrity constrains improved precision, recall and f1 scores by 3%, 9% and 12.2% respectively. 2) The implementation and evaluation of semi-supervised learning model based on elastic search document similarity score to solve class imbalance problem. This approach improved precision, recall, f1 scores by 7.3%, 9.9% and 13.4% respectively.

The rest of this document is organized as follows, chapter 2 provides the background information about the darkweb/deepweb primer and the related work, chapter 3 describes technical preliminaries necessary to understand the whole document, chapter 4 describes the different approaches we use in our research, chapter 5 describes the experiments and the results of those experiments, and chapter 6 provides the conclusion of this research.

Chapter 2

BACKGROUND

2.1 Darkweb/Deepweb Primer

The dark web forms a small part of the deep web that is not indexed by web search engines. Darknet websites provide a underground communication and are accessible by special software like Tor(The Onion Router) and I2p Invisible Internet Project. Tor software uses onion routing protocol to provide anonymity for both the service user and the service provider. Tor uses onion routing protocol to provide anonymity for the users. Onion routing is a technique where messages are repeatedly encrypted and then sent through several network nodes, called onion routers. Like someone peeling an onion, each onion router removes a layer of encryption to uncover routing instructions, and sends the message to the next router where the process is repeated. This technique prevents intermediary nodes from knowing the origin, destination and contents of the message 2017a" (2017).

In this research, we are interested in darkweb forums that are anonymously hosted and are identified by the domain '.onion'. Forums are online discussion sites where like-minded community can hold conversations. These anonymous hosting forums discuss on various cyber-security related topics such as worms, botnet, zero-days, hacking tools, backdoor services and etc.

2.2 Related Work

[Nunes *et al.* (2016)] presents a system that helps gather cyber threat intelligence from various social platforms on the darknet and deepnet. They gather data from

both the hacker forum discussions and marketplacess offering products and services. They use data mining and machine learning models to find relevant information from noise in the dadta collected from the online platforms. They use binary classifier on the collected data to find whether the products in marketplaces and topics on forum post contain communication relevant to malicious hacking.

[Samtani *et al.* (2016)] describes a prototype of the malware analysis portal. This portal collects and analyzes malicious assets directly from the online hacker communities. They collect data from only 14 malware portals and use supervised learning technique to categorize the conversations into 3 classes - 'web', 'system', and 'network'. They use hand-labelled data as ground truth. This research considers limited number of malware sites. However, they consider limited number of sites and do not address problem of scarcity of labeled data when we try to scale by increasing the number of sites.

[Marin *et al.* (2016)] describes the systems involved to collect data from the market places and present effective unsupervised learning techniques to categorize the data. They collect data from 17 marketplaces and use K-means algorithm to categorize the products. Again, they rely on field experts on labeling the training set which is a time consuming process. They do not address the problem of scarcity of labeled data.

Chapter 3

TECHNICAL PRELIMINARIES

3.1 Problem Statement

Classification of darkweb-deepweb forum discussions is a multiclass-multilabel classification problem, where the goal is to automatically assign one or more labels to each discussion topics. Formally, let X be a set of forum topics, $y = \{0, 1\}^k$, let k be possible tags and D be an unknown distribution on the product space $X \times Y$. Each element (x,y) in this space is composed of an instance x and a label vector y, which is a vector of indicators $y = [y_1, ..., y_k]$ that specifies classes associated with x. A classifier is a function h: $X \rightarrow Y$, that maps instance x to a label vector y = h(x) [Dekel and Shamir (2010)]. For example, the sample forum topic 'warning enigma badlock - upcoming smb/cifs and samba vulnerability' belongs to three classes - vuln, smb and file-share.

3.2 Description of the Dataset

We have crawled data from 283 cyber security related online forums. From the collected html pages we parse for important fields such as topic title, posts, user name, title posted date, user ratings, number of replies, etc. Our dataset includes discussions related to 486996 different topics with 4188345 posts and 748698 users participating in those topics.

The ground truth to build machine learning model is hand labelled by field experts. We have 2046 labelled topics as training set and they belong to 226 unique tags. Figure 3.1 shows the tag frequency in the training set. Some tags have more than

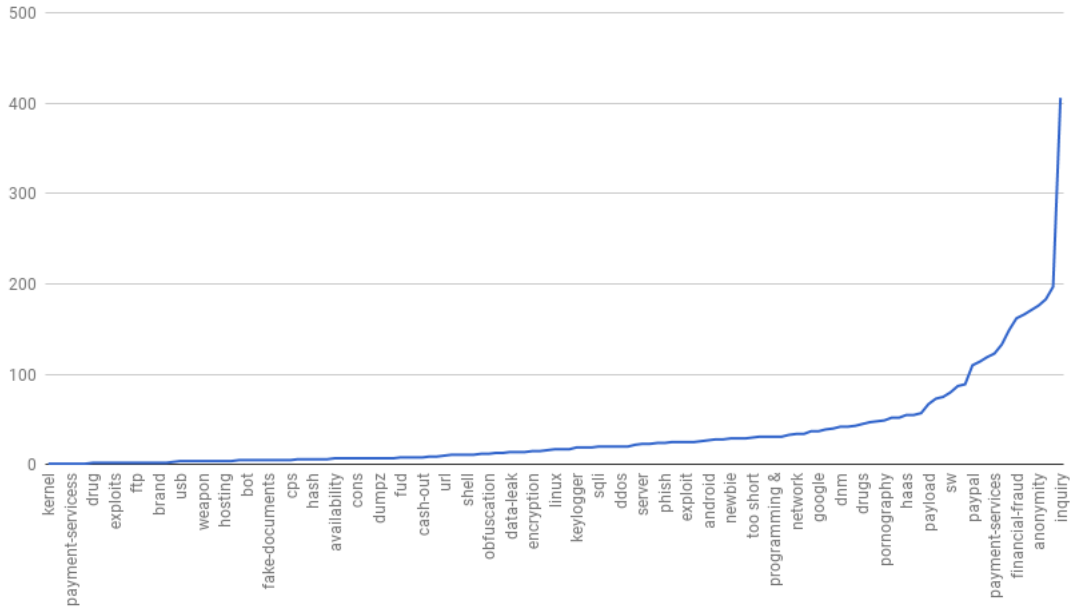

Figure 3.1: Frequency of Tags in the Training Set

400 topics in the training set while some have just one topic. Table 3.1 provides the sample ground truth topics and the tags associated.

3.3 Feature Extraction

We perform textual feature extraction on topic title using Doc2vec vectorization technique. Doc2vec is an extension of word2vec that relies on the idea that words which appear in the similar context have the same semantic meaning [Harris (1954)]. Word2vec is computationally-efficient predictive model that uses shallow 2 layer neural network for learning word embeddings. We use Distributed Memory Model of Paragraph Vectors(PV-DM) Doc2vec architecture which is based on Continuous Bag Of Words(CBOW) word2vec architecture. PV-DM architecture is faster to train and has better accuracy for frequent words [Rong (2014)].

Word2vec CBOW. In this architecure the model predicts the current word from

Topic titles	Tags
enigma badlock - upcoming smb cifs and samba vulnerability.	vuln, smb, file-share
u nessus vulnerability scan	vuln, pentest, network
warning enigma badlock - upcoming smb/cifs and samba vulnerability.	vuln, smb, file-share
question about school-sponsored wifi?	wifi, inquiry
technical question about wifi cards	wifi, inquiry
(b) paypal unchecked accounts (s) btc account	financial-fraud, darkweb product, paypal, payment-services, market
there are phishing attempts reported!	phishing
[u'[payload] sharing some payloads!']	malware, payload, hack

Table 3.1: Sample Ground Truth

a sliding window of context words. Figure 3.2 shows the simple CBOW architecture with just one word considered per context. Consider the settings with V as vocubulary size and N as hidden layer size. The input is a one-hot encoded vector i.e for a given input context word only one of the V units, {x1, x2,...., xv}, is 1. Weights between input and hidden layer is represented by W which is of dimensions $V \times N$. The activation function for hidden layer is linear. Formally, row i of W and for a given context word w,

$$h = \mathbf{W}^{\intercal} \times x := \mathbf{v}_w^{\intercal} \tag{3.1}$$

v_w is the the vector representation of the input word w. From the hidden layer to the output layer, there is different weight matrix W' which is of dimensions $N \times V$. We

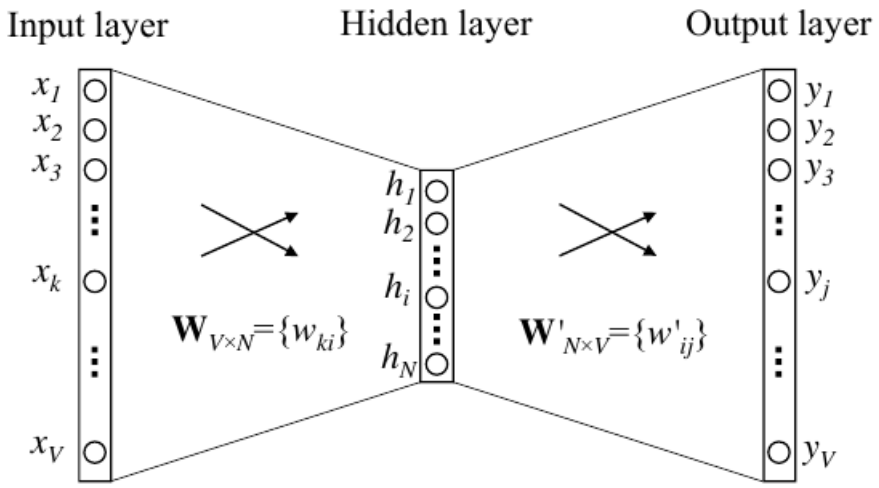

Figure 3.2: A Simple Word2vec Cbow Model in with Only One in the Context. [Rong (2014)]

compute a score u_j for each word in the vocabulary,

$$u_j = \mathbf{v}'^{\mathsf{T}}_{w_j}\mathbf{h} \tag{3.2}$$

where $\mathbf{v}'^{\mathsf{T}}_{w_j}$ is the j-th column of the matrix $\mathbf{W}'$. A log-linear softmax function is applied to obtain posterior distribution of words. Formally, y_j is the output of jth unit in the output layer.

$$y_j = \frac{exp(u_j)}{\sum_{j'=1}^{V} exp(u_{j\prime})} \tag{3.3}$$

We update the weights($\mathbf{W}$ and $\mathbf{W}'$) using logarithmic loss function with backpropogation method. After updating the weights, we use $\mathbf{v}_w$ for word w from equation 3.1 as the 'input vector' word embedding [Rong (2014)].

Doc2vec PV-DM. Paragraph vector is an supervised framework that learns continuous distributed vector representations for documents instead of words. The text in the documents can be of variable-length. As shown in Figure 3.4, every paragraph is mapped to a unique vector, represented by a column matrix D and every word is also mapped to a unique vector, represented by a column in matrix W. The paragraph

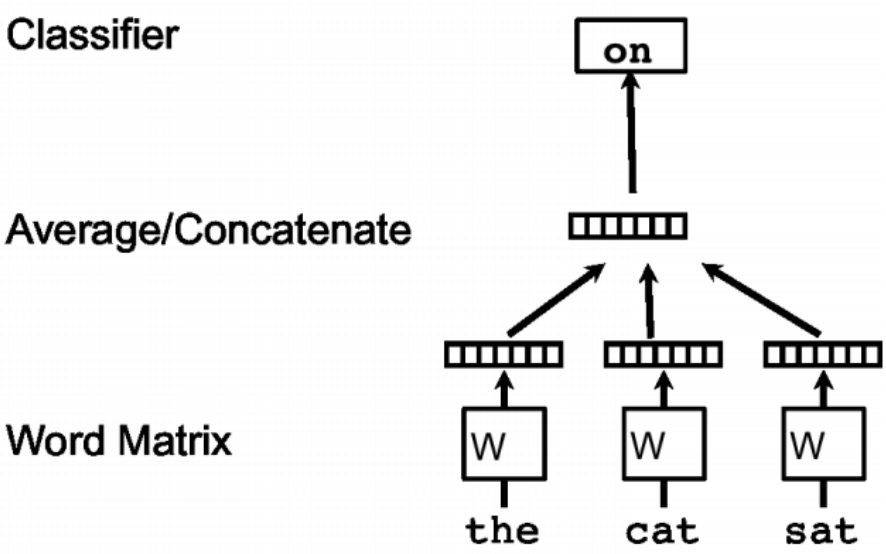

Figure 3.3: Word2vec CBOW Framework for Learning Word Vectors. Context of Three Words Is Used to Predict the next Word. Le and Mikolov (2014)

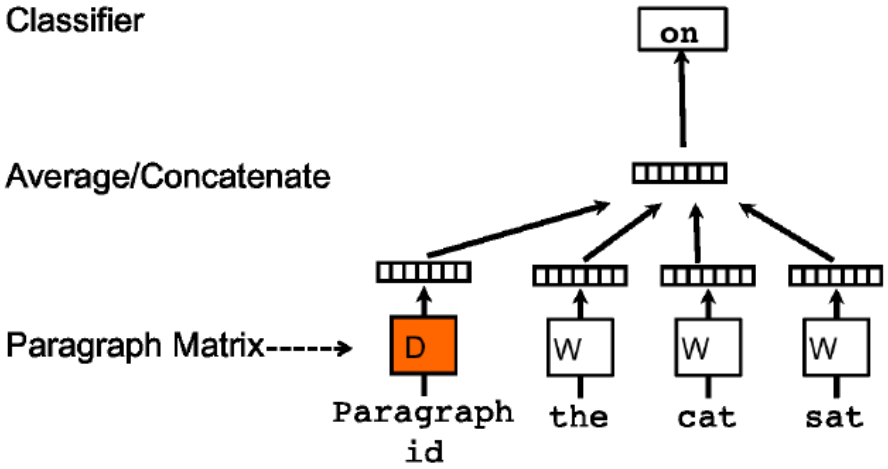

Figure 3.4: Doc2vec Pv-dm Framework for Learning Vectors. Context of Three Words, and Paragraph Vectors Is Used to Predict the next Word.[Le and Mikolov (2014)]

vector and word vectors are averaged or concatenated to predict the next word in a context. The only change in this model compared to word2vec framework is addition of paragraph vector along with the word vectors [Le and Mikolov (2014)].

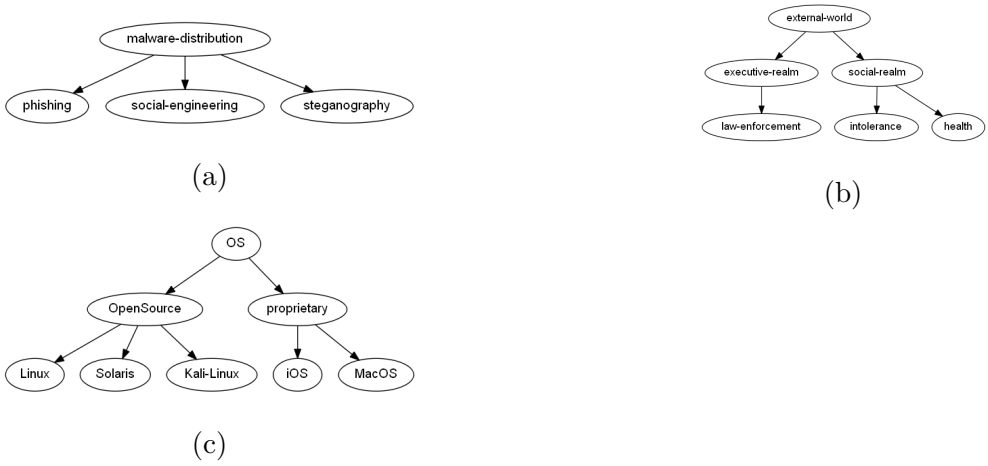

Figure 3.5: Sample of Tag-Hierarchical Structure

3.4 Tags and the Hierarchy Among Tags

In the ground truth, each topic is associated with multiple tags. We observed that there is a natural hierarchical structure in our tags. There are group tags (parent tags) that can be seen as the "broader term" for its set of specific tags(child tags). Nesting specific tags under group tags creates a hierarchy of tags. Figure 3.5 shows an example of the hierarchical structure in the training data set. Here, external-world tag is the more generic parent tag for social-realm and health tag. Similarly social-realm and health tags are children tags to external-world tag.

Formally, hierarchy constraint is given by a set $\mathcal{H} = \{H_1, H_2, ...H_n\}$ where H_i is a tree. For all the tags T, a set of tags S ($S \subseteq T$) are said to be consistent if every tag t($t \subset S$) satisfies either of the two conditions:

1. p_t = t i.e t is a root node

2. p_t is consistent wrt $\mathcal{H}$

where p_t tag is parent of tag t.

Chapter 4

APPROACH

4.1 Baseline Approach

For the baseline approach, we use SVM, decision tree and random forest classifiers to perform multi-class and multi-label classification.

Decision trees are a non-parametric supervised learning method used for classification and regression. They learn the simple decision rules from the data features to predict the target label. Decision tree method is a recursive partitioning algorithm that is widely used for classification problems. This algorithm aims to maximize the information gain at every step.

Support Vector Machine is a supervised learning method that finds a hyper-plane that differentiate two classes in a multi-dimensional feature space. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using kernel function, implicitly mapping their inputs into high-dimensional feature spaces [Wikipedia (2017)]. We used SVM radial basis function(rbf) kernel with "one-vs-the-rest" strategy to perform multi-class multi-label classification [Pedregosa *et al.* (2011)].

Random Forest classifier is an ensemble algorithm that creates different decision trees by randomly selecting subset of training set and the subset of features. The number of features that are searched at each split point is specified as the parameter. To reduce memory consumption, the total number of trees and size of the trees should be controlled. It then aggregates the votes from different decision trees to decide the final class of the test object [Breiman and Cutler (2007)].

4.2 Overview of Problem Specific Challenges

Ensuring Consistency in Results Based on Hierarchy In our ground truth, we observed that the tags had natural hierarchy. As our original ground truth is hand labelled, we cannot expect all of the ground truth entries respect the tag hierarchy due to human error. Therefore, we developed a simple automation script that establishes the hierarchy in each of our topic titles in the ground truth. Now with hierarchy established in the tags, our classifier provided detailed tag prediction for all the documents.

Ideally the classifier should predict the hierarchical representation of tags i.e for a predicted child tag every parent tag should also be predicted. However we observed that for some tags returned by the classifier function, the corresponding parent tags is not returned. This happens when either the tag is wrongly predicted (false positive) or when some of the correct tags(false negatives) are not predicted.

Class Imbalance The class imbalance problem typically occurs when, in a classification problem, there are many more instances of some classes than the others [Chawla *et al.* (2002)]. In such cases, classifiers tend to be overwhelmed by the majority classes and ignore the minority classes. These minority classes usually have lesser precision and recall scores in the baseline approach.

Semi-supervised learning techniques are usually used to over-sample and under-sample the minority or majority class for adjusting the class distribution of a dataset [Chawla *et al.* (2002)].

4.3 Leveraging Tag Hierarchy

We use three approaches - adding parent tag, removing child tag, combination of adding parent and removing child tag - to ensure tag hierarchy is maintained in the

prediction list.

Adding Parent Tag. If a child tag is predicted by the classifier and if the corresponding parent tags in the hierarchy are not predicted then we add all the parent tags to the prediction list. In Figure 3.5(b), with the tag hierarchy – external-world, social-realm, health – if health tag is predicted without external-world and social-realm, then we add external-world and social-realm tags to the prediction list.

Removing Child Tag. In this approach, we remove all predicted child tags that does not have all the hierarchical parent tags. In Figure 3.5(b), with the tag hierarchy - external-world, social-realm, health - if health tag is predicted without external-world and social-realm, then we remove health tag from the prediction list.

Algorithm 1 Combination of Adding Parent Tags and Removing Child Tags (**T**)

Input: Prediction list of tags(T) for each document.

Output: Prediction list of tags that respect the tag hierarchy for each document.

$\beta \leftarrow$ remove child threshold value

$\alpha \leftarrow$ add parent threshold value

for `t:= 1 to T` **do**

    $p(t) \leftarrow$ prediction probability of tag t.

    if $p(t) < \beta$ **then** *remove all parent tags of tag t from prediction list*

for `t:= 1 to T` **do**

    $p(t) \leftarrow$ prediction probability of tag t.

    if $p(t) > \alpha$ **then** *add all parent tags of tag t from prediction list*

return updated prediction list

Combination Of Adding Parent Tag And Removing Child Tag (CAR). In this approach, we want to regulate the way we add parent tags and remove child tags based on the probability of the predicted tag. We use threshold values - 'add parent

threshold'(α) and 'remove child threshold'(β) - to decide whether to add parent tag or remove child tag. The procedure for this approach is described in Algorithm 1.

4.4 Dealing with Class Imbalance

The discussions in the dark forums are based on various topics which are unequally distributed. Therefore, we observe class imbalance in our ground truth. In our dataset, the ratio of the minority to the majority classes is 1:400 which is quiet drastic. To deal with class imbalance problem, we use semi-supervised learning methods. Semi-supervised learning is a class of supervised learning techniques that also make use of additional unlabeled data to better capture the shape of the underlying data distribution and generalize better to new samples [Pedregosa *et al.* (2011)].

4.4.1 SMOTE

We use Synthetic Minority Over-Sampling Technique(SMOTE) as a baseline approach to solve the class imbalance problem. Oversampling is to correct for a bias in the original data set. In SMOTE, the minority class is over-sampled by taking each minority class sample and introducing synthetic examples along the line segments joining all of the k minority class nearest neighbors. Depending upon the amount of over-sampling required, neighbors from the k nearest neighbors are randomly chosen. Our implementation currently uses five nearest neighbors. Synthetic samples are generated in the following way: 1) Take the difference between the feature vector under consideration and its nearest neighbor 2) Multiply this difference by a random number between zero and one 3) Add it to the feature vector under consideration [Chawla *et al.* (2002)].

4.4.2 Leveraging Document Similarity Through Elastic Search

One way to solve class imbalance problem is by increasing the ground truth size, but that is difficult as we are dealing with human-labeled data-set. Therefore, we use semi-supervised method to increase the sample size of the minority classes. Semi-supervised learning methods make use of unlabeled data for training – typically a small amount of labeled data with a large amount of unlabeled data [Rajaraman (2011)]. For every sample in the minority class, we find the top similar documents from the database using the elastic search similarity score and add them to the training data.

Elastic search is a high-performance, full-featured text search engine library. It's ranking function is applied to determine how relevant a retrieved document is to a given query. The ranking function is based on a combination of Vector Space Model and Boolean model of Information Retrieval. The main idea behind this approach is more times a query term appears in a document relative to the number of times the term appears in the whole collection, the more relevant that document will be to the query [Sparck Jones (1972)]. The elastic search uses BM25 as the default ranking function to provide the relevance score to a given search query. It is not a single function, but a family of TF-IDF like retrieval functions [Okapi-BM25 (2017)]. One of the most commonly used scoring function is described below. Given a query Q, containing keywords q1,..,qn, the BM25 score of a document D is,

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i).\frac{f(q_i,D).(k_1+1)}{f(q_i,D+k_1.(1-b+b.\frac{|D|}{avgdl})} \tag{4.1}$$

where $f(q_i, D)$ is q_i's term frequency in document D, $|D|$ is the length of the document D in words and avgdl is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$ [Sanderson (2010)]. IDF(q_i) is

the Inverse document frequency weight of the query term q_i. It is usually computed as:

$$IDF(q_i) = log\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \tag{4.2}$$

where N is the total number of documents in the collection, and n(q_i) is the number of documents containing q_i. BM25 ranking function is considered as state-of-the-art in the Information Retrieval community [Pérez-Iglesias *et al.* (2009)]. The semi-supervised relevance score based is described by algorithm 2.

Algorithm 2 Semi-Supervised Learning with Document Similarity (**T, Tr(T_i, T_p), F**)

Input: Tags in training set T, Training set Tr with tuple (document titles T_i, tag predictions T_p, Frequency of tags in training set F.

Output: Increased minority class samples from untrained dataset, Ims.

$T_{min} \leftarrow$ array of minority classes
for `t in T` **do**
    if $F(t) < 150$ **then**
        $T_{min} \leftarrow$ *add tag t*
for `ti in Ti` **do**
    $tp \leftarrow T_p(ti)$
    if tp in T_{min} **then**
        $sd \leftarrow$ use relevance score and find top similar documents from untrained data
        $Ims \leftarrow$ add tuple(sd, tp)
$Ims \leftarrow$ add Tr
return Ims

Chapter 5

EXPERIMENTS

5.1 Description of Systems Used to Conduct Experiments

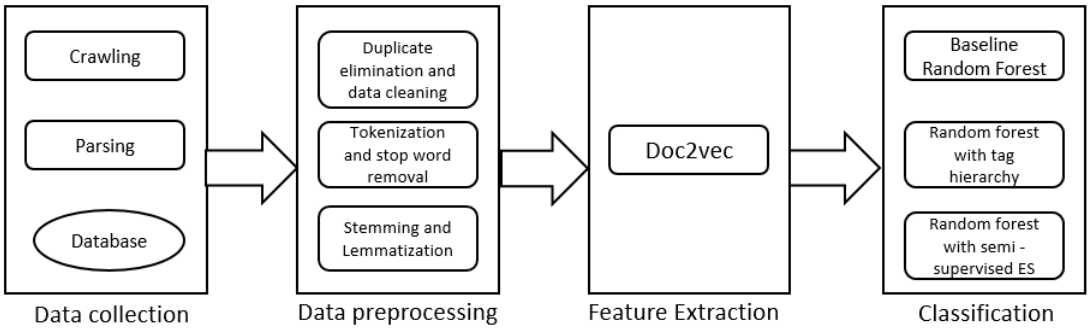

Figure 5.1: System Overview

Fig 5.1 shows the overview of the system that we have developed. The system consists of four main modules - Data collection, data-preprocessing, feature extraction, and classification. We use spider programs to discover darkweb forums and human analysts filter these forums that are relevant to cyber security domain. On these forum sites, we use tor based specialized crawlers to traverse the website and retrieve HTML pages. Each site has a parser program that extracts important information, such as topic title, topic content, post content, topic author, post author, author reputation, posted date for each title and posts, etc., from HTML pages. We store all this information on database. Expert analysts use a fraction of this data to prepare ground truth by labelling tags to topic titles. This ground truth forms the basis for the next 3 modules to develop machine learning model.

Figure 5.1 shows the data preprocessing steps - eliminating duplicates in the

ground truth, eliminating non-english content in both the ground truth and the test data, tokenizing the data, removing stop words, and applying stemming and lemmatization processes on the data.

Tokenization. Given a sequence of characters, tokenization is the task of chopping it up into pieces, called tokens, at the same time throwing away certain characters, such as punctuation and non-ascii characters [Manning (2015)]. In the meanwhile, we keep the useful characters, such as currency symbols, question mark etc that add semantic meaning to the features.

Stop Word Removal. Extremely common words that appear in most of the documents are of little value in feature extraction. Therefore, they are removed from the topic titles.

Stemming and Lemmatization. Stemming is the process for reducing inflected or derived words to their stem, base or root form. Lemmatization, is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization and stemming are closely related. The small difference is that lemmatization uses the context, whereas stemming operates on a single word [Manning (2015)]. For example, stemming process reduces the derived words - caresses, ponies and cats to caress, poni, and cat respectively. On the other hand, lemmatization reduces the derived word 'saw' to 'see' or 'saw' depending on whether the use of token was a verb or a noun.

We have explained about feature extraction and classifier modules in Section 3.3, and Sections 4 respectively.

5.2 Description of Software

Table 5.1 provides the summary of the software and version description for each of the module used. For data-preprocessing, we used module textblob with version

Module	**Software**
Data preprocessing	textblob '0.12.0'
Tokenization, Stemming and Lemmatization	natural languange toolkit(nltk) '3.2.4'
Doc2vec	Gensim '2.3.0'
SVM, the decision tree and random forest classifications	scikit-learn '0.19.0'
SMOTE	imblearn '0.3.0'

Table 5.1: Software Description of the Modules.

'0.12.0' for part-of-speech tagging and natural language toolkit (nltk) library with version '3.2.4' for tokenization, stemming and lemmatization. For vectorization, we used scikit-learn, sklearn version '0.19.0' for Tf-IDF feature extraction and gensim module version '2.3.0' for doc2vec vectorization. For supervised classification, we used svm, decision tree and random forest classifiers built in sklearn '0.19.0'. and for SMOTE technique, we used imblearn module version '0.3.0'. All the above modules are developed using python '2.7.13' programming language. To find similar documents using ES, we used elastic search version '5.3.2' with backend lucene search engine version '6.4.2'. We implemented all the systems in linux based Ubuntu 16.04 LTS operating system.

5.3 Description of Metrics

We evaluate the performance of our classifier models based on four metrics - precision, recall, F1 score, and percentage of documents titles with at least one correct prediction tag. Precision, recall and F1 scores are calculated for each of the tags

and for all the tags put together. Table 5.1 lists the formal definitions of precision, recall and F1 scores. Precision score is defined as the fraction of correctly predicted document titles from all the predicted document titles. Recall score is defined as the fraction of correctly predicted document titles from the total number of document titles. F1 measure is the harmonic mean of precision and recall score. We calculate these precision, recall and F1 scores for individual tags and for the cumulative of all the tags.

Metric	**Formula**
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
F1	$2*\frac{precision*recall}{precision+recall}$

Table 5.2: EVALUATION METRICS. True Positives(TP) , False Positives(FP), False Negatives(FN).

5.4 Results

In this section, we describe the results of different experiments; baseline classifiers, classifiers with tag hierarchy considered, and classifiers with semi supervised learning technique.

5.4.1 Results of Baseline Classifier

In all our experiments, we use training set with topic titles to perform 10-fold cross validation to validate our models. With 10-fold cross validation, the ground truth

Experiment	Precision %	Recall %	F1 score %	Percentage of documents with at least one correct prediction
Baseline Decision trees	58.3	6.41	11.54	24
Baseline SVM	67.2	8.51	15.1	27.9
Baseline Random Forest	71.1	11.3	19.7	31.37
Add parent	71.9	20.2	31.5	46.07
Remove Child	74	11.4	19.47	43.6
SMOTE	40.3	30.1	34.4	61.76
Semi-supervised learning based on Elastic Search relevance score	78.4	21.2	33.1	54.9

Table 5.3: Experimental Results

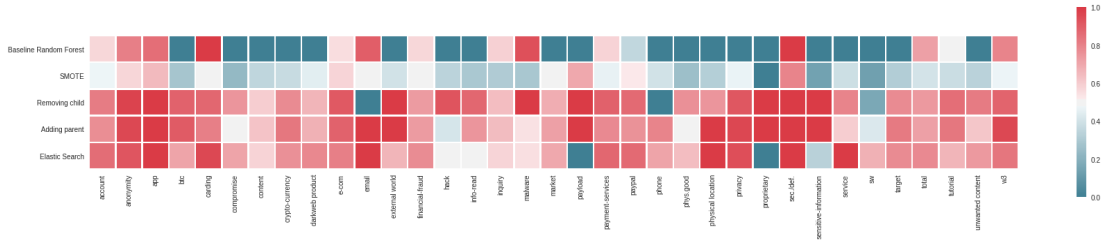

Figure 5.2: Tag-wise Precision Scores for Each Experiment.

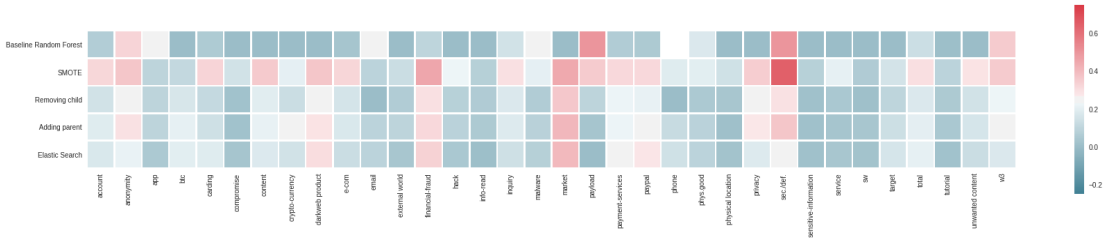

Figure 5.3: Tag-wise Recall Scores for Each Experiment.

data is randomly partitioned into 10 equal subsample buckets. Out of 10 buckets, one bucket is used for testing the model and the rest of the ground truth is used for training the model. Each of the 10 bucket subsample is used for testing the model in cross validation. The aggregate of the accuracy scores of k scores is used as the

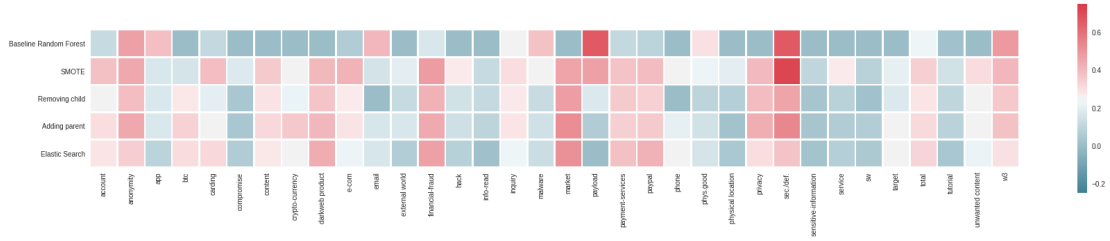

Figure 5.4: Tag-wise F1 Scores for Each Experiment.

final accuracy score. As our ground truth data contains 2,046 titles, each of the 10 buckets consists of 204 samples for testing and 1842 for training the model.

We used Decision Tree, SVM and Random Forest to perform multi-class and multi-label classification. For the decision tree classifier, we used pruning to avoid overfitting, by setting the minimum number of samples required at a leaf node to 5%. For SVM kernel, we used 'one-vs-the-rest' strategy with radial basis function(rbf) kernel. For the random forest, we tuned the number of estimators of the tree parameter and found optimal results when number of estimators is 200. In the above mentioned setting, the Random Forest performed best with the 71.1% precision score and 11.3% recall score. Therefore, we used Random Forest as the default classifier for all the experiments.

5.4.2 Change in Precision and Recall Scores for Each Tag Vs Frequency of Each Tag in Training Set for Es Based Semi-supervised Technique

We used 3 different approaches - adding Parent tag, removing child tag, combination of adding parent and removing child tag - to preserve the tag hierarchy in all our tag predictions.

Adding Parent Tags

In every iteration of K-fold cross validation, if a child tag is predicted by the classifier and if the corresponding parent tags in the hierarchy are not predicted then we add all the parent tags to the prediction list. Adding parent tags decrease the false negatives and increase true positives. Hence, the recall score would increase, but if the child tag is wrongly predicted then the error is propagated and increases the false positives. Thereby, decreasing the precision score. Our observation reveals that there is slight decrease in the precision score, but the recall score improved significantly from 11.3% to 20.2%.

Removing Child Tags

In every iteration of K-fold cross validation, we remove all predicted child tags that does not have all the hierarchical parent tags. This experiment reduces the false positives, and thereby increasing the precision. Our observation from this experiment reveal that our precision increased from 71.1% to 74%, and the recall score did not change from the baseline.

Combination of Adding Parent Tags and Removing Child Tags (CAR)

In this approach, we wanted to regulate the way we add parent tags and remove child tags based on the probability of the predicted tag. We used add parent threshold(α) or remove child threshold values(β) to decide on whether to add or remove a particular tag respectively.

Removing children reduces the false positives, increasing the precision score, while adding parents reduces the false negatives, increasing the recall. We repeat the above experiment for different remove child threshold(β) and add parent threshold(α). As shown in the figure 5.5, with increase in add parent threshold(α) value, the recall

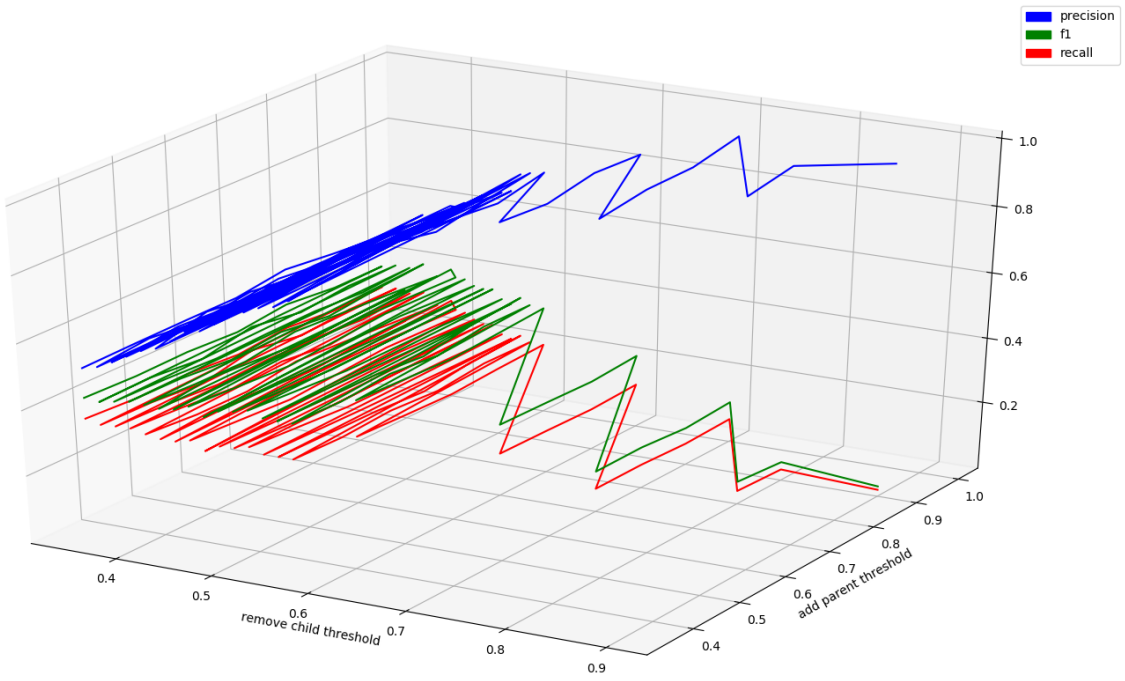

Figure 5.5: Combination of Adding Parent Tags and Removing Child Tags with Threshold.

score increased and precision score decreased. Conversely, with increase in remove child threshold value(β), the precision increased and recall decreased. Our aim was to find a optimum increase in F1 score with out drastically affecting the precision score by tuning the (α) and (β) values. The best F1 score was achieved when we set $\beta = 0.5$ and $\alpha = 0.9$. The corresponding F1 score is 31.9%, precision score is 73.43% and recall score is 20.4%.

5.4.3 Results for Experiments on Class Imbalance

We used semi-supervised learning methods to solve class imbalance problem by making use of unlabeled data for training. For both SMOTE and elastic search, we use the ground truth with tag hierarchy and preserve the tag hierarchy in prediction

list by using 'combination of adding parent tags and removing child tags' method with $\alpha = 0.5$ and $\beta = 0.9$.

SMOTE Semi-Supervised Technique

In SMOTE, synthetic samples are introduced along the line segments joining all of the k minority class nearest neighbors. In our experiment, we used 5 as the number of nearest neighbours to construct synthetic samples. The minority classes targeted will be over-sampled to achieve an equal number of sample with all the classes. With the introduction of synthetic classes, we observed reduction in false negatives and increase in false positives. Therefore, recall for SMOTE increased from 11.3% to 30.1% and precision decreased from 71.1% to 40%. The overall F1 score increased from 19.7% to 34.4%.

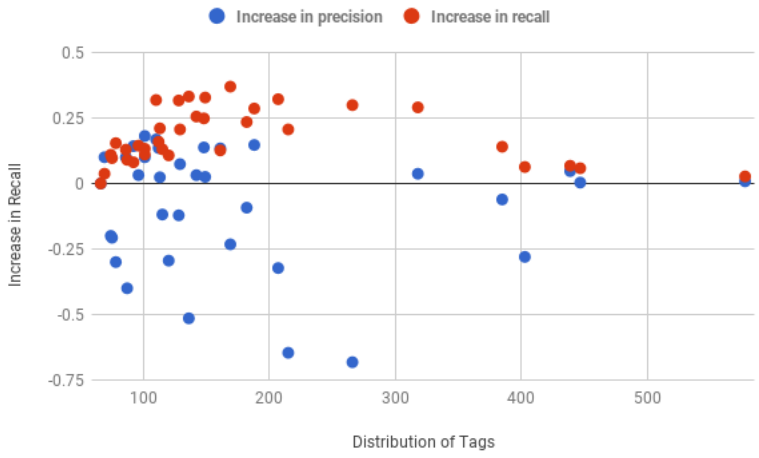

Figure 5.6: Change in Precision and Recall Scores for Each Tag Vs Frequency of Each Tag in Training Set for Smote Technique

As shown in Figure 5.6, SMOTE technique was most effective in increasing the recall score for the tags that had frequency between 95 to 300 in the ground truth. For this range, the average increase in recall for a tag is 23.35%. But this technique

was not very useful for the tags that had frequency fewer than 95 or for the tags that had frequency higher than 300. For these ranges the average increase in recall was just 8.33%. As shown in the Table 5.3, percentage of documents with at least one correct prediction increased from 31.37% to 61.76% with SMOTE semi-supervised learning.

Elastic Search Based Semi-Supervised Technique

In this experiment, we extend the training samples of the each minority class by using document similarity score.

Using the above procedure, we extended the training samples of each of the minority classes to have 500 documents titles each. With this approach, precision increased from 71.1% to 78.4%, recall increased from 11.3% to 21.1%, and F1 increased from 19.7% to 33.1%. Compared to SMOTE, precision increased from 40% to 78.4% but there is a dip in recall from 30.1% to 21.1%. When augmenting synthetic samples with the training set using SMOTE technique, we observed that false negatives decreased and false positives increased with the introduction of synthetic samples. However with ES based technique, the false negatives decreased and true positives increased.

As shown in the figure 5.7, precision and recall increased for most of the tags. This technique was most effective for the tags that had frequency between 86 to 318. For this range the average increase in precision was 52% and average increase in recall was 12%. For the tags having frequency below 86, the increase in precision and recall was just 2.86% and 5.6% respectively. As shown in the Table 5.3, percentage of documents with at least one correct prediction increased from 31.37% to 54.9% with this approach.

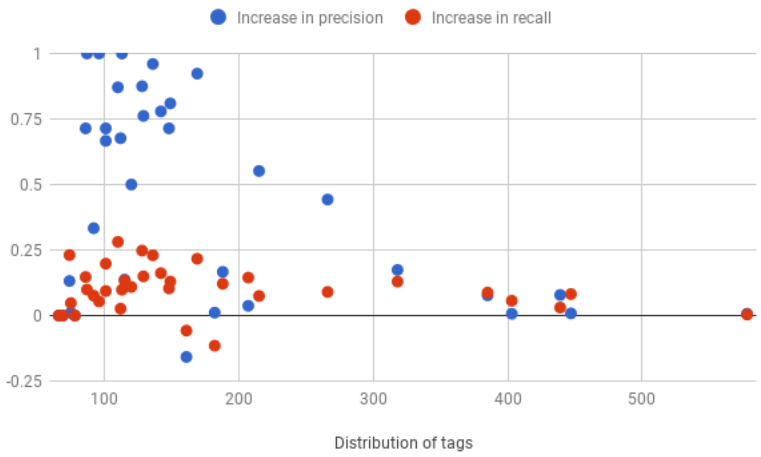

Figure 5.7: Change in Precision and Recall Scores for Each Tag Vs Frequency of Each Tag in Training Set for Es Based Semi-Supervised Technique

Chapter 6

CONCLUSION

In this research, we implement a system to categorize the data from darknet and deepnet into multiple domains. We use two novel approaches — enforcing hierarchical integrity constraint on the predicted tags and semi-supervised learning technique using elastic search(ES) based document similarity – to solve multi-class multi-label classification problem. Enforcing the hierarchical integrity constraint provides detailed tag predictions and improves the performance of the baseline classifiers. Also, we address class imbalance problem and scarcity of labeled data problem by using semi-supervised learning technique using elastic search(ES) based document similarity score. This method outperforms all other standard supervised and semi-supervised methods.

REFERENCES

2017a", T., ""tor project overview"", URL "https://www.torproject.org/about/overview.html.en accessed 27-December-2018]" ("2017").

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

Breiman, L. and A. Cutler, "Random forests-classification description", Department of Statistics, Berkeley **2** (2007).

Chawla, N. V., K. W. Bowyer, L. O. Hall and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique", Journal of artificial intelligence research **16**, 321–357 (2002).

Chertoff, M. and T. Simon, "The impact of the dark web on internet governance and cyber security", (2015).

Dekel, O. and O. Shamir, "Multiclass-multilabel classification with more classes than examples", in "Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics", pp. 137–144 (2010).

Harris, Z. S., "Distributional structure", Word **10**, 2-3, 146–162 (1954).

Le, Q. and T. Mikolov, "Distributed representations of sentences and documents", in "Proceedings of the 31st International Conference on Machine Learning (ICML-14)", pp. 1188–1196 (2014).

Manning, R. R., "Introduction", in "Retrieving the Radical Tillich", pp. 1–19 (Springer, 2015).

Marin, E., A. Diab and P. Shakarian, "Product offerings in malicious hacker markets", in "Intelligence and Security Informatics (ISI), 2016 IEEE Conference on", pp. 187–189 (IEEE, 2016).

Moore, D. and T. Rid, "Cryptopolitik and the darknet", Survival **58**, 1, 7–38 (2016).

Nunes, E., A. Diab, A. Gunn, E. Marin, V. Mishra, V. Paliath, J. Robertson, J. Shakarian, A. Thart and P. Shakarian, "Darknet and deepnet mining for proactive cybersecurity threat intelligence", in "Intelligence and Security Informatics (ISI), 2016 IEEE Conference on", pp. 7–12 (IEEE, 2016).

Okapi-BM25, "Okapi-bm25 ranking function Wikipedia, the free encyclopedia", URL https://en.wikipedia.org/wiki/Okapi-BM25, [Online; accessed 12-December-2017] (2017).

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, "Scikit-learn: Machine learning in Python", Journal of Machine Learning Research **12**, 2825–2830 (2011).

Pérez-Iglesias, J., J. R. Pérez-Agüera, V. Fresno and Y. Z. Feinstein, "Integrating the probabilistic models bm25/bm25f into lucene", arXiv preprint arXiv:0911.5046 (2009).

Rajaraman, A. U., "Jd (2011)." data mining", Mining of Massive Datasets pp. 1–17 (2011).

Rong, X., "word2vec parameter learning explained", arXiv preprint arXiv:1411.2738 (2014).

Samtani, S., K. Chinn, C. Larson and H. Chen, "Azsecure hacker assets portal: Cyber threat intelligence and malware analysis", in "Intelligence and Security Informatics (ISI), 2016 IEEE Conference on", pp. 19–24 (IEEE, 2016).

Sanderson, M., "Christopher d. manning, prabhakar raghavan, hinrich schütze, introduction to information retrieval, cambridge university press 2009.", Natural Language Engineering **16**, 1, 232–234 (2010).

Sparck Jones, K., "A statistical interpretation of term specificity and its application in retrieval", Journal of documentation **28**, 1, 11–21 (1972).

Wikipedia, "Random forest — wikipedia, the free encyclopedia", URL `https://en.wikipedia.org/w/index.php?title=Random`$_{f}orestoldid = 822509370, [Online; accessed29 - December - 2018](2017).$