

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Automated Analysis of Cyber Attack Reports

Author:

Livia Giulia Laurence Fries

Supervisor:

Prof Emil Lupu

Submitted in partial fulfillment of the requirements for the MSc degree in
Computing of Imperial College London

September 2022

Abstract

With mounting evidence of our vulnerability to the cyber threat, scholars have witnessed an “information explosion of Cyber Threat Intelligence (CTI) reports” (1) published by companies and governments alike. Included in these, Tactics, Techniques and Procedures (TTPs) are particularly valuable in describing behavioral patterns from the view point of the attacker. Unfortunately, TTPs are often hidden in unstructured text which requires the manual intervention of the cybersecurity professional (2; 3). The time-consuming and expensive nature of TTP information extraction is therefore at odds with the volatility of the cyber threat environment (2). To fill this gap, this thesis aims to automate the analysis of CTI reports. Several multi-class, multi-label classifiers and Natural Language processing techniques are tested to develop a system capable of automating the extraction of TTPs from CTI reports according to MITRE’s ATT&CK knowledge base. The experiments conducted outperform state-of-the-art research by 20% and 80% for tactic and technique classification respectively. These results are ultimately incorporated in 2aCTI, a web application designed to facilitate TTP extraction for the cybersecurity researcher and practitioner. In doing so, this thesis hopes to contribute to the cybersecurity community’s efforts to increase the transparency, awareness and sharing of critical cybersecurity information.

Acknowledgments

I would like to express my gratitude to Prof Emil Lupu for his feedback, guidance and time throughout this project. I would also like to thank Alexander Brady for his unending support and friendship this past year.

Contents

1	Introduction	1
1.1	Motivation and Objective	1
1.2	MITRE's ATT&CK Framework	2
1.3	Challenges	3
1.4	State of the art	4
1.5	Contributions	5
1.6	Outline	6
2	Background	7
2.1	Text Classification:	7
2.1.1	Aim:	7
2.1.2	Pipeline:	8
2.2	Related Works:	12
2.2.1	Methodology:	12
2.2.2	RQ1: Aims and Approaches:	13
2.2.3	RQ2: Available Techniques:	16
2.3	Potential Threats to Validity	18
2.4	Summary of related work	19
3	Data Characterisation	21
3.1	Data Collection Strategies	21
3.1.1	Web scraping and ethical concerns	23
3.1.2	Text Processing:	25
3.2	Distribution of the distilled and enhanced dataset:	26
4	Experiments:	29
4.1	Methodology Overview:	29
4.2	Hypothesis:	31
4.3	Baseline:	32
4.4	Preprocessing:	32
4.5	Feature Extraction:	35
4.6	Classification Models	37
4.6.1	Linear Classifiers:	37
4.6.2	Non-linear classifiers:	41
4.6.3	Neural Networks	43
4.7	Preliminary Conclusion and Choice of Model	45

5	Software Implementation:	47
5.1	Building blocks - user perspective:	48
5.2	Project Architecture:	49
5.3	Software Evaluation	51
6	Conclusion	53
6.1	Summary of contributions:	53
6.2	Limitations:	54
6.3	Future work	55
6.4	Practical, ethical and professional considerations	56
	Appendices	59
A	Appendix A - Ethics Checklist:	60
B	Appendix B - MITRE's ATT&CK license:	61

Chapter 1

Introduction

1.1 Motivation and Objective

IN a rare move earlier this year, the UK, EU and US governments released a joint statement accusing the Russian Federation of carrying out the cyber attack against Ukraine's satellite network which disrupted the country's communication networks ahead of the February invasion (4). This unprecedented attribution is just the latest light to be shed on our vulnerability to the cyber threat and further highlights the intractable technical and political problems that arise when seeking to identify the source of a malicious cyber incident. As a result of mounting awareness and concern, scholars have witnessed an "information explosion of Cyber Threat Intelligence (CTI) reports" (1) published by companies and governments alike. While valuable in uncovering attackers' Tactics, Techniques and Procedures (TTPs), CTI reports are often written in an unstructured manner, meaning that information extraction requires the manual intervention of a cybersecurity professional (2; 3). The time-consuming and expensive nature of TTP information extraction from CTI reports is therefore at odds with the volatility of the cyber threat environment, diminishing their utility over time (2).

The aim of this thesis is to fill this gap by automating the extraction of TTPs from cyber attacks reports. To this end, this thesis builds on existing research (notably, (2; 3)) and approaches the problem using state-of-the-art Natural Language Processing (NLP) and Machine Learning methods. Several classifiers and text representation techniques will be experimented with for the purpose of text classification. Finally, the results of these experiments will be implemented in a system which will retrieve TTPs using MITRE's® ATT&CK knowledge base - a framework used to describe adversarial tactics and techniques. Ultimately, this thesis hopes to contribute to the existing literature on text classification for unstructured and technical documents by providing an answer to the following research question: How can we best extract TTPs from CTI reports(2) ?

1.2 MITRE's ATT&CK Framework

In order to answer the research question at hand, this project turns to MITRE's ATT&CK knowledge base. Created in 2013, the ATT&CK framework - which stands for Adversarial Tactics, Techniques, and Common Knowledge - describes how adversaries penetrate networks, move laterally and are able to evade defences from the perspective of the attacker (5). More specifically, ATT&CK matrices organise adversary behaviours in a series of tactics which describe the attacker's reason or short-term goals for performing an action (6). In turn, each tactic comprises a series of techniques - that is, the way an adversary may try to achieve that tactical objective and the consequences for the victim if the attack is successful (6). Tactics and Techniques are central to this work as these constitute the labels used for text classification. Figure 1.1 shows the relationship between tactics and techniques in the ATT&CK framework.

Initial Access	Execution	Persistence	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
Data Historian Compromise	Change Program State	Hooking	Exploitation for Evasion	Control Device Identification	Default Credentials	Automated Collection	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Drive-by Compromise	Command-Line Interface	Module Firmware	Indicator Removal on Host	I/O Module Discovery	Exploitation of Remote Services	Data from Information Repositories	Connection Proxy	Alarm Suppression	Change Program State	Denial of Control
Engineering Workstation Compromise	Execution through API	Program Download	Masquerading	Network Connection Enumeration	External Remote Services	Detect Operating Mode	Standard Application Layer Protocol	Block Command Message	Masquerading	Denial of View
Exploit Public Facing Application	Graphical User Interface	Project File Infection	Rogue Master Device	Network Service Scanning	Program Organization Units	Detect Program State		Block Reporting Message	Modify Control Logic	Loss of Availability
External Remote Services	Man in the Middle	System Firmware	Rootkit	Network Sniffing	Remote File Copy	I/O Image		Block Serial COM	Modify Parameter	Loss of Control
Internet Accessible Device	Program Organization Units	Valid Accounts	Spoof Reporting Message	Remote System Discovery	Valid Accounts	Location Identification		Data Destruction	Module Firmware	Loss of Productivity and Revenue
Replication Through Removable Media	Project File Infection		Utilize/Change Operating Mode	Serial Connection Enumeration		Monitor Process State		Denial of Service	Program Download	Loss of Safety
Spearphishing Attachment	Scripting					Point & Tag Identification		Device Restart/Shutdown	Rogue Master Device	Loss of View
Supply Chain Compromise	User Execution					Program Upload		Manipulate I/O Image	Service Stop	Manipulation of Control
Wireless Compromise						Role Identification		Modify Alarm Settings	Spoof Reporting Message	Manipulation of View
						Screen Capture		Modify Control Logic	Unauthorized Command Message	Theft of Operational Information
								Program Download		
								Rootkit		
								System Firmware		
								Utilize/Change Operating Mode		

Figure 1.1: MITRE ATT&CK for Enterprise Matrix

The choice of the ATT&CK framework was motivated by two considerations:

- First, ATT&CK is considered the “industry standard” (7)(8) by many analysts as it offers a common taxonomy to describe and categorise the actions perpetrated by malicious actors (9). In 2020, for instance, a survey conducted by the Center for Long-Term Cybersecurity (CLTC) at the University of California Berkeley across “leading security professionals” in the United Kingdom, United States and Australia, found that 81 percent of large to medium-sized companies use MITRE's matrices in their security operations centre (8)(10). As a result of MITRE's wide acceptance in both research and industry domains (10), the literature on mining CTI reports frequently draws on the framework as exemplified by the tools developed by (2)(3) on which this thesis builds.
- Second, the depth and of its analysis, which is regularly updated with industry input, contrasts starkly with other well-known frameworks such as Lockheed

Martin’s Cyber Kill Chain (9). Indeed, while the Kill Chain model outlines seven high-level steps through which an adversary must successfully progress in order to compromise a system, the ATT&CK framework describes more than 200 unique adversarial techniques as well as mitigation strategies (11). As such, ATT&CK should be viewed as a complementary framework, sitting at a “lower level of definition” to the Cyber Kill Chain¹. Its depth is notably exemplified by the three technology domains it covers, each represented by distinct matrices and iterations of the framework:

1. **ATT&CK for Enterprise:** created in 2013 to describe attacks perpetrated against IT networks and cloud technologies (10). It addresses Windows, macOS and Linux operating systems as well as Office 365, Google Workspace, Network and SaaS platforms. As of 2022, it covers 222 techniques organised in 14 tactics.
2. **ATT&CK for Mobile:** released in 2017 to describe attacks perpetrated against Android and iOS devices. It covers 12 tactics organised in 86 techniques (10).
3. **ATT&CK for ICS:** describes post-compromise attacks perpetrated against Industrial Control Systems (ICS) defined as embedded cyber-devices that operate critical infrastructures (9) such as hospitals, government services or nuclear power plants². It covers 12 tactics and 88 techniques.

Of these three domains, this thesis focuses specifically on the **ATT&CK for Enterprise matrix**. The decision is driven both by data availability and the desire to compare the system built with existing works such as Legoy et al.’s rcATT (2) which refers exclusively to the Enterprise matrix.

1.3 Challenges

Simplifying the extraction of critical security information from CTI reports is an important task as the cyber security community is built around the premise that sharing TTPs and corresponding mitigation strategies “makes everybody safer” because it hinders the attacker’s ability to reuse them (13). Yet, automating this process is not a trivial task and several challenges — cautioned both in the relevant literature and encountered throughout this project — hinder the ability to use off-the-shelf language and classification models (13).

To achieve its objective, the following challenges are identified and will have to be overcome:

¹For MITRE’s view see:
<https://attack.mitre.org/resources/faq/>

²The UK Government has defined 13 Critical National Infrastructure: Chemicals, Civil Nuclear, Communications, Defence, Emergency Services, Energy, Finance, Food, Government, Health, Space, Transport, Water. (12)

First, as performance is inevitably correlated with the existence of sufficient, relevant and balanced training data, this thesis will have to collect a robust dataset. By relevant, we mean data that is a true representation of the data we are likely to be given as input by the user; by balanced, we mean a dataset in which each class has a (roughly) equal number of data points (14). However, and as will become clear in the literature review, neither of these requirements are currently met by available datasets. Instead, this thesis will have to adopt mitigation strategies such as web-scraping in order to collect more training data.

Second, unlike traditional text classification tasks, a key challenge for achieving the objective set is to disambiguate and attach meaning to technical terms and neologisms which suffuse CTI reports as a result of the “security arms race” (13). These includes cybersecurity-related concepts or names of Advanced Persistent Threat (APT) groups which often do not have sufficient linguistic meaning to fit predetermined models. Consequently, many pre-trained techniques for text representation will have to be adapted in order to take into account domain-specific vocabulary.

Third, the labels used for classification in this thesis far exceed standard text classification exercises. This is especially the case for technique classification. Indeed, the absence of a large dataset to classify 222 unique techniques will likely affect performance. Therefore, this thesis will have to consider establishing a threshold for the document to label ratio or combining certain techniques together. The work of Legoy et al., on which this thesis builds, will prove useful to answer this concern (2).

Coupled together, these impediments make CTI report classification a challenging endeavour which requires the adaptation of existing methodologies to the specificities of the task. At the heart of this project is therefore the aim to circumvent structural limitations and simplify the extraction of meaning from unstructured, technical language. Alongside existing research, the aim set by this thesis is important as government officials are pointing to the surprising discrepancy between public opinion’s limited appreciation of the cyber threat relative to its potentially severe impact on economic, social and human life (15) — a matter of “when, not if” for Ciaran Martin, former head of the UK’s National Cyber Security Centre (15).

1.4 State of the art

The literature on CTI mining is limited and research has predominantly focused on the extraction of Indicators of Compromise (IoCs) such as URLs and IP addresses(16; 13). Few efforts aim to automate the extraction TTPs as this thesis does. Existing work of TTP extraction specifically includes the works of (2; 3; 1) on which this thesis builds.

To the best of our knowledge, Legoy et al.’s research is the latest effort to automated the extraction of TTPs from CTI reports using MITRE’s ATT&CK framework. Using data provided by the MITRE organisation, the authors experimented with standard

multi-class multi-label classifiers. The best performing models — a Support Vector Machine (SVM) with TF-IDF text representation — was then implemented in a tool named *rcATT*. Specifically, given an input text, the system predicts both the tactic and technique IDs present in the document.

For ease of comparison with the tool built in this project, Figure 2.1 illustrates how *rcATT* is presented.

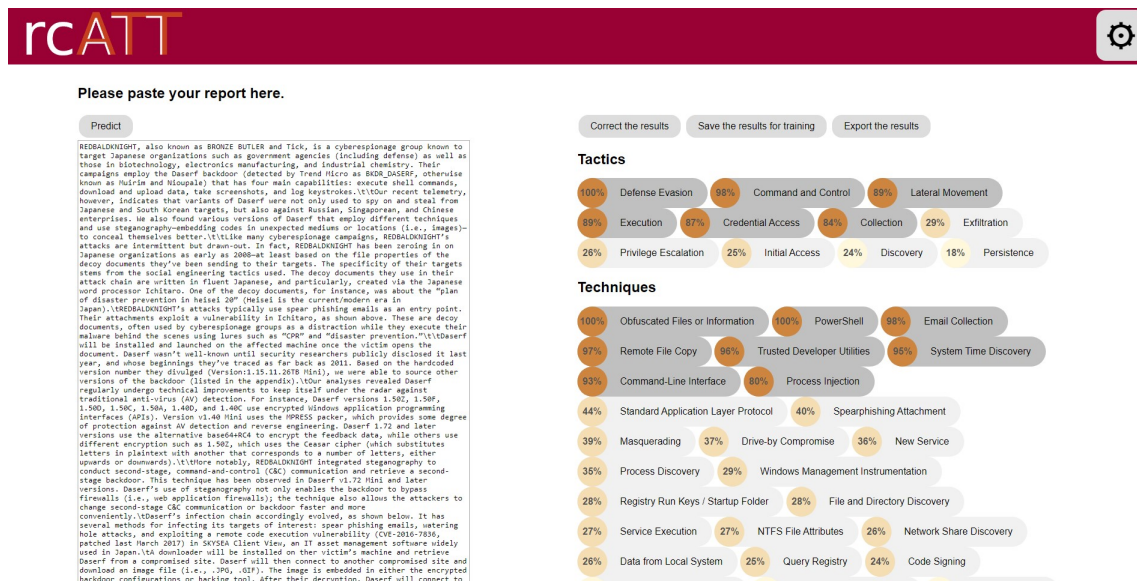


Figure 1.2: *rcATT*'s graphical interface

From their efforts, this thesis adopts the same approach to classification - namely, building separate classifiers for tactic and technique classification separately. This thesis also borrows from them the decision to approach CTI classification at the document rather than the sentence-level due to data availability. A more thorough description and assessment of related research can be found in section 2.2 of Chapter 2.

1.5 Contributions

Given this background, the contributions of this thesis to the relevant literature are:

1. A new data collection and preprocessing pipeline is presented using web-scraping and natural language processing techniques to mitigate against issues of data scarcity found in the relevant literature.
2. Several multi-class, multi-label text classification models are compared and evaluated. These experiments notably improve on existing work by testing state-of-the-art feature extraction methods as well as neural network-based architectures. Looking at the macro average f-score, *the experiments conducted*

improve on rcATT by 20% for tactic classification and more than 80% for technique classification.

3. 2aCTI, a system to automate the extraction of TTPs from CTI reports based on the findings is developed. Compared to rcATT, 2aCTI incorporates the following novel features:
 - Users can retrieve TTPs either by searching the name of a malware or by predicting their presence in CTI reports.
 - In the first case, a description of the attack is provided by drawing on MITRE's attack database.
 - In the second case (which is most similar to that of Legoy et al.), 2aCTI retrieves the most relevant sentences present in the document to understand the elements which contributed the most to the overall classification.
 - In both cases, a description of each tactic and technique as described by the MITRE organisation is provided.
 - Finally, the confidence scores for tactic and technique classification are more robust than those of rcATT.

1.6 Outline

The remainder of this thesis is organised as follows. Chapter 2 provides the background to this thesis. It begins by introducing a standard text classification pipeline before discussing the approaches and techniques used in academic works related to this project as well as potential threats this thesis may face. Chapter 3 introduces the dataset used in this project, how it was collected and what techniques were used to process reports. Chapter 4 focuses on the experiments conducted for the text classification task at hand. This includes a discussion of the preprocessing and feature extraction strategy and presents the performance of linear, non-linear and neural network-based models. Finally, Chapter 5 presents 2aCTI, the software developed to extract TTPs from CTI reports.

Chapter 2

Background

This Chapter is designed to provide the background to this thesis. It is organised in two sections: the first is intended to grant the reader with the technical background for a standard text classification task; the second reviews and evaluates approaches and techniques used in the related literature.

2.1 Text Classification:

2.1.1 Aim:

Intuitively, text classification aims to map a new, unseen, text input to a predefined output called a label (17)(18). More formally, the input is typically a document, sentence or sequence of text part of a dataset D where $D = \{X_1, X_2, \dots, X_N\}$ (19). Each input X_i is a data point which in turn contains one or more words called *tokens*, each made up of one or more characters. In the context of this thesis, we consider X_i to be an entire report rather than a segment. To achieve the objective set, the classification algorithms will thus seek to learn a function f to predict one or more labels C for an unseen document X_i with a reasonable degree of confidence so that each document is assigned at least one label (20)(17). Here, the label set C is decomposed in two, where C_T is used to refer to the label set for tactic classification and C_t refers to the labels for technique classification.

$$f : X_i \rightarrow (C_{Tk} || C_{tk}), f \in H \quad (2.1)$$

As seen in equation 2.1, the function f takes as input a CTI report X_i from D and maps it to k possible output labels, where C_k is a label from the set of classes $C = \{C_1, C_2, \dots, C_k\}$. Moreover, H is the hypothesis space within which we search for a mapping function f consistent with the training set. Once a model is learned on the training set, it can be used to classify an unseen document X_j from a test set.

This learning process is typically achieved via supervised learning which rests on a similar logic as inductive reasoning: algorithms seek to induce patterns from a set of examples in order to generalise their classifier to new, unseen, instances (21) (22). In a traditional classification task, each input is considered in isolation of each

other (23) and problems typically involve a binary response (for instance, either a mail is a spam or it is not) (24). In this case, however, the classification task is less straightforward as a single CTI report can make reference to more than one label (for both technique and tactic). The classification model at hand is therefore a **multi-class, multi-label classification** task in which a single instance is associated with a set of non-mutually exclusive labels.

2.1.2 Pipeline:

Text classification tasks, whether multi-label problems or not, typically follow the same four phases, as deconstructed by Kowsari et al. (19):

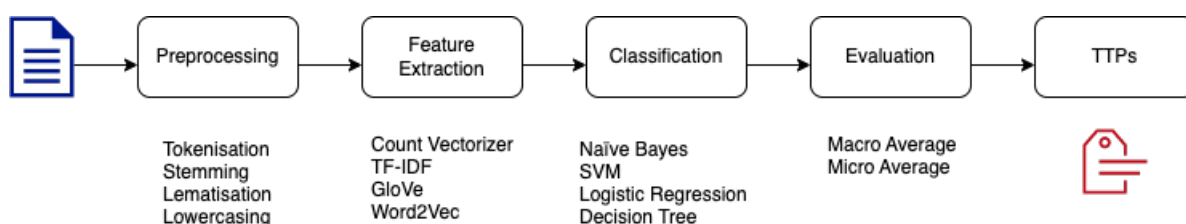


Figure 2.1: Text Classification Pipeline as described by Kowsari et al. (19)

1) Data Collection:

The first step for any NLP-related task is to collect sufficient and relevant textual data. Important properties include corpus balance - that is, the range of genre and language used in a dataset - and data that is representative of the language variety found in the real world - which often includes mistakes, typos and, in the case of unstructured documents, varying lengths. These requirements can easily be achieved by turning to readily available corpuses such as NLTK's WordNet or gensim's collection of Wikipedia articles and are crucial in determining a classifier's performance (14). Yet, when the data of interest is either of a technical nature or in a foreign language, researchers have to design their own corpuses. Corpus design can be achieved either by annotating data manually or by adopting new strategies, as is the case for this project.

2) Text Pre-Processing:

Once we have sufficient labeled data, we need to ensure that textual data can be converted into a machine-readable format. Removing noise from documents and cleaning our dataset is crucial as it could have detrimental effects on the model's performance later on.

To do so, researchers often begin by breaking down documents or sentences into

words in a process termed *tokenisation* so that each token can be individually inspected (19). Upon inspection, textual data typically includes irrelevant or insignificant words known as stop words (such as *the* or *a*) which do not help us in our classification endeavours (19). Additional cleaning can be done by performing stemming (reducing a word to its stem, so that *performing* becomes *perform*) or lemmatisation (reducing a word to its most basic form, so that *ran* becomes *run*). Last, textual data can be standardised by lowercasing it.

3) Feature Extraction:

After text pre-processing, the data must be converted into a structured feature space to be passed into a classifier (19). Popular techniques for feature extraction include:

- **Count Vectorizer:** Count vectorizer is a bag of words model in the sense that it is only concerned with the count of words. The order in which words appear, grammar, sentence boundaries and paragraphs are all ignored in this representation (25). For instance, the sentence [Stuxnet is a complex malware with many complex components] is transformed into the matrix:

Stuxnet	is	a	complex	malware	with	many	components
1	1	1	2	1	1	1	1

To predict a label using a bag of words representation, each token in a vocabulary is attributed a unique ID and each document X_i is then converted into a vector of token counts, with all duplicates collapsed together (23). As such, documents with similar words or with a similar frequency of words will have their vectors close to each other in Euclidean space (14).

- **Term Frequency and Term Frequency-Inverse Document Frequency:** In TF-IDF, tokens in a dataset that do not help us in deciding whether a document X_i is part of label C_j will contribute to a lesser extent to the overall classifier. In other words, terms that appear rarely in the corpus D but often in a document X_i are probably important in classifying X_i (14). Formally, the TF-IDF vectoriser is the product of TF and IDF scores, where term frequency measures the relative frequency of a term in a given document and inverse document frequency measures its importance across a corpus by giving higher weights to rare terms and lower weights to more common ones such as stop words (14). Both TF and IDF measures are described in mathematical terms in equation 2.2 and 2.3 (14).

$$TF(t, d) = \frac{f_{t,d}}{\sum_{t \in d} f_{t,d}} \quad (2.2)$$

Where the numerator is the number of occurrences of a term t in a document d and the denominator is the total number of terms in the document d (14).

$$IDF(t) = \log_e \frac{N}{|(d \in D : t \in x)|} \quad (2.3)$$

Where the numerator is the total number of documents N in a corpus and the denominator is the number of documents in our corpus in which the term t appears (14).

- **Word Embeddings:** So far, all feature extraction methods reviewed fail to captured any meaningful relationships between words (14). In recent years, methods using neural networks have emerged as a means to capture the semantic meaning of a word from the context in which it appears and its distribution within that context or “textual vicinity” (14). To do so, words with similar meaning are mapped to a N dimension vector and are represented close to each other within that vector space (19).

An example of a word embedding algorithm is the **word2vec algorithm**. It was first published in 2013 by Mikolov et al. in a seminal paper (26) presenting a two-layer neural network-based algorithm which takes words close to each other as contexts with respect to a target word w_m (25). To illustrate the algorithm, we can draw on their popular example which represents the word *Queen* by the vectors:

$$\text{vector}(\textit{“King”}) - \text{vector}(\textit{“Man”}) + \text{vector}(\textit{“Woman”}) \quad (2.4)$$

In particular, the word2vec algorithm is based on the hypothesis of “distributional similarity” which holds that the meaning of a word can be derived from its context and be a predictor of semantic similarity (27). In other words, the algorithm uses the logic of word embedding because words distributionally similar will appear clustered together in a vector space (14).

While word2vec is one of the most popular embedding techniques, more recent versions have sought to extend Mikolov et al.’s work. This includes GloVe (for “Global Vectors for Word Representation”) or Facebook’s FastText. In 2014, Le and Mikolov (28) proposed doc2vec, an algorithm which rests on a similar logic to that of word2vec with the difference that it learns the distributed vector representation on texts of variable length (28). These embeddings are tested in Chapter 4.

Finally, researchers have two options when using word embedding. They

can either turn to pre-trained word embeddings on news articles or Wikipedia pages. These notably include Google’s word2vec or Facebook’s FastText which are trained on Google News or Wikipedia articles respectively. Alternatively, researchers can train their own embeddings, as will become clear in the following chapter.

4) Dimensionality Reduction:

In some cases, feature representations can be very large, meaning that time complexity and memory consumption become expensive. To mitigate this risk, dimensionality reduction algorithms such as Principal Component Analysis (PCA) are available to researchers seeking to reduce the size of their feature space, avoiding problems of overfitting (19). The PCA algorithm will be tested when experimenting with logistic regression classifier in Chapter 4.

5) Classifier Selection:

Choosing the best classifier is one of the “most important step of the text classification pipeline” (19) as it determines how well the classification task performs. Popular text classification models used in the reviewed literature can be divided between linear models (Naive Bayes classifiers, Logistic Regression, Support Vector Machines (SVM)), non-linear models (Decision Tree, k-Nearest Neighbours) and, to a lesser extent, Neural Networks-based models (Transformers, Multi Layer Perceptrons). To develop rcATT, Legoy et al. experimented with the first two, omitting more complex models. By contrast, this thesis will seek to experiment with neural network-based classifiers. This effort will prove particularly fruitful for tactic classification as we observed a improvement of the macro average f-score of up to 70%. A description of each classifier implemented can be found in Chapter 4.

6) Evaluation:

Last, once textual data has been passed into a classifier, its performance must be assessed. Importantly, evaluating the performance of a multi-class, multi-label classification task poses unique challenges not found in more traditional single-label tasks (20). Performance measures are typically based on a confusion matrix (see Figure 2.1), a table displaying the number of correct and incorrect predictions to measure a classifier’s recall, precision, F -measure and accuracy (24). However, since the dataset collected is unbalanced (with more reports for certain tactics and techniques than others), focusing on accuracy is not appropriate as the model which yields the highest accuracy may simply be the one which predicts the tactics or techniques with the largest number of reports (24). Instead, this thesis focuses on the more nuanced metrics recall, precision and F -score, as displayed in equations 2.3, 2.4 and 2.5.

- **Precision:** Measures the accuracy of a predictive positive outcome.

$$Precision = \frac{TP}{TP + FP} \quad (2.5)$$

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figure 2.2: Confusion matrix for a binary response (29)

- **Recall:** Measures the ability of the model to predict a positive outcome.

$$Recall = \frac{TP}{TP + FN} \quad (2.6)$$

- **F-score:** Measures the harmonic mean of a model's precision and recall.

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (2.7)$$

For all three metrics, we distinguish between macro and micro averaging. While the latter assigns equal weight to each document, the former disregards the frequency of each category and instead assigns equal weight to each label (19). As such, we have focused here predominantly on macro scores and specifically on F beta score to evaluate the performance of our classifiers. The reason behind the choice of a $F_{0.5}$ score is that it gives more importance to precision than recall - that is, it minimises the risk of false-positive errors (predicting a tactic or technique in a report when in fact it was not present) rather than minimizing the false-negative errors (not predicting a tactic or technique in a report when in fact it is present).

2.2 Related Works:

2.2.1 Methodology:

With an understanding of how text classification tasks are typically accomplished, we now turn to an analysis of related work. The aim of this literature review is to identify and evaluate research which focuses on the automation of TTP extraction from cyberattack reports. As pointed to by Legoy et al. (2), prior to their work, only five research papers have sought to retrieve TTPs from the MITRE ATT&CK framework in unstructured text. This thesis thus seeks to contribute to this nascent effort.

For the purpose of this review, we excluded publications that were not peer-reviewed nor published in English and focused on publications which sought to extract information from CTI reports. This meant excluding research papers which have drawn on a wide variety of sources, from Twitter feeds (30) to hacker forums (31) and version control repositories such as Github (32)(33). With this criterion in mind, we analyse in this section nine works related to CTI report mining, structured around two research questions:

1. RQ1: What are the objectives addressed by researchers, how do they achieve their objectives, and what sources do they draw on?
2. RQ2: What techniques are available to classify unstructured CTI reports and how successful are they in doing so?

Table 2.1 serves to summarise our discussion.

Software Name	Author	Date	Dataset Size	Aim
FeatureSmith	Zhu et al.	2016	1,068 security papers	Feature Engineering
iACE	Xiaojing et al.	2016	71,000 security blogs	IoC extraction
TTPDrill	Husari et al.	2017	1,700 reports from Symantec	TTP and IoC extraction
N/A	Ramnani et al.	2017	tested on 18 threat advisories	Information extraction
ChainSmith	Zhu et al.	2018	14,155 security blogs	IoC extraction
N/A	Ayoade et al.	2018	18,257 reports	TTP extraction
ActionMiner	Husari et al.	2018	2,200 malware reports	Information extraction
rcATT	Legoy et al.	2020	1,490 security reports (MITRE)	TTP extraction
Extractor	Satvat et al.	2021	8,000 threat sentences	Information extraction

Table 2.1: Review of related work - Sources used and aim

2.2.2 RQ1: Aims and Approaches:

The majority of tools surveyed mine CTI reports for one of three purposes: a) information identification and extraction such as threat actions, b) IoC extraction and c) TTP extraction using existing taxonomies such as MITRE's ATT&CK framework,

CAPEC or the Cyber Kill Chain (33). In the first category, we find works that seek to retrieve threat actions through the search of keywords or semantic relations (33). As each study has its own definition of what constitutes a threat actions, these work often prove difficult to compare. By contrast, those whose aim is to extract Indicators of Compromise (IoCs) such as URLs or IP addresses are easier to compare as they can rely on existing patterns or rule-based methods for identification. However, our view is that these often fail to retrieve meaningful data as IoCs are stripped of their threat context. Last, research that fall in c) can leverage on more extensive and standardised descriptions which facilitates the use of advanced techniques such as Machine Learning. Among these efforts, it should be noted that Zhu and Dumitra's FeatureSmith (2016) stands as an outlier as the only tool whose purpose is to automatically generate features for training machine learning classifiers, specifically associated with Android malware (34).

For the purpose of cross-study comparison, we compare our software to those studies whose purpose is the extraction of TTPs specifically. At the same time, as the literature on the automated analysis of CTI reports is limited, this thesis considered methods and approaches used in all three research areas to devise its strategy. This covers decisions for data collection (where can we retrieve CTI reports? How should we approach text data - at the document or sentence level?), information extraction (are rule-based models sufficient or should we turn to supervised learning approaches?) and evaluation (which research papers are the most effective in achieving their intended aim and what can we learn from them?).

Husari et al.'s ActionMiner (35), Satvat et al.'s Extractor (36) and Ramnani et al.'s work (37) all seek to extract threat actions: ActionMiner borrows from Information Theory entropy and mutual information gain metrics to design a system capable of extracting "low-level threat actions" from CTI reports (35). Here, threat actions are understood as crucial verb-object pairs in CTI reports such as *upload files* or *inject process*, used to reduce CTI reports to a structured list (35). Moreover, information gain is a common feature selection method in cases where features are redundant to filter inappropriate and uninformative features (38). In our context, we decided not to use information gain as it sensitive to attributes with a large number of values — for instance, tactics with a large number of reports — thus increasing risks of overfitting. Ramnani et al., on the other hand, use a semi-automated methodology to identify a sequence of tokens to assist the extraction of "information nuggets" such as exploit targets and mitigation techniques (37; 35). Due to the lack of any available labelled dataset, the authors fail however to test their methodology on a supervised learning classifier (37). Last, in Extractor, Satvat et al. aim to visualise attack behaviours from CTI reports through the use of provenance graphs - a form of directed acyclic graph - by passing reports through several rounds of text normalisation, homogenisation and summarisation using NLP techniques (36). Yet, as the authors approached attack behaviours at the sentence-level, Extractor can difficulty take into account instances where the threat is described over several sentences(36), a scenario likely to occur in the case of large reports.

Second, the vast majority of efforts to mine CTI reports aim at extracting Indicators of Compromise (2) - that is, “an observable attack artefact (...) that the analyst has linked to a given security threat” (13). Xiaojing et al.’s iACE (16) and Zhu et al.’s ChainSmith (13), but also the works of (39; 40) to name a few, have sought to contribute to this effort. They were notably inspired by platforms such as Facebook ThreatExchange(41) or AlienVault OTX (42) which are designed to exchange IoCs between participants. For instance, ChainSmith extracts IoCs and maps them to their relevant campaign stages (13) which form part of a malware delivery model constructed by the authors to understand the role a given IOC played in an attack and to reconstruct campaigns automatically (13). However, as noted by (2), IOC extraction is now facilitated by the existence of multiple open-source tools (see “IOC extraction” on Github) and the authors often resort to defining a set of regular expressions (see below).

In terms of their added value, we agree with Zhu et al. that extracting IoCs must be coupled with insights about the campaign itself, especially as “network identities of servers involved in malware delivery campaigns already change frequently” (13). This is not facilitated by the fact that the web application on which the ChainSmith system is released no longer works. As the aim of this thesis is to facilitate the analysis of CTI reports, we decided to focus on the extraction of more meaningful information rather than concise artefacts whose meaning is context-dependent. Indeed, by relying on loosely defined threat actions and techniques rather than existing taxonomies shared by the wider cyber security community such as the MITRE framework, the research papers mentioned above fail to generalise well and cannot easily be used in cross-study comparisons such as (25). In our view, these efforts miss an opportunity to leverage on standardised frameworks to build their tool which would facilitate their use by cybersecurity practitioners.

Bridging both b) and c) is Husari et al.’s TTPDrill which is one of the research papers which most closely echoes the aim of this thesis. Indeed, TTPDrill differs from previous efforts as it draws on MITRE’s ATT&CK framework and the CAPEC¹ threat repository to build a new threat-action ontology (1). TTPDrill extracts threat actions from CTIs and maps them to the TTP included in the novel ontology which treats tactics and techniques as similar labels (1)(2). TTPDrill, however, is not publicly available and thus cannot be easily reproduced to other contexts (2). More importantly, by building a new ontology rather than drawing on MITRE’s framework, TTPDrill suffers from the same limitation as the research papers described above.

Building on TTPDrill, two studies closely resemble the intention set by this thesis. These are Ayoade et al.’s work and Legoy et al.’s rcATT. Both leverage NLP techniques and Machine Learning classifiers to extract TTPs from CTI reports following MITRE’s ATT&CK framework. Like TTPDrill and ActionMiner, Ayoade et al.’s work is

¹CAPEC (Common Attack Pattern Enumeration and Classification) is MITRE’s dictionary of known attack patterns.

not publicly available and cannot be reproduced for our purposes (2). To the best of our knowledge, the only accessible tool is that of Legoy et al. who have developed rcATT. Their system is a Python application which accepts a text as input and identifies the corresponding MITRE TTPs from the Enterprise ATT&CK framework. Users can visualise the confidence score associated with each TTP prediction and can interact with rcATT's feedback mechanisms if they disagree with the tool's results (2).

In our view, however, rcATT has several limitations: First, due to a lack of sufficient labelled data, the macro average F -score reported by rcATT rarely surpass 50%. Second, while the authors designed a means for users to correct the prediction to improve the overall classification, we noticed that the installation process is not straightforward, limiting its use for research purposes. Third, if users do not agree with the result, they must read through the entire report in order to identify which tactics or techniques were misclassified. This requires both time and a strong grasp of MITRE ATT&CK database which young cybersecurity analysts may not have.

Finally, regarding data collection strategies, papers reviewed differed in the sources they used: they either turned to existing repositories (Legoy et al. (2)) or created custom scrapers (13; 1; 16). Moreover, some used the entire CTI report to train their model (2; 1; 35; 3; 16), while others extracted individual sentences from reports (34; 36; 13). Due to constraints in data availability, we chose to focus on document-level classification and thus follow closely Legoy et al.'s approach (2). This was also in part justified by the fact that for Ramnani et al., the low precision of their model across documents was attributed to difficulties in extracting complex sentences from documents (37).

For those who turned to Machine Learning classifiers, access to labelled data proved particularly challenging (2; 13; 1; 3). To counter this limitation, Ayoade et al. were distinctly productive: to enhance a small labelled datasets retrieved from the MITRE ATT&CK website (169 reports), the authors downloaded 17,600 reports from the Symantec threat report website and 488 from a variety of security vendors and labelled all 18,088 reports manually. Likewise, (43) annotated their corpus manually by turning to "security professionals with NLP-related work experience". Finally, those more constrained by time turned to third-party platforms such as Symantec's WINE platform for data-intensive experiments (13).

2.2.3 RQ2: Available Techniques:

The techniques used in existing work range from NLP and semantic analysis to Machine Learning models. Most papers reviewed broadly followed Kowsari et al.'s text classification pipeline (19) with differing levels of success depending on their chosen strategy. Here, we follow Kowsari's work to review techniques used in related research.

First, looking at preprocessing techniques, some (Legoy et al. (2)) implemented

traditional methods such as stemming, lemmatisation and stop-word removal by focusing on contractions and “computer science related terms that hinder the classification” (44) while others turned to more custom techniques. This was notably the case of Husari et al. (1) who turned to Support Vector Machines (SVM) to filter out irrelevant reports and focus solely on the frequency of “highly-reputable” security standards or nouns as defined by the authors (1). These choices illustrate the lack of consensus regarding the impact of preprocessing techniques on text classification tasks (45). Indeed in the wider literature on text classification, scholars have at times shown that stop-word removal improves classification accuracy in both English and foreign language datasets (46) and at others proven that common classification models, notably SVM, perform better when preprocessing is not used (47). To ensure that preprocessing would not have a negative effect on our classifier, we tested several techniques which we describe in Chapter 4.

Second, regarding feature extraction tools, Zhu and Dumitra’s FeatureSmith is the first attempt at automating the feature engineering process from CTI reports and security conferences (34). Their tool is designed to suggest features for the detection of malware relying on type dependency and part-of-speech (PoS) techniques. The former is concerned with how words relate to others while PoS tagging refers to the process of classifying words according to a tag set such as *adjective*, *verb* or *noun* (19). To generate the relevant features and extract malware behaviours, Feature-Smith begins by parsing the literature using the Stanford typed dependency parser in search of “semantically related” concepts (34). Once extracted, the tool computes a similarity score for each feature using a semantic network - that is, a graphical representation of concepts (in this case, Android malware behaviours) and their relationships with each other (that is, connecting related concept) (34) - to indicate their usefulness in detecting future Android malware (34). In doing so, Feature-Smith is able to mirror the manual feature engineering process (34). While this effort is valuable for the application of Machine Learning techniques to the security literature, the author’s narrow focus on Android malware detection means that it cannot be easily applied to other contexts(2) and could therefore not be leveraged by this thesis.

Third, we distinguish two approaches for classifier selection and information extraction: those who only turned to rule based classification and those who used either linear or non-linear classifiers. The authors of TTPDrill, Extractor and iACE all draw to a certain extent on rule based classification to extract the information of interest(1; 36). To do so, the authors construct a set of rules used to test whether an unseen document or sentence adheres to a corresponding rule or not. This is notably achieved by devising a list of regular expressions (regexes) to map common objects such as IP addresses and port numbers with natural language (1). Examples of rules borrowed by Zhu et al.’s ChainSmith (13) to identify common IoCs can be found in table 2.3.

While rule based classification is comparatively easy to implement, it requires sig-

Type	Rules
URL	Top level domain must be identified
IPv4	Contains 4 digits (i256) and the address is not reserved
hash	A hexademical string of length 32, 40 or 64

Table 2.2: Example of regex rules used by Zhu et al. in ChainSmith (13)

nificant manual effort to identify which rules ought to be created. Instead, others (13; 2; 3) leverage on supervised learning techniques to induce patterns for classification. In ChainSmith, the authors combine common NLP techniques with a Machine Learning Model to extract IOCs and their role in a given attack (13). To build their IOC classifier and the campaign stage a given word is associated with, ChainSmith uses a neural network with 1 hidden layer and 50 hidden nodes with a logistic function used to scale the output probability (13). The system achieves more than 90% in both precision and recall in extracting IoCs. The robustness of these results inspired this thesis to experiment with neural network-based model.

Last, faced with the same multi-label classification problem as we are, Legoy et al. (2) used two approaches to train their dataset: 1) binary relevance which ignores any relationship between labels and treats each label as a separate binary classification problem (48) and 2) classifier chain which trains each classifier on the input space of all previous classifiers in the chain (49). After testing several standard classifiers using both approaches, the authors built rcATT using a binary relevance SVM classifier with TF-IDF text representation. The authors achieve a macro average f -score of 59.47% for tactic classification and 25.06% for technique classification, leaving ample room for improvement. As will be detailed in Chapter 4, this thesis extends their work by experimenting with neural network-based feature extraction and classifiers.

2.3 Potential Threats to Validity

From the above review, four limitations cautioned by authors are discussed in this section.

First, a number of authors (43; 2; 3) point to the lack of sufficient labeled data available for training. This limitation is critical to those who followed a Machine Learning route, as these tasks typically require large amounts of data in order for algorithms to learn and generalise well to unseen data (50). As discussed, some palliated this problem by annotating their corpus manually (43)(3) while others (13) resorted to third-party platforms. Due to time and resource constraints, however, neither of these avenues were available to us. Moreover while a number of repositories exist such as (51) or (52), these provide either unlabelled data or do not follow MITRE's ATT&CK framework. Instead, we followed (2)'s advice and turned to MITRE's ATT&CK repository to build a first dataset on. A more detailed descrip-

tion of our data collection strategy can be found in Chapter 3.

Second, in our context, data scarcity is worsened by the unstructured and non-standardised nature of CTI reports. By contrast to structured text, which is understood as data in which there is a “regular and predictable organisation of entities and relationships” (23), unstructured data requires the researcher to undertake an important preprocessing exercise in order to retrieve meaning from text. Here, devising a standardised preprocessing pipeline is notably hindered by the fact that CTI reports are sourced from different organisations and companies, each employing different formats and who categorise attack behaviours in different ways (3). Moreover, while these reports are often written by cybersecurity professionals, the human component inevitably means that some CTI reports may be incorrect, introducing some noise and bias affecting empirical results (13). To mitigate against this threat, we followed the advice of researchers such as (13) use reports from a variety of sources to minimise the impact of individual false positives. (13).

Third, the profusion of cybersecurity-related terms such as *worm* or *domain*, as well as names of attack groups such as *Poseidon* or *Orange Worm* — which also refers to a species of insect — often have different meanings from words used in the English dictionary on which traditional NLP techniques are trained (36). Taken together, these limitations make it more challenging to classify security articles with off-the-shelf NLP and ML techniques (13).

2.4 Summary of related work

We conclude this Chapter by providing a summary of the research reviewed and introduce the approach undertaken by this thesis in light of the survey.

The literature on CTI mining reviewed in this chapter covered three research areas: 1) non-standardized information extraction, often referred to as threat actions (see (35; 36; 37), 2) IoC extraction whose meaning is unfortunately context-dependent (see (13; 16) and 3) TTP extraction which use well-established frameworks such as MITRE’s ATT&CK knowledge base. Given the limitations identified in the first two areas — namely, their reliance of frameworks created *ex-novo* which cannot easily be generalised and their failure to capture meaningful information — this thesis focuses specifically on TTP extraction. It therefore most closely resembles the works of (2; 3; 13).

Regarding data collection strategies, related research has either approached the problem of data scarcity by turning to existing but limited datasets (2) or by implementing strategies for data augmentation such as web-scraping (1; 13). To improve classifier performance, this thesis combines both approaches as described in the following Chapter. Additionally, the existing literature is divided between those who treat document classification at the sentence-level (34; 36; 13) and those who approach it at the document-level (2; 1; 35; 3; 16). As a result of data availability,

this thesis focuses on document-level classification specifically.

Last, the classification methods reviewed included both rule-based (1; 36; 16) and supervised learning approaches (2; 3; 13). In the latter case, the complexity of models implemented varied from standard classification linear and non-linear classifiers (2) to more complex neural network-based models (13). Legoy et al.'s work was identified as the work which most closely resembles the aim set by this thesis and similar strategies — notably the use of both binary relevance and classifier chain to tackle the problem of multi-class, multi-label classification — will be tested in Chapter 4. This thesis will extend their efforts by experimenting with neural network-based classifiers and text representation techniques, inspired by the promising results of (13).

Chapter 3

Data Characterisation

3.1 Data Collection Strategies

Faced with the absence of any labelled dataset which fit the requirements set by the project, this thesis had to resort to a number of strategies to build a new dataset. Figure 2.1 shows the structure of our data collection pipeline which we used to create our labelled set.

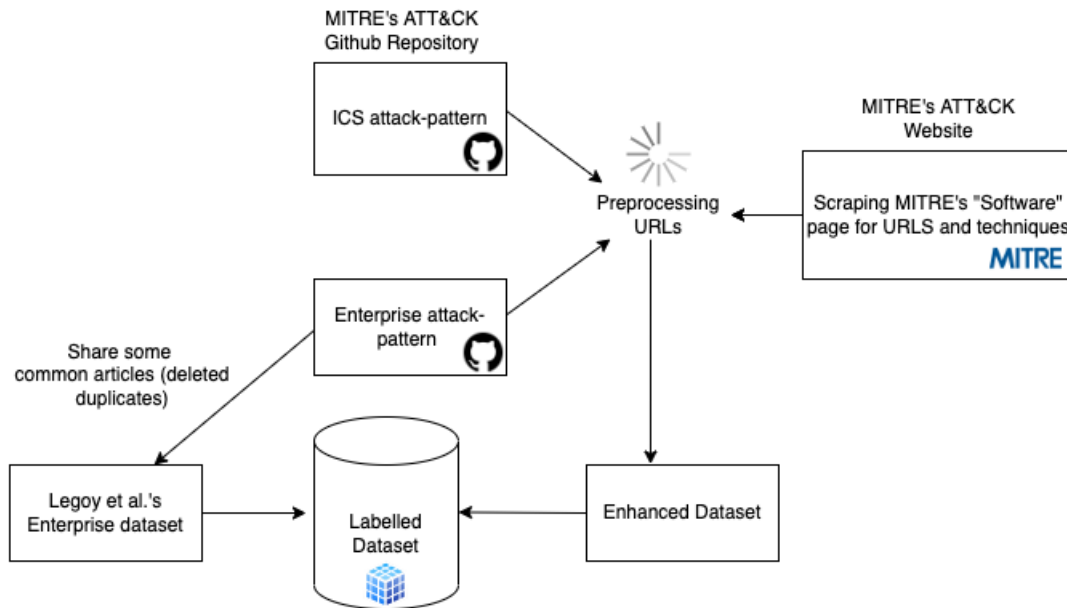


Figure 3.1: Data Collection Pipeline

First, we drew on Legoy et al.'s work (2) which is one of the only paper reviewed which has made its code publicly available¹. From their work, we retrieved their “training data original” file which we used as a first dataset. The rcATT original dataset comprises of 1,490 reports which have already been pre-processed by the authors and 228 labels (tactics and techniques) (2). We noticed however that 23 reports were in fact duplicates which we removed, leaving us with a dataset of 1,467

¹<https://github.com/vlegoy/rcATT>

reports. Moreover, as noted by the authors and seen in Figures 3.2 and 3.3, their dataset is characterised by some imbalanced, particularly regarding the number of reports per technique. As an attempt to solve imbalances, the authors chose to omit all techniques with fewer than 5 reports (2). All tactics, on the other hand, were kept as they linked to "at least 80 reports" (2).

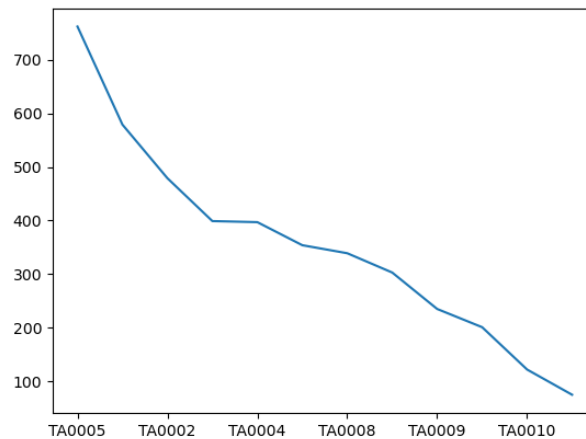


Figure 3.2: rcATT's distribution of reports per tactic

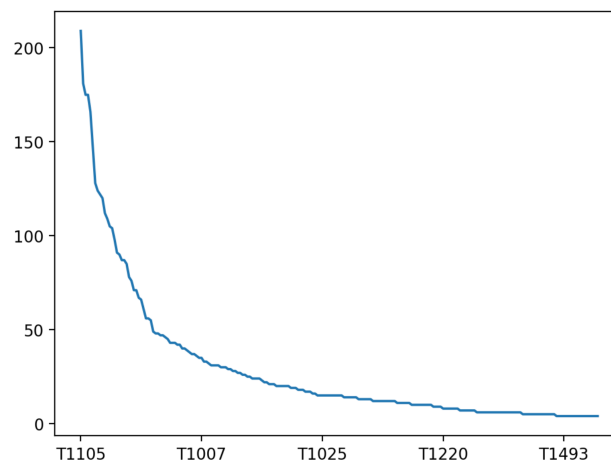


Figure 3.3: rcATT's distribution of reports per technique

Second, looking at rcATT's performance it was quickly clear that the dataset at hand was not large enough. We thus proceeded to collect additional reports directly from the MITRE ATT&CK repository which is stored on Github². The repository consists of a series of JSON files (one per technique) which include one or more external references. Figure 3.4 shows the structure of a JSON for the Wireless Sniffing technique. All JSON part of the Enterprise and ICS attack patterns were processed in order to create a second dataset which included URLs and their associated TTPs.

²<https://github.com/mitre/cti>

```

{
  "type": "bundle",
  "id": "bundle-9a51255a-f8a6-4864-ac87-64da8501b9a9",
  "spec_version": "2.0",
  "objects": [
    {
      "x_mitre_platforms": [
        "PRE"
      ],
      "x_mitre_domains": [
        "enterprise-attack"
      ],
      "object_marking_refs": [
        "marking-definition--fa42a846-8d90-4e51-bc29-71d5b4802168"
      ],
      "id": "attack-pattern--0a241b6c-7bb2-48f9-98f7-128145b4d27f",
      "type": "attack-pattern",
      "created": "2020-10-02T17:05:43.562Z",
      "created_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5",
      "external_references": [
        {
          "source_name": "mitre-attack",
          "external_id": "T1597.002",
          "url": "https://attack.mitre.org/techniques/T1597/002"
        },
        {
          "source_name": "ZDNET Selling Data",
          "url": "https://www.zdnet.com/article/a-hacker-group-is-selling-more-than-73-million-user-records-on-the-dark-web/",
          "description": "Cimpanu, C. (2020, May 9). A hacker group is selling more than 73 million user records on the dark web. Retrieved October 20, 2020."
        }
      ],
      "modified": "2021-04-15T03:44:43.900Z",
      "name": "Purchase Technical Data",
      "description": "Adversaries may purchase technical information about victims that can be used during targeting. Information about victims may be available for",
      "kill_chain_phases": [
        {
          "kill_chain_name": "mitre-attack",
          "phase_name": "reconnaissance"
        }
      ],
      "x_mitre_detection": "Much of this activity may have a very high occurrence and associated false positive rate, as well as potentially taking place outside the",
      "x_mitre_is_subtechnique": true,
      "x_mitre_version": "1.0",
      "x_mitre_modified_by_ref": "identity--c78cb6e5-0c4b-4611-8297-d1b8b55e40b5"
    }
  ]
}

```

Figure 3.4: Json file for the Wireless Sniffing technique from which URLs were extracted

At this stage, we conducted a cleaning exercise to remove URLs from sources which did not qualify as CTI reports. These included Wikipedia pages, Github links or pages which were not written in English. While Legoy et al. (2) considered all external references to be suitable for their dataset and used to train their classifier, our data cleaning exercise proved crucial to remove noise and improve our model's performance.

3.1.1 Web scraping and ethical concerns

To further enhance our dataset we resorted to web scraping, a common data collection mechanism defined as any use of “technology tools for automatic extraction and organisation of data from the web”(53). Web scraping practices typically consist of the following phases(53):

1. Website analysis: using developer tools, the researcher seeks to understand the structure of a website's document object model (DOM) (54). This requires an understanding of HTML and CSS languages.
2. Website crawling: using the requests library, the researcher retrieves the HTML data. The Python library BeautifulSoup is then used for interaction with HTML and parsing structured data(54).
3. Data organisation: after scraping, the data needs to be cleaned and pre-processed so that it can be used for further analysis (53).

Following this three-step process, we scraped the MITRE website's Software page itself³. The page consists of a series of links describing malicious code, including their name, associated techniques and description. In particular, we retrieved references linked in technique descriptions as well as the associated attack. In the best case, the references referred to reports by well-established cybersecurity vendors such as Dragos, Mandiant or Trellix. In the worse case, we stumbled upon unavailable or corrupted pages which we describe in the following section. An example of a page which was scraped can be visualised in Figure 2.3. By doing so, our database was uniquely enriched with reports scraped directly from MITRE's website, guaranteeing that our tool is up-to-date with the framework.

The screenshot shows the MITRE ATT&CK Software page for Stuxnet. The page layout includes a navigation bar at the top with links to Matrices, Tactics, Techniques, Data Sources, Mitigations, and Groups. Below the navigation bar, the left sidebar lists various software entries under the 'SOFTWARE' category, with 'Stuxnet' highlighted. The main content area displays the 'Stuxnet' entry, which includes a detailed description of the malware, its ID (S0603), associated software (W32.Stuxnet), type (MALWARE), platforms (Windows), version (1.1), creation date (14 December 2020), and last modified date (20 May 2022). Below this, there is a section for 'Associated Software Descriptions' with a table listing the associated software (W32.Stuxnet) and its description. Finally, there is a section for 'Techniques Used' with a table listing the techniques (Access Token Manipulation: Token Impersonation/Theft and Account Discovery: Local Account) and their use in the Stuxnet attack.

Name	Description
W32.Stuxnet	[1]

Domain	ID	Name	Use
Enterprise	T1134	Access Token Manipulation: Token Impersonation/Theft	Stuxnet attempts to impersonate an anonymous token to enumerate bindings in the service control manager.[1]
Enterprise	T1087	Account Discovery: Local Account	Stuxnet enumerates user accounts of the local host.[1]

Figure 3.5: MITRE ATT&CK's page for the Stuxnet attack from which additional data was scraped.

An important consideration when undertaking any web scraping exercise is to abide by ethical and legal principles. Indeed while web scraping is a popular technique used by researchers and practitioners alike, there is a legal gap between the rapid rise in the use of scraping techniques and the existence of legal guidelines(55). As a result, this gray area (53) has so far been guided by related laws such as GDPR or the 1988 Copyright, Designs and Patents Act⁴, which act as temporary legal frameworks.

³For MITRE's software page see:
<https://attack.mitre.org/software/>

⁴For the Copyright Act, see:
<https://www.gov.uk/government/publications/copyright-acts-and-related-laws>

Concerned with the ethical implications of our project, we developed our data collection strategy by following the set of best practices developed by the Office for National Statistics⁵ and the European Union’s Statistical Office (Eurostat)⁶. In particular, we took care to limit the number of queries to the absolute necessary (56), ensure that the MITRE organisation allowed scraping and emphasised throughout the process that the content retrieved belonged solely to the MITRE company (56). Beyond the framework itself, this notably includes descriptions of attacks, tactics and techniques displayed on our web app described in Chapter 5. A more detailed explanation of ethical considerations can be found in Section 6.4 of the conclusion.

3.1.2 Text Processing:

Data collected from MITRE’s Github repository and its Software page had to be processed in order to retrieve textual data from their URLs. To do so, we parsed documents, presented either in HTML or PDF format, using Python’s library BeautifulSoup. This process is a “non-trivial task” (34) as it is difficult to extract relevant text from a variety of unstructured, non-standardized formats. For HTML pages, difficulties included having to guard against issues when a server could not find a requested link or when websites which did not contain CTI reports. For PDF, we used the PyPDF package to extract the relevant information. As some PDFs were very large (more than 50 pages for some), we experimented with using only the first few paragraphs but quickly found that we could not guarantee that these contained the necessary TTP description.

By merging all three datasets (Legoy et al.’s, MITRE’s Github repository and the scraped dataset), we were left with an enhance dataset of 2,376 reports. This dataset was characterised by some duplicates which we removed by turning to similarity measures. In this context, we are more concerned with lexical similarity i.e., with words that have a similar sequence of terms, rather than semantic similarity i.e., words which denote similar things or are opposite of each other (57). As such, we used term-based measures which measure the similarity — or distance — between two texts (57). Popular term-based similarity metrics include Manhattan and Euclidean distance and Jaccard similarity. Here, we used the **cosine similarity** a clustering algorithm widely used in NLP (see (58), (59)) which measures the pair wise distance between sets of vectors, irrespective of their magnitude (60). More formally:

$$\cos(\theta(u, v)) = \frac{|u \cdot v|}{|u||v|} \quad (3.1)$$

Where $|u|$ and $|v|$ are the length of two vectors u and v and $\theta(u, v)$ is the cosine of the angle between these two vectors (60). The denominator thus performs the product

⁵For the ONS, see:

<https://www.ons.gov.uk/aboutus/transparencyandgovernance/datastrategy/datapolicies/webscrapingpolicy>: :text=Pol

⁶For Eurostat guidelines, see:

<https://ec.europa.eu/eurostat/cros/content/>

of the length of u and v . Where the cosine similarity was above a threshold of 90% in our dataset, we decided to remove the data points.

3.2 Distribution of the distilled and enhanced dataset:

After preprocessing our dataset, we are left with a total of **2,229 reports** in our enhanced dataset. The average length of a report is about 21,542 words, which is well beyond the threshold of 5000 words which researchers consider when examining long document classification tasks. The unusually large length of our text data will inevitably affect our classifiers' performance, as discussed in the following chapter. Figures 3.6 and 3.7 illustrate the distribution of reports per tactic and technique respectively.

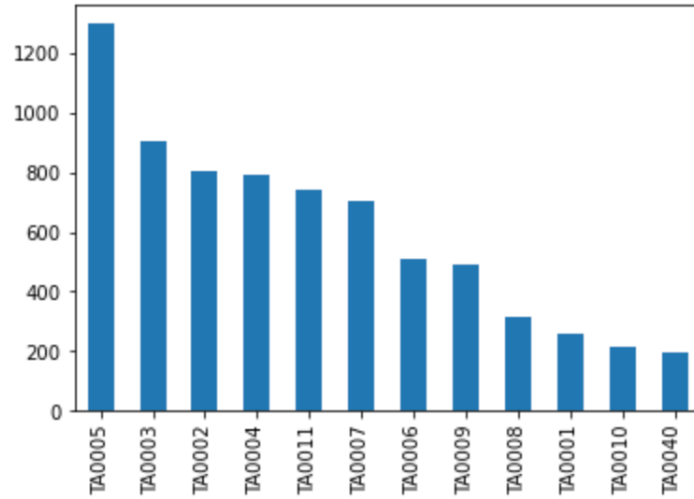


Figure 3.6: Distribution of reports per tactic in the enhanced dataset

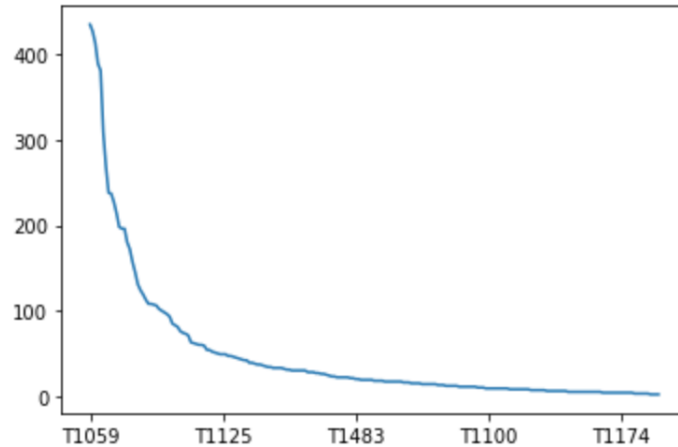


Figure 3.7: Distribution of reports per technique in the enhanced dataset

We observe that the distribution roughly follows the same pattern as that of Legoy et al.'s dataset and that we are subject to the same class imbalance, particularly in the case of technique classification. In both datasets, the tactics and techniques with the largest (tactic TA005 Defense Evasion; technique T1105 Ingress Tool Transfer) and fewest (tactic TA0010 Exfiltration) number of reports are the same. This can be explained by the fact that those tactics are over-represented and contain a large number of either techniques (in the case of tactics) or sub-techniques (in the case of techniques) - with tactic TA005 referencing 42 techniques and 115 subtechniques against 9 techniques and 8 subtechniques for tactic TA0010. As class imbalance was most visible in the case of techniques, we followed Legoy et al.'s advice and merged all subtechniques together in order to increase the number of reports per technique. We mitigated against risks of bias present in Legoy et al.'s work by collecting more data as advised by (14), but in our context, class imbalance is unfortunately inevitable. Indeed, some tactics and techniques are simply more descriptive or are used more often than others and are therefore frequently referenced in CTI reports. We thus followed Legoy et al.'s strategy by establishing a threshold for the minimum number of reports per tactic and technique to ensure that we have a sound training and testing ratio. This meant that we included all tactics used by Legoy et al. and omitted any techniques with less than 4 reports, leaving us with 12 tactics and 215 techniques.

Last, we were concerned with the language variety present in our reports. Figure 3.8 shows how the vocabulary for each report is distributed for tactics TA0011 Command and Control and TA0005 Defense Evasion using GloVe's word embedding. Yellow data points indicate that a report mentions the tactic at hand whereas a blue data point indicates that it does not. Moreover, reports whose vocabulary are similar are clustered together. For tactic TA0011, we can therefore observe a small cluster towards the highest end of spectrum whereas the vocabulary for tactic TA0005 is sparse, likely due to the large number of reports referencing this tactic.

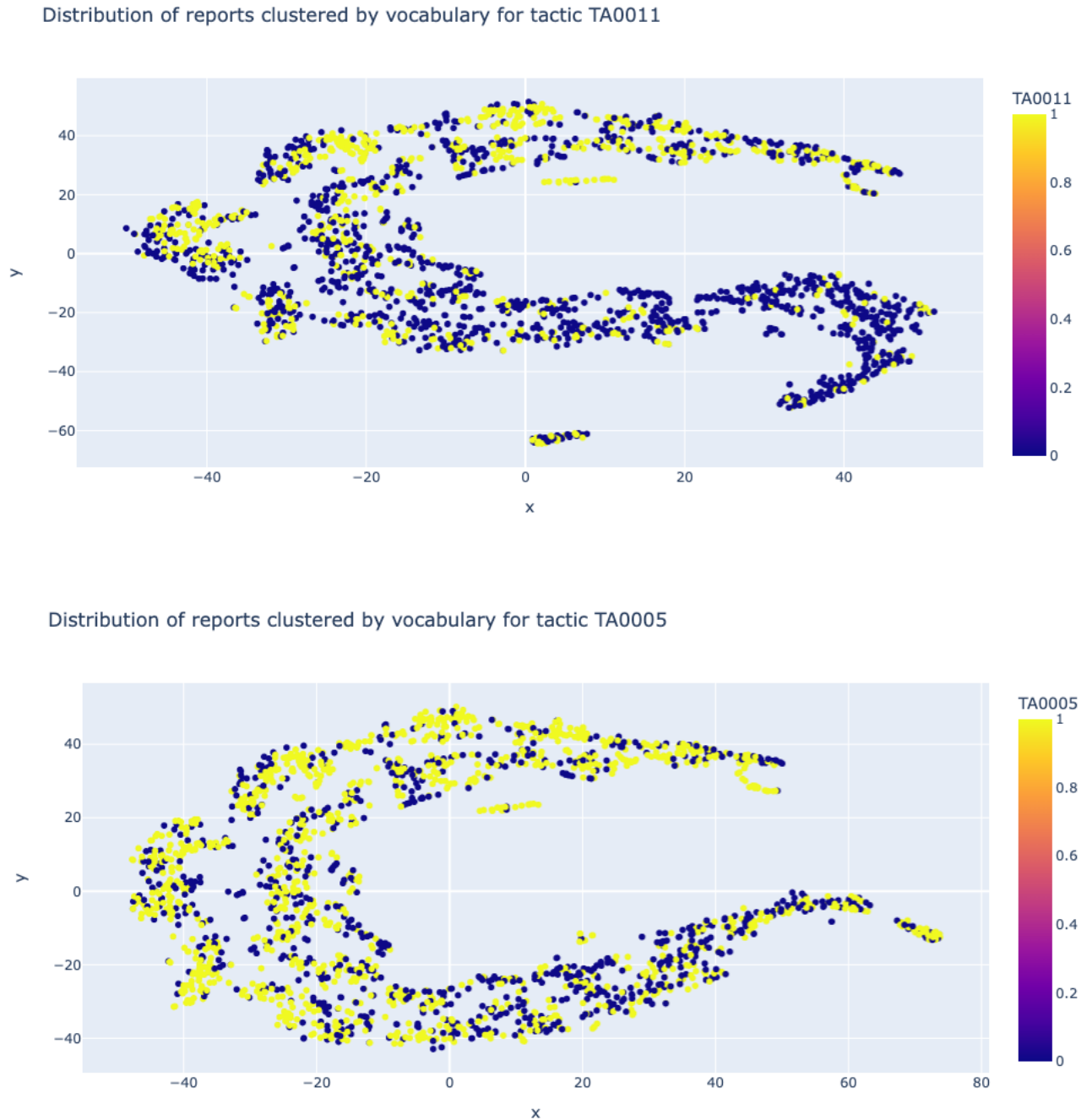


Figure 3.8: Distribution of reports clustered by vocabulary for tactic TA0011 and tactic TA0005. A yellow data point indicates that a report mentions the tactic; a blue data point indicates that it doesn't.

Chapter 4

Experiments:

In this chapter, we describe the experiments undertaken in order to build our text classification system. In light of related work, we build our experiments on Legoy et al.'s rcATT, who tested several standard linear and non-linear classifiers using both binary relevance and classifier chain approaches. Finally, we extend their work by experimenting with neural network-based models on our enhanced dataset.

A summary of results comparing our experiments with that of Legoy et al. can be found in Table 4.1.

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactic model						
rcATT						
SVC	65.54%	64.69%	65.38%	60.26%	58.50%	59.47%
2aCTI						
MLP	75.19%	62.46%	72.25%	76.69%	57.70%	70.87%
Technique model						
rcATT						
SVC	37.18%	29.79%	35.02%	28.84%	22.67%	25.06%
2aCTI						
SVC	54.48%	40.48%	50.96%	52.39%	36.40%	45.29%

Table 4.1: Summary of best performing classifiers for rcATT and the experiments undertaken in this Chapter (2aCTI). In all cases, TF-IDF is used as the text representation technique.

4.1 Methodology Overview:

For the purpose of comparison, the first approach adopted by this thesis was to conduct similar experiments as those conducted by Legoy et al. Here, we experimented only with those classifiers which Legoy et al. found yielded the best performance.

These include standard linear classifiers (Logistic Regression, Naive Bayes Classifier and Support Vector Machine) and non-linear classifiers (Decision Trees, AdaBoost Decision Tree and k-NN). Regarding text representation techniques, we began by implementing the three techniques used by rcATT: Count Vectoriser, TF-IDF and word2vec.

Second, and similarly to Legoy et al., this thesis had to tackle the key challenge of a multi-class, multi-label classification task - namely the fact that the number of label sets grows exponentially as the number of class labels increases (20). To cope with the issue of “exponential-sized output space” (20), we followed Legoy et al. and implemented two classification strategies: binary relevance and classifier chain. These strategies - whose logic can be visualised in Figures 4.1 and 4.2 - aim to facilitate the learning process from multi-label data by either exploiting or ignoring altogether the correlation amongst labels (20).

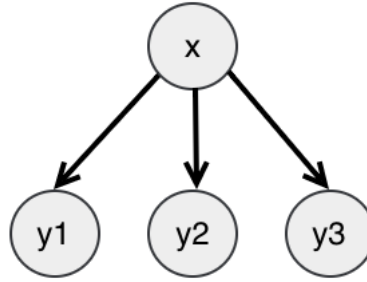


Figure 4.1: Binary Relevance (BR)

1. **Binary Relevance:** Binary relevance (BR) is perhaps the most intuitive solution to multi-label classification problems and has attracted significant research (61). It works through *problem transformation* (62), that is, by decomposing a given classification problem into a set of independent tasks, in order to learn one binary classifier per label (61). By independent, we mean that each decomposed binary classifier learns by ignoring all other class labels (61), making binary relevance a “first-order approach” (61)(20). Formally, for each class label C_k a binary training set D_i is derived so that each training example (x, y) is transformed into a binary set (x^i, y^i) such that:

$$D_i = (x^i, y_j^i) \mid 1 \leq i \leq m \quad (4.1)$$

Aside from its conceptual simplicity, an advantage of a binary relevance approach is its ability to scale to different single-label models (62) such as Support Vector Machines or Perceptrons which we implement below. At the same time, researchers have argued that by ignoring label correlation, binary relevance may lead to sub-optimal outcomes and fail to improve generalisation (20) (61). Researchers (Godbole and Sarawagi (63), Ji et al. (64)) have therefore often overlooked this approach

on the basis that the information loss affects the system's predictive performance (62). In the context of our project, using binary relevance means that any relationship between tactics and techniques (with a technique necessarily part of one or more tactic) is overlooked.

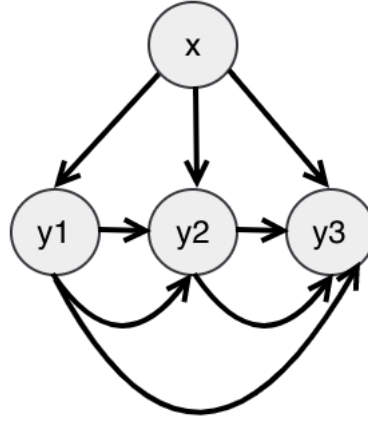


Figure 4.2: A Classifier Chain (CC)

1. *Classifier Chain or Higher-order strategy:*

A classifier chain (CC) approach extends the binary relevance logic but provides correlations between labels. In particular, a chain $h = (h_1, h_2, \dots, h_n)$ is built by deriving for each class label the prediction of the classifier preceding it in the chain (61). Starting at h_1 , the chain predicts the relevance of label h_j given the prediction and label information of all previous binary classifiers from $h_{(j-1)}$ in the chain (62). As such, the information loss resulting from a binary relevance approach is counterweighted by the passing of information down the chain.

With the exception of the k Nearest Neighbour algorithm and Multi-Layer Perceptrons, all of these models were tested using both a binary relevance and classifier chain approach. Last, the scope of Legoy et al.'s work was extended to experiment with neural network-based classifiers (transformers, multi-layer perceptrons) and text representation techniques (GloVe, doc2vec).

4.2 Hypothesis:

Before proceeding to our experiments, the following pre-testing hypothesis are formulated:

- **H.1 - A classifier chain approach should overperform compared to binary relevance:** Given the strong relationship between tactics and techniques in MITRE's framework, we suspect that an approach which leverages the characteristic of our label space will give our classifiers more predictive power (62).

- **H.2 - Simpler classification models will perform better than neural networks-based models:** Following Occam’s razor, we anticipate that a complex neural architecture and word embeddings will underperform as they are more sensitive to hyperparameter fluctuations (65).
- **H.3 - Pre-trained feature extraction algorithms should underperform due to the specificity of the cybersecurity vocabulary:** Given that CTI reports include domain-specific words which may not be found in public corpuses such as Google News or Wikipedia used to train word embedding algorithms, we expect that a trained word embedding (on a cybersecurity-related corpus) will perform better than its pre-trained version.

4.3 Baseline:

To assess our classifiers’ performance and identify unsuccessful models, we begin by defining a baseline model. Typically, researchers use a Naïve Bayes classifier which uses the probability of observing an outcome given some predictor values (24). Instead, our system can draw on the performance of Legoy et al.’s rcATT to evaluate how different classifiers compare. Tables 4.1 and 4.2 show the performance for tactics and techniques classification using three different text representation methods: Count Vectorizer, TF-IDF and word2vec. The authors also trained their classifiers using either a classifier chain (CC) or binary relevance (BR).

Surprisingly, and contra to our first hypothesis, Legoy et al. find that the classification using a binary relevance (BR) performs better than classifier chains, implying that “the relationship between labels did not have as much impact as expected” (2). Focusing on the macro-average $F_{0.5}$ score, the authors conclude that **binary relevance with a TF-IDF text representation and a SVM classifier** yields the best performance for both tactics and techniques, **with macro $f_{0.5}$ scores of respectively 59.47% for tactic classification and 25.06% for technique classification**. In the following sections we seek to improve this performance.

4.4 Preprocessing:

Preprocessing is the first step in the text classification pipeline. While research has confirmed that both feature extraction (66) and classifier selection (67) have a significant impact on classification performance, the preprocessing step is typically approached as a standardised procedure: Tokenisation, stemming and lemmatisation are all implemented without a close examination of each method’s contribution (45). Legoy et al., for instance, used both stemming and lemmatisation on their dataset and cleaned their dataset from non-word and computer science related terms¹. Instead, we test and evaluate different approaches to adopt the most effective prepro-

¹For their preprocessing, see:
<https://github.com/vlegoy/rcATT/blob/master/classification>

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR Logistic Regression	63.71%	54.42%	61.54%	58.74%	47.81%	55.76%
BR SVC	64.70%	51.55%	61.51%	59.20%	44.36%	54.89%
BR AdaBoost DT	62.30%	49.27%	59.08%	57.26%	42.37%	52.91%
CC SVC	63.81%	51.02%	60.71%	58.89%	44.31%	54.64%
CC Logistic Regression	63.81%	54.35%	61.61%	58.86%	47.78%	55.85%
CC AdaBoost DT	64.73%	50.84%	61.30%	60.87%	45.20%	56.45%
TF-IDF						
BR AdaBoost DT	61.02%	51.02%	58.61%	56.61%	44.67%	53.19%
BR Logistic Regression	71.04%	50.70%	65.61%	59.00%	40.53%	51.21%
BR Perceptron	65.20%	55.35%	62.80%	60.54%	48.29%	56.24%
BR SVC	65.54%	64.69%	65.38%	60.26%	58.50%	59.47%
CC SVC	71.63%	44.89%	63.41%	65.70%	36.76%	54.59%
CC AdaBoost DT	61.42%	49.86%	58.59%	57.71%	44.09%	53.79%
word2vec average						
CC AdaBoost DT	58.59%	44.21%	54.98%	52.69%	35.95%	47.34%
CC Logistic Regression	62.80%	34.15%	53.78%	55.27%	26.87%	42.63%
CC SVC	62.88%	40.82%	56.75%	59.98%	34.04%	49.66%
BR AdaBoost DT	57.97%	46.24%	55.11%	50.77%	38.54%	46.70%
BR Logistic Regression	66.86%	41.97%	59.68%	61.02%	31.95%	46.17%
BR SVC	64.85%	44.91%	59.49%	56.57%	35.35%	47.81%

Table 4.2: rcATT’s classification results for **tactics**. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Decision Tree (DT).

cessing strategy.

First, given the nature of our dataset and the frailty of meaning that cybersecurity concepts can have, some common preprocessing steps such as lowercase conversion were not implemented. Indeed, malware names or APT groups such as *REvil* or *MuddyWater* are case-sensitive and their syntax can be leveraged for the purpose of identification. Instead, we used the spaCy library to tokenise our data.

Regarding word normalisation, we turned to two different techniques: stemming and lemmatisation (68). In the literature, the latter is usually preferred because it removes or replaces the suffix of a word altogether, reducing a word to its “most basic word form” (68). At the same time, researchers have found that normalisation can have a neutral and even negative impact on English text classification (46). This was confirmed when testing both a Naïve Bayes classifier and SVM with TF-IDF embedding for tactic classification, with stemming performing much worse than when no word normalisation or lemmatisation were used (as seen in Table 4.3). In the case of lemmatisation, the conclusion was less clear-cut and we implemented some

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR SVC	22.87%	19.22%	21.98%	15.36%	11.39%	13.41%
BR Decision Tree	23.06%	18.53%	21.94%	18%	14.76%	16.18%
BR AdaBoost DT	35.60%	16.04%	28.56%	18.23%	9.74%	14.23%
CC Decision Tree	22.44%	17.94%	21.31%	18.64%	14.98%	16.46%
CC AdaBoost DT	36.64%	13.27%	27.05%	18.38%	8.98%	13.72%
TF-IDF						
Logistic Regression	52.82%	3.66%	14.33%	7.86%	2.76%	5.18%
BR Perceptron	30.45%	18.26%	26.82%	21.89%	14.61%	18.32%
BR SVC	37.18%	29.79%	35.02%	28.84%	22.67%	25.06%
BR AdaBoost DT	35.04%	14.77%	27.41%	17.23%	9.05%	13.36%
BR Decision Tree	20.72%	18.31%	20.15%	16.88%	14.57%	15.55%
CC Decision Tree	23.18%	18.47%	22.02%	18.83%	14.85%	16.77%
CC AdaBoost DT	37.06%	13.06%	26.98%	17.70%	8.44%	13.19%
word2vec average						
CC AdaBoost DT	29.88%	9.29%	20.67%	9.35%	4.05%	6.70%
CC SVC	36.21%	3.33%	11.70%	7.58%	3.60%	5.35%
BR AdaBoost DT	26.88%	11.79%	21.36%	9.15%	4.82%	7.31%
BR Perceptron	17.43%	15.88%	17.038%	8.33%	7.80%	6.37%
BR SVC	49.01%	6.33%	20.70%	8.57%	3.27%	5.64%

Table 4.3: rcATT's classification results for **techniques**. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Decision Tree (DT).

further tests using both TF-IDF and Count Vectoriser for tactic and technique classification. The results in Table 4.4 indicate that word normalisation has a negative effect on our classifier. As such, we conducted our experiments without stemming or lemmatising our dataset, in line with other text classification experiments (69).

	Micro $F_{0.5}$	Macro $F_{0.5}$
Naïve Bayes Classifier		
Baseline	39%	28%
Lemmatisation	39%	27%
Stemming	27%	26%
SVM		
Baseline	69%	66%
Lemmatisation	69%	65%
Stemming	17%	3%

Table 4.4: Performance of Naïve Bayes classifier and SVM with TF-IDF using different word normalisation approaches for tactic classification.

	Micro $F_{0.5}$	Macro $F_{0.5}$
Tactic Classification		
With lemmatisation		
Count Vectorizer	34%	33%
TF-IDF	39%	28%
Without lemmatisation		
Count Vectorizer	37%	38%
TF-IDF	40%	28%
Technique Classification		
With lemmatisation		
Count Vectorizer	14%	10%
TF-IDF	6%	0.8%
Without lemmatisation		
Count Vectorizer	15%	11%
TF-IDF	7%	10%

Table 4.5: Performance of a Naïve Bayes classifier with Count Vectorizer and TF-IDF using both a lemmatised and non-lemmatised dataset for tactic and technique classification.

4.5 Feature Extraction:

Different methods to represent text and retrieve features were tested in order to achieve two concomitant objectives: a) reduce the number of features present in our dataset to optimise computational resources and b) improve the performance of the classifiers tested (70). To do so, we began by following Legoy et al.'s strategy and implemented basic feature extraction methods such as Count Vectorizer, Term Frequency-Inverse Document Frequency (TF-IDF) and word2vec (2). We also followed their intuition that the number of features could have an impact on the classifier and experimented with the maximum and minimum frequency of feature appearance in our dataset (2). After several tests, we found that increasing the threshold for the minimum frequency of features was correlated with a consistent

decrease in classifier performance and settled for a broad scope with a minimum frequency of 2% and maximum frequency of 99%.

Regarding word embeddings, we implemented two algorithms: word2vec and doc2vec described in Chapter 2. Of these, Legoy et al. only tested word2vec which they trained on their training set comprised of CTI reports, ignoring instead all pre-trained version due to performance issues (2). A clear advantage of doing so is that it ensures that the algorithm is trained specifically on data which includes cybersecurity-related terms which may otherwise not be present in pre-trained models such as Google News's version. Yet despite its promising logic, the authors found that word2vec consistently underperformed compared to a Count Vectorizer or TF-IDF weighting system (2).

Drawing on Legoy et al.'s idea, we decided to compare the effect of using either Google News's pre-trained model and our own enhanced word2vec trained model. However, instead of simply turning to our training set, we scraped MITRE's tactic and technique matrices to retrieve every tactic ID with its associated name, description and technique used (see Figure 4.3). We then used the description provided by MITRE alongside the training set to train both our word2vec and doc2vec algorithms. This ensures that the algorithms are trained on vocabulary referenced by MITRE directly for classification.

The screenshot shows the MITRE ATT&CK web interface. The left sidebar lists various tactics under 'TACTICS', with 'Reconnaissance' selected. The main content area displays the 'Reconnaissance' tactic page, which includes a description and a table of associated techniques.

Reconnaissance
The adversary is trying to gather information they can use to plan future operations.

Reconnaissance consists of techniques that involve adversaries actively or passively gathering information that can be used to support targeting. Such information may include details of the victim organization, infrastructure, or staff/personnel. This information can be leveraged by the adversary to aid in other phases of the adversary lifecycle, such as using gathered information to plan and execute Initial Access, to scope and prioritize post-compromise objectives, or to drive and lead further Reconnaissance efforts.

Techniques

ID	Name	Description
T1595	Active Scanning	Adversaries may execute active reconnaissance scans to gather information that can be used during targeting. Active scans are those where the adversary probes victim infrastructure via network traffic, as opposed to other forms of reconnaissance that do not involve direct interaction.
.001	Scanning IP Blocks	Adversaries may scan victim IP blocks to gather information that can be used during targeting. Public IP addresses may be allocated to organizations by block, or a range of sequential addresses.
.002	Vulnerability Scanning	Adversaries may scan victims for vulnerabilities that can be used during targeting. Vulnerability scans typically check if the configuration of a target host/application (ex: software and version) potentially aligns with the target of a specific exploit the adversary may seek to use.
.003	Wordlist Scanning	Adversaries may iteratively probe infrastructure using brute-forcing and crawling techniques. While this technique employs similar methods to Brute Force, its goal is the identification of content and infrastructure rather than the discovery of valid credentials. Wordlists used in these scans may contain generic, commonly used names and file extensions or terms specific to a particular software. Adversaries may also create custom, target-specific wordlists using data gathered from other Reconnaissance techniques (ex: Gather Victim Org Information, or Search Victim-Owned Websites).

Figure 4.3: MITRE ATT&CK page for Reconnaissance tactic with its associated techniques and description which was scraped to train our word2vec algorithm.

Finally, we further extended Legoy et al.'s work by implementing two other word embedding techniques for feature learning trained on Wikipedia articles: GloVe and FastText. The former is similar to the word2vec algorithm in that the training of each token is based on the words surrounding it (19). To implement GloVe, we used the word embedding vectorization with 100 dimensions. FastText, on the other hand, represents each word as the sum of the representation for each character in order to capture words that do not appear in the training set (14). For instance, given a token

t , FastText will represent t using a bag of characters n -gram so that if $t = \text{highjacks}$ and $n = 3$, then FastText will represent t as (14):

$$\langle hi, hig, igh, ghj, hja, jac, ack, cks, ks \rangle \quad (4.2)$$

In our context, FastText was used to train transformers models by using cybersecurity documents found on Flair, a popular text embedding library.

4.6 Classification Models

The classifiers tested are distinguished between linear classifiers, non-linear classifiers and neural network-based classifiers. Most machine learning models were implemented through Python’s Scikit-learn library(71). The dataset was split into a training and test set using a 80:20 split. Each text sample was vectorised using a feature extraction approach described above. We also fine-tuned the hyperparameters for each classifier and sought reduced the number of features. The performance of our classifiers are reported in Tables 4.5 to 4.10. For the purpose of visualisation, we did not include classifiers that significantly underperformed compared to the baseline model.

4.6.1 Linear Classifiers:

We began by implementing linear classification models, effective in providing clear boundaries for classification (72). At a high-level, linear classification models seek to define a hypothesis H to demarcate data points using a linear decision boundary (73) such that:

$$h = x \rightarrow \text{sign}(w \cdot x + b) \quad (4.3)$$

Where given an input x and $h \in H$ labels all points categorized as one class are labeled as positive and all others as negative (73).

Of these, the **multinomial Naïve Bayes classifier** is perhaps the simplest to implement and is often used as a baseline algorithm in text classification task (14). The classifier draws on Bayes’s theorem to estimate the joint probability for each feature in a document d to belong to each class c independently (14) drawing a linear decision boundary. The classifier is described in equation 4.3.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \quad (4.4)$$

The term naïve is used here to refer to the fact that the classifier can still yield satisfactory performance when the independence assumption is violated (74). In our case, this is important as domain-specific terms are often non-independent and appear frequently in the dataset such as “Public IP”, “Email address” or “MITRE ATT&CK”. This is further complicated in the case of large documents as informative

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR Naive Bayes ($\alpha = 0$)	69.51%	27.96%	5.58%	75.94%	27.20%	51.58%
BR Naive Bayes ($\alpha = 1$)	40.74%	20.52%	34.03%	66.45%	25.45%	33.62%
BR SVM	66.32%	64.26%	64.95%	65.90%	60.78%	62.03%
BR Logistic Regression	62.53%	64.02%	62.83%	58.56%	60.59%	58.87%
CC SVM	66.35%	63.53%	65.76%	62.46%	60.13%	61.90%
CC Logistic Regression	64.95%	63.04%	64.56%	61.61%	59.94%	61.16%
TF-IDF						
BR Naive Bayes ($\alpha = 0$)	34.67%	89.45%	39.51%	32.61%	83.94%	37.02%
BR Naive Bayes ($\alpha = 1$)	35.72%	68.92%	39.53%	39.73%	51.07%	28.03%
BR SVM	70.48%	65.41%	69.40%	66.80%	63.00%	65.88%
BR Logistic Regression	76.8%	39.24%	64.46%	81.32%	29.42%	49.31%
CC SVM	64.04%	64.51%	64.13%	60.30%	62.78%	60.54%
CC Logistic Regression	70.03%	38.02%	59.93%	70.40%	30.30%	49.88%
GloVe						
BR SVM	43.18%	49.34%	33.72%	40.06%	40.28%	47.15%
BR Logistic Regression	49.49%	40.14%	47.29%	43.35%	33.92%	40.57%
CC Logistic Regression	44.20%	49.39%	34.40%	40.14%	41.28%	47.22%
CC SVM	39.93%	41.52%	58.55%	57.48%	41.57%	43.96%
word2vec (pre-trained)						
BR SVC	45.68%	50.30%	60.43%	61.32%	48.56%	52.17%
BR Logistic Regression	50.93%	55.93%	46.61%	50.12%	49.75%	54.66%
CC SVM	46.71%	50.66%	60.68%	61.32%	48.47%	51.97%
CC Logistic Regression	51.08%	56.07%	45.12%	49.79%	49.63%	54.69%
word2vec (trained)						
BR SVM	41.97%	45%	58.66%	59.36%	43.72%	47.29%
BR Logistic Regression	55.22%	58.59%	36.24%	43.74%	48.25%	54.87%
CC SVM	42.01%	44.92%	59.43%	60.00%	43.83%	47.31%
CC Logistic Regression	54.27%	58.12%	35.40%	43%	47.33%	54.30%
doc2vec (trained)						
BR SVM	46.66%	45.46%	57.20%	55.43%	45.96%	48.18%
BR Logistic Regression	65.22%	38.18%	49.16%	57.37%	33.29%	49.16%
CC SVM	44.38%	56.82%	46.41%	43.04%	59.29%	44.21%
CC Logistic Regression	59.87%	38.43%	53.86%	54.91%	34.83%	48.71%

Table 4.6: Classification results for **tactics** prediction for linear classifiers. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Support Vector Machine (SVM). For word2vec, *pre-trained* means trained using Google News’s corpus while *trained* means trained using cybersecurity-related vocabulary.

features for categorization tend to be hidden in a myriad of noisy terms (74). As

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR Naïve Bayes ($\alpha = 0$)	27.14%	14.89%	23.31%	22.01%	13.48%	17.33%
BR Naïve Bayes ($\alpha = 1$)	13.98%	23.23%	15.19%	14.80%	17.00%	11.60%
BR SVM	38.36%	35.65%	37.79%	28.27%	26.97%	26.63%
BR Logistic Regression	34.65%	38.24%	35.31%	27.35%	31.39%	26.53%
CC SVM	38.23%	35.71%	37.70%	28.88%	27.01%	26.97%
CC Logistic Regression	36.58%	38.64%	36.97%	28.26%	31.90%	27.30%
TF-IDF						
BR Naïve Bayes ($\alpha = 0$)	13.1%	0.2%	0.6%	0.1%	0.3%	0.7%
BR Naïve Bayes ($\alpha = 1$)	0.7%	40.42%	0.9%	0.2%	13.06%	37.02%
BR SVM	54.48%	40.48%	50.96%	52.39%	36.40%	45.29%
CC SVM	45.47%	33.23%	42.35%	45.58%	32.09%	39.57%
GloVe						
BR SVM	9.8%	41.4%	11.6%	8.1%	30.27%	9.2%
BR Logistic Regression	93.47%	2.47%	11.18%	7.68%	1.03%	2.87%
CC SVM	9.5%	43.87%	11.33%	7.9%	31.15%	9.07%
CC Logistic Regression	00.47%	33.23%	42.35%	45.58%	32.09%	39.57%
word2vec (pre-trained)						
BR SVM	18.57%	18.75%	36.93%	47.72%	19.09%	21.34%
BR Logistic Regression	17.89%	21.77%	22.94%	25.64%	17.77%	22.45%
CC SVM	18.49%	18.66%	36.53%	47.26%	18.99%	21.23%
CC Logistic Regression	19.30%	22.42%	22.40%	25.01%	18.69%	22.89%
word2vec (trained)						
BR SVM	31.69%	41.80%	13.66%	8.67%	27.50%	9.40%

Table 4.7: Classification results for **technique** prediction for linear classifiers. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Support Vector Machine (SVM). For word2vec, *pre-trained* means trained using Google News’s corpus while *trained* means trained using cybersecurity-related vocabulary.

such, the more the conditional independence assumption holds true, the better the data can be linearly separable and thus the better the performance of the naïve classifier (69). To mitigate against the problem of “rough parameters estimation” identified by recent research on Naïve Bayes text classification (74), we implemented a multinomial model with Count Vectorizer and TF-IDF - where each word occurrence in a document is treated as an independent token - and tested hyperparameter tuning by turning to Laplace smoothing which is used to handle cases of zero probability in Naïve Bayes.

As can be seen in Table 4.5 and 4.6, the use of Laplace smoothing significantly worsened our classifier performance when used with Count Vectorizer or TF-IDF, in line with previous research (69). This suggests that the conditional independence

assumption in our dataset somewhat holds true (69). Regarding the model's performance without smoothing, we find that in the case of tactic and technique classification, TF-IDF underperforms compared to Count Vectorizer but yields low results, likely due to the fact that our test set is hard to linearly separate.

Next, we considered **multinomial logistic regression** as it typically works well for predicting categorical outcomes (19) and relies on fewer assumptions than other linear model models such as Linear Discriminant Analysis (LDA) (75). The classifier aims to separate classes in the training set by learning how important a given feature is for determining whether an input belongs to a given class or not before predicting a probability distribution over all classes (14). It does so by learning a variable z representing a vector of weights and biases to an input vector X from the training set (69) such that:

$$z = (w_1, w_2 \dots w_j) \cdot (x_1, x_2 \dots x_j) + b \quad (4.5)$$

As each feature in the dataset is assigned a weight in the vector w , we implemented dimensionality reduction using PCA to our logistic regression model to reduce its computational complexity to 75 dimensions. Looking at Tables 4.5 and 4.6, we observe that a Classifier Chain approach using Count Vectoriser yields the best performance for both tactic and technique classification. In particular, the performance for tactic classification improves by roughly 6 points rcATT's performance with the same classifier and feature extraction method. For technique classification, the performance of the logistic regression was more disappointing, although the classifier chain with Count Vectoriser still outperformed Legoy et al.'s experiment. Finally, looking at word embedding techniques, we notice satisfactory results in the use of the word2vec algorithm for tactic classification, with the pre-trained and trained versions yielding similar performances.

The last linear classifier implemented is the multi-class **support vector machine**, which Legoy et al. found yielded the best performance on their dataset when combined with a TF-IDF feature extraction. The aim of a SVM classifier is to find an optimal hyperplane to separate classes in the dataset (14) given a margin p known as the maximum-margin hyperplane (73). The effectiveness of SVM classifiers lies in their ability to learn a linear threshold function regardless of the feature space dimension (76) which makes it an effective algorithm when dealing with sparse datasets (77). The ability to linearly separate a dataset characterised by a high presence of feature may explain why SVM performed well in Legoy et al.'s experiments (76).

In our experiments, we found that the SVM classifier outperformed all other linear classification models implemented using a binary relevance approach with TF-IDF. Compared to rcATT's best performing model, our classifier improves the f -score by roughly 7 and 20 points for tactic (macro- $f_{0.5}$ score: 65.88%) and technique (macro- $f_{0.5}$ score: 45.29%) classification respectively.

Finally regarding word embeddings, we find that the word2vec algorithm outperforms GloVe. Doc2vec, on the other hand, yields disappointing results. Indeed, while Le and Mikolov (28) reported promising results when testing doc2vec on a sentiment analysis task, others have subsequently failed to find similar results casting doubts over the effectiveness of the algorithm (78). In 2016, Lau and Baldwin suggested that using pre-trained models and large corporas could improve the performance of doc2vec (78). This could also serve to explain why the pre-trained version of word2vec performs surprisingly better than the trained version, thus contradicting our third hypothesis which held that pre-trained feature extraction algorithm would underperformed compared to their trained version.

4.6.2 Non-linear classifiers:

Tree-based and boosting methods have been widely adopted by research in text classification due to their conceptual simplicity and robust performance (75). Here, we experimented both with multi-label decision tree classifier and ensemble methods which combine several predictors (73) for performance gain as well as multi-label k-Nearest Neighbour algorithm.

We observe that in general multi-label decision trees outperformed the AdaBoost algorithm for both tactic and technique classification, with the exception of binary relevance TF-IDF in the tactic model. For background, multi-label decision trees seek to maximize the information gain by identifying the optimal feature and splitting value using an inductive learning approach (i.e., learning from examples) (20). On the other hand, the ensemble method AdaBoost algorithm is designed to adaptively select a new classifier at each “boosting round” by drawing on the performance of the previous classifier (73). In doing so, the performance of AdaBoost is designed to exponentially increase as a function of the number of boosting rounds (73). While our decision tree classifier outperforms that of Legoy et al. for technique prediction (tactic performance was not provided), researchers have recognised that tree-based classifiers can easily overfit which may explain why boosting does not increase performance (19) in this case.

With the exception of the Count Vectorizer for tactic classification, the classifier chain underperforms across all models, so our attention turned to the binary relevance approach. Looking at tactic classification, the Decision Tree using the trained word2vec algorithm stands out with a f -score of roughly 62%. This improves rcATT’s best performing model for tactic classification (which was a classifier chain AdaBoost Decision Tree with count vectorizer text representation) by roughly 6 points. In the case of technique classification, we found that a decision tree using count vectorizer performs best and improves on rcATT by around 15 points.

Last, a **multi-label k-Nearest Neighbour** algorithm was implemented. The non-parametric model is designed to find the k nearest neighbours (here, $k=3$) of a text input x before scoring each class and assigning x to the class with the highest score

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR DT	55.13%	58.32%	53.50%	55.84%	54.65%	57.81%
BR AdaBoost DT	65.1%	66.57%	53.38%	56.66%	61.9%	64.32%
BR Multi-label kNN	54.06%	40.80%	50.76%	49.04%	33.21%	43.46%
CC DT	54.79%	57.97%	51.48%	53.47%	53.97%	57.01%
CC AdaBoost DT	65.1%	66.57%	53.38%	56.66%	61.90%	64.32%
TF-IDF						
BR DT	50.62%	54.72%	48%	51.18%	49.93%	53.97%
BR AdaBoost DT	64.12%	49.55%	60.56%	62.36%	44.34%	56.86%
BR Multi-label kNN	59.83%	41.04%	54.81%	52.84%	36.57%	47.49%
CC DT	50.16%	54.38%	46.64%	51.18%	49.27%	53.71%
CC AdaBoost DT	62.36%	64.12%	44.34%	49.55%	56.86%	60.56%
GloVe						
BR DT	43.05%	52.43%	24.29%	31.64%	35.91%	46.34%
BR AdaBoost DT	52.43%	31.64%	46.34%	43.05%	24.29%	35.91%
CC DT	37.02%	38.51%	37.31%	31.07%	32.50%	31.28%
CC AdaBoost DT	41.67%	52.04%	24.17%	31.15%	34.61%	45.89%
word2vec (pre-trained)						
BR DT	36.63%	41.27%	41.24%	44.48%	37.26%	41.87%
BR AdaBoost DT	52.28%	58.11%	33.33%	41.29%	44.99%	53.73%
CC DT	33.14%	38.29%	35.61%	38.92%	33.29%	38.41%
CC AdaBoost DT	51.36%	58.08%	34.05%	41.70%	44.47%	53.85%
word2vec (trained)						
BR DT	66.32%	64.26%	65.90%	62.47%	60.78%	62.03%
BR AdaBoost DT	56.55%	33.85%	49.86%	53.00%	26.52%	39.78%
CC DT	39.96%	41.01%	40.17%	33.27%	33.88%	33.33%
CC DT AdaBoost	52.46%	58.65%	29.67%	36.30%	43.94%	52.22%
Doc2vec (trained)						
BR DT	38.63%	41.53%	39.17%	33.92%	41.53%	34.39%
BR AdaBoost DT	58.25%	40.39%	53.52%	51.86%	34.20%	46.29%
CC DT	38.18%	41.37%	38.78%	33.46%	37.65%	34.06%
CC AdaBoost DT	59.47%	38.75%	53.72%	55.32%	34.94%	48.74%

Table 4.8: Classification results for **tactics** prediction for non-linear classifiers. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Decision Tree (DT).

(19). In our experiment, the kNN algorithm yields the same performance as that of rcATT and fails to outperform tree-based models. This may be due to the fact that the algorithm's performance is largely data dependent and under-performs in cases where the number of classes is large as it relies on finding a meaningful distance function between x and k (19).

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
BR DT	32.45%	31.85%	32.33%	32.08%	31.21%	29.74%
BR AdaBoost DT	38.36%	35.65%	37.79%	28.27%	26.97%	26.63%
CC DT	32.25%	29.78%	31.96%	31.08%	28.67%	28.71%
TF-IDF						
BR DT	28.92%	31.26%	23.39%	24.61%	26.15%	29.66%
BR AdaBoost DT	00.92%	31.26%	23.39%	24.61%	26.15%	29.66%
CC DT	32.04%	30.31%	23.29%	23.11%	28.04%	28.53%
CC AdaBoost DT	000.04%	30.31%	23.29%	23.11%	28.04%	28.53%
word2vec (pre-trained)						
BR DT	0.59%	10.54%	10.84%	16.27%	0.63%	11.34%
BR AdaBoost DT	17.06%	35.97%	0.58%	10.69%	11.34%	24.42%
CC AdaBoost DT	14.24%	39.70%	0.40%	0.62%	0.85%	19.10%
word2vec (trained)						
BR DT AdaBoost	29.37%	0.7%	18.59%	10.02%	3.16%	6.35%
CC DT AdaBoost	26.72%	0.33%	11.12%	0.6%	0.13%	0.30%

Table 4.9: Classification results for **technique** prediction for non-linear classifiers. Abbreviations: Classifier Chain (CC), Binary Relevance (BR), Decision Tree (DT).

4.6.3 Neural Networks

While long discarded by research on natural language processing, recent efforts focused on the use of word embeddings and deep learning methods have made neural networks “the default approach” for many text classification tasks (25). Indeed, in the context of cybersecurity text classification and information extraction, researchers have increasingly turned to neural networks to collect Indicators of Compromise from cybersecurity articles (39) or extract intelligence from hacker forums (79). Considering these developments and the promising results yielded by the linear and non-linear classifiers above, we thus decided to extend Legoy et al.’s research and experiment with neural network-based architectures. The experiments included multilayer perceptrons (MLP) and transformers. To optimise the hyperparameter search, a Grid Search was implemented to generate all candidates considered. The results of the Grid Search are reported in Tables 4.9 and 4.10.

At first glance, transformers under-perform in the context of tactic classification and therefore efforts were focused on optimizing multi layer perceptrons. A MLP is a feed-forward neural network consisting of an input, output and one or more hidden layer. It learns by back-propagation algorithm which calculates a gradient descent used for training (80). All experiments use a ReLu (Rectified Linear Units) activation function as it has sparse property which mitigates against over-fitting problems in text classification tasks (80).

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
FastText						
Transformers	41.36%	19.38%	26.39%	21.96%	12.91%	15.47%
Count Vectorizer						
MLP [100]	72.12%	62.63%	70.05%	71.80%	57.97%	67.79%
MLP [200]	73.04%	62.71%	70.71%	73.22%	58.88%	69.42%
MLP [300]	70.47%	65.00%	69.31%	66.912%	61.63%	65.23%
MLP [1000, 200]	65.75%	64.51%	65.49%	66.647%	62.41%	64.94%
TF-IDF						
MLP [100]	75.19%	62.46%	72.25%	76.69%	57.70%	70.87%
MLP [200]	74.42%	64.02%	72.08%	75.05%	59.32%	70.41%
MLP [300]	70.47%	65%	69.31%	66.91%	61.63%	65.23%
MLP [1000, 200]	76.45%	60.01%	72.48%	76.60%	56.11%	70.47%

Table 4.10: Classification results for **tactic** prediction for neural network-based models classifiers. Abbreviations: Multi-layer perceptron (MLP). The brackets indicate how many layers and neurons were implemented. For instance, MLP [100, 100] indicates two layers with 100 neurons each. All neural networks used a reLu activation function.

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Count Vectorizer						
MLP [100]	57.28%	28.92%	47.89%	42.85%	23.34%	34.01%
MLP [200]	62.71%	31.33%	52.25%	49.59%	27.64%	39.29%
MLP [1000, 200]	64.91%	34.905%	55.39%	52.05%	290.31%	42.39%
MLP [1000, 300]	55.19%	36.63%	50.11%	48.40%	31.18%	40.44%
TF-IDF						
MLP [100]	77.22%	20.47%	49.67%	41.84%	14.13%	27.51%
MLP [200]	75.17%	24.38%	53.06%	47.73%	19.52%	33.80%
MLP [1000, 200]	75.88%	22.25%	51.20%	48.66%	18.49%	33.32%

Table 4.11: Classification results for **technique** prediction for neural network-based models classifiers. Abbreviations: Multi-layer perceptron (MLP). The brackets indicate how many layers and neurons were implemented. For instance, MLP [100, 100] indicates two layers with 100 neurons each. All neural networks used a reLu activation function.

To our surprise, multi-layer perceptrons outperformed many linear classifiers. In the case of tactic classification, we noticed that increasing the number of layers somewhat decreased the performance of the classifier and we therefore focused on changing the number of neurons in the single hidden layer. In doing so, we found that for tactic classification, a neural network with one hidden layer and 100 neuron and TF-IDF feature extraction stands out compared to other models. Results for

technique classification were less robust and performance increased as the number of layers was augmented. The best architecture for technique classification is a neural network with two hidden layers, the first with 1000 neurons and the second with 200 using a Count Vectoriser text representation. Overall, the performance for technique classification using a neural network is closer to that of the best linear model (SVM with TF-IDF) than expected.

These results illustrate how neural networks can be powerful architectures for text classification even when handling sparse and noisy data. For tactic classification, our experiment **outperformed rcATT's best classifier for tactic classification by 11 points**.

4.7 Preliminary Conclusion and Choice of Model

First, looking back on Tables 4.5 to 4.10, we notice that there is a significant degree of variance between the performance of tactic compared to that of technique classification, consistent with that experienced by related research (2). This is primarily caused by class imbalance which particularly affects the technique model: Indeed, while technique T1220 XSL Script Processing only references 11 reports, technique T1105 Ingress Tool Transfer links to more than 400 reports, meaning that T1120 is less likely to be picked up by the classifier. While we sought to mitigate against the imbalance problem by collecting more labelled data as described in Chapter 3, we recognise that our efforts could not completely mend this issue.

What the presence of imbalance unveils, however, is that certain classifiers are more sensitive to the problem than others. Indeed, we noticed that for technique classification, the Support Vector Machine significantly outperforms all other classifiers, while tree-based models appear to be more sensitive to imbalance. These findings echo research on the effect of class imbalance on classification performance which found that a negative effect is conditional on both the size of the training set and the classifier used (81). Classifiers found to be less sensitive to the problem of data imbalance include Support Vector Machine and Multilayer Perceptrons (81), which happen to be our best performing model for technique and tactic classification respectively.

Second, reflecting on our pre-testing hypothesis, we can further conclude that:

1. ***H.1: Binary relevance performed better than a classifier chain approach:***
In line with Legoy et al.'s findings, and contra to our initial intuition, we found that with the exception of logistic regression, adopting a binary relevance approach leads to better performance than a classifier chain. This can be explained by the fact that binary relevance is more resistant to overfitting because it does not expect samples to be associated with the preceding prediction in the chain (62).

2. **H.2: In the context of tactic performance, more complex neural network architectures outperformed simpler linear models:** Our experiment show that neural network-based models can perform well for text classification tasks. In our case, a simpler architecture with one hidden layer stood out amongst our tests, which may be a result of the quality of our datasets. The performance nonetheless signals promising results for future experiments.

Regarding word embeddings, it is harder to derive conclusive evidence for the third hypothesis which expected trained embeddings to perform better than pre-trained ones. First, we observe that the word2vec embedding consistently performed better than GloVe. At the same time, while word2vec's pre-trained version stood out for linear classifiers, it is the trained version that performed best in the case of non-linear classifiers. This is likely a consequence of the limited corpus size used when training the embedding on. Likewise, the disappointing performance of doc2vec is likely a result of a small corpus size. We suggest improving the trained embeddings in section 6.3 of our conclusion.

Finally, we can conclude that for both tactic and technique classification TF-IDF outperforms other feature extraction methods. For tactic classification, a multi layer perceptron with one hidden layer and 100 neurons stands out compared to other classifiers with a $F_{0.5}$ of 70.87%. Regarding technique classification, we found similarly to Legoy et al. that SVM is the best classifier with a $F_{0.5}$ of 45.29%. Taken together, **our experiments improve the performance of rcATT by 11 and 20 points respectively for tactic and technique classification.**

Chapter 5

Software Implementation:

This chapter presents 2aCTI¹, an interactive web application to host the classifier tested in Chapter 4 whose name stands for *Automated Analysis of Cyber Threat Intelligence*. Based on the experiments above, the system was built using a TF-IDF text representation with a binary relevance approach (2). For classification of TTPs the system draws on two models, one for tactic classification and one for technique classification. The system is built with Vue.js for the front-end and Python at the back-end using FastApi. We used Heroku to host both front- and back-end. A complete picture of our pipeline is illustrated in Figure 5.1.

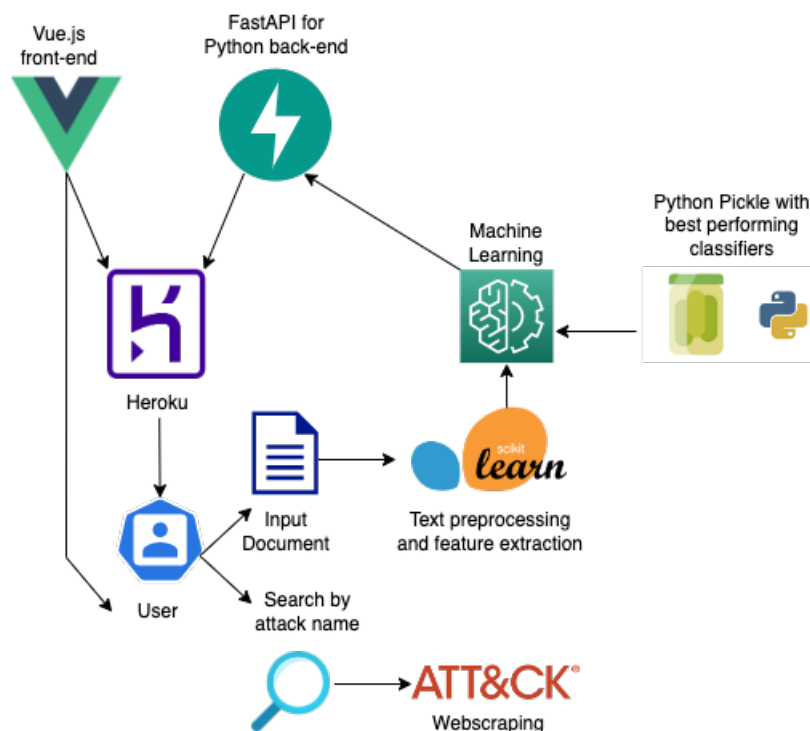


Figure 5.1: Pipeline diagram

¹For 2aCTI, see:
<https://cti-analysis.herokuapp.com/>

5.1 Building blocks - user perspective:

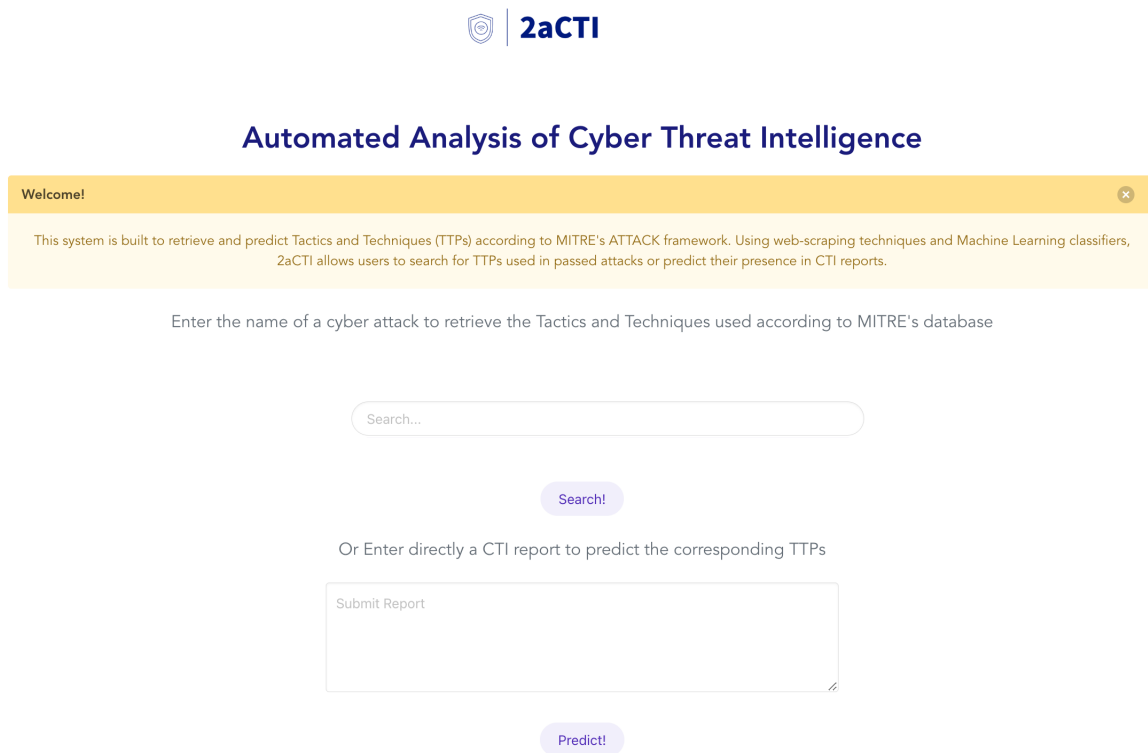
2aCTI is composed of two elements to satisfy differing user needs.

1. **Scenario 1 - Malware search:** The first component of the API is a search bar. From a malware entered by the user, the API scrapes the MITRE Software page to check whether the malware has already been classified by the organisation. If it has, the API displays a description of the malware as well as the techniques used by malicious actors during the malware. The purpose of this functionality is to reduce the user's search time by searching directly into MITRE's database. If an malware is not present in the database, a message prompts the user to turn to the second scenario.
2. **Scenario 2 - Document classification:** The second and main element of the API is a text field to classify a report and predict its corresponding TTPs. This is the central component of our web app because it most closely resembles Legoy et al.'s rcATT and because it is based on our findings from Chapter 3. Given an input report by the user, the system begins by performing the relevant tokenisation and feature extraction (in this case, TF-IDF) on the text. Then, having saved our best performing classifiers during the testing stage as pickles - a method used to serialise and deserialise python objects - the system applies the appropriate model for tactic and technique classification. The API returns the predicted tactic and techniques present in the dataset as well as its associated confidence score (macro average F -score).

Extending Legoy et al.'s work, 2aCTI also outputs the five most relevant sentences found in the report. These sentences are useful in determining the elements which contributed the most to the overall classification. To identify these segments, the API evaluates the performance of the tactic model when given each sentence individually. When all sentences of the report have been tested, the API retrieves all sentences with a maximum probability of predicting the tactics of at least 30%. If more than 5 sentences are extracted, the API outputs the 5 sentences with the highest probability of predicting the tactics.

In both cases, the visual output is roughly the same: tactics or techniques with a positive prediction are displayed in green; those with a negative prediction in red or yellow. We settled for a threshold of 50% for the macro average $F_{0.5}$ score to determine whether a tactic qualifies as a positive prediction or not. In both cases, a description of the tactic or technique at hand is displayed through click-events.

The main functionalities of 2aCTI can be visualised in Figures 5.2 and 5.3.



2aCTI

Automated Analysis of Cyber Threat Intelligence

Welcome!

This system is built to retrieve and predict Tactics and Techniques (TTPs) according to MITRE's ATTACK framework. Using web-scraping techniques and Machine Learning classifiers, 2aCTI allows users to search for TTPs used in passed attacks or predict their presence in CTI reports.

Enter the name of a cyber attack to retrieve the Tactics and Techniques used according to MITRE's database

Search...

Search!

Or Enter directly a CTI report to predict the corresponding TTPs

Submit Report

Predict!

Figure 5.2: 2aCTI's homepage

5.2 Project Architecture:

Our pipeline starts with our API built using FastAPI, a framework used for developing Python-based applications similar to flask². The API is composed of two URLs used in our front-end depending on the user's need, as described in the above section. The "search attack" URL calls a function to perform the scraping and satisfy the first scenario. By contrast, the "classification" URL uses the three Python pickles which have been serialised based on our findings in Chapter 4: one is used to perform feature extraction on an input document using TF-IDF, one to perform tactic classification and a third to perform technique classification. When a request is sent from the front-end in this second scenario, the API deserialises the pickles to extract TTPs from a given document provided by the user. All three of these pickles were developed in our experiments stage which can be downloaded from our Github repository.

Importantly, the deployment of our API to Heroku proved particularly challenging due to our resource requirements: First, the three pickles used were too large for Github, which meant that the entire deployment had to pass through Heroku's Command Line Interface. Second, while our TF-IDF and tactic model could be deployed on the cloud platform, the technique model using the Support Vector Machine con-

²For FastAPI see:
<https://fastapi.tiangolo.com/>



Figure 5.3: Example of a search result for the Stuxnet attack

stantly surpassed the memory quota provided. We considered upgrading our memory limit by scaling the number of containers — called “dynos” on Heroku — but this would have meant facing a fee of up to 250£. Ultimately, we decided to test several alternative classifiers to decrease the size of our technique pickle. Similarly to the tactic model, we settled for a multi-layer perceptron with one hidden layer and 50 neurons. The technique classifier used in 2aCTI — whose performance can be seen in Table 5.1 — is therefore much less performant than the one tested in the above Chapter. If our system is reproduced, future work should keep this structural

	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactic						
MLP [100]	75.19%	62.46%	72.25%	76.69%	57.70%	70.87%
Technique						
MLP	78.15%	18.51%	47.53%	36.96%	11.34%	24%

Table 5.1: 2aCTI's performance for tactic and technique classification using TF-IDF text representation.

constraint in mind and, ideally, seek to deploy 2aCTI using the best performing classifier.

Last, our front-end was developed from scratch using Vue.js, a JavaScript framework. On top of Vue.js we used Buefy, a user-interface component library based on the Bulma framework. Other notable elements of our front-end include event-based interactivity and the ability for users to chose their preferred request mode. To intercept HTTP requests and responses in our front-end, we turned to the promise-based Node.js client axios³.

5.3 Software Evaluation

2aCTI was tested to evaluate the software contribution of this thesis.

Testing the search functionality is quite straightforward as the output maps those found on MITRE's Software page. One minor limitation of this functionality is that MITRE only classified malware names per technique and not tactic. We tested some mapping functions to ensure that if a technique is classified by MITRE, then the corresponding tactic would also be displayed. One issue with this logic, however, is that as a technique can be part of more than one tactic, we could not guarantee with a sufficient level of reliability that the tactic classification was correct. We therefore settled for the classification of techniques when searching cyber attacks by malware name and not tactics.

Regarding the prediction of TTPs from CTI, several reports were tested to evaluate how the prediction model performed. In line with the issue of memory quota on Heroku described above, it was quickly evident that the technique performance was lower than that achieved in our experiments. While we could not find a way to resolve this issue other than increasing the number of Dynos on Heroku, we sought to leverage the characteristic of the Enterprise framework itself to take advantage of the tactic-technique relationship.

³For axios and node.js see:
<https://axios-http.com/docs/intro>

To this end, we take a cue from the poor performance yielded by the classifier chain approach in the experiments to devise a new means of linking tactics and techniques. This took the form of a hanging node algorithm, which was implemented following Legoy et al.'s results. The hanging node approach leverages the tactic-technique relationship to ensure that in the case in which a technique is predicted with a confidence score close to a threshold th ⁴ but its corresponding tactic is not predicted, then the technique can be removed from the prediction (2). Similarly, if a tactic is not predicted but has a relatively high confidence score (macro f -score of 40%, for instance) *and* has a technique predicted with a high degree of probability (macro f -score of more than 50%), then the tactic is added to the prediction, even if its confidence score is technically below the defined threshold.

One advantage of this approach is that it uses the confidence score of the tactic classification - which is higher than that of technique classification due to the reduced number of labels to be predicted for tactics - as an indicator of the likelihood for a technique to be predicted. It also ensures that there can never be a case in which a technique is predicted but a tactic isn't.

Following the implementation of the hanging node algorithm, 2aCTI was tested using a CTI report on the 2010 Stuxnet worm. The choice of this malware is motivated by the fact that it was previously classified by the MITRE organisation, providing a baseline for evaluation. MITRE identified the following techniques used by Stuxnet under the tactics: Initial Access, Persistence, Privilege Escalation, Defence Evasion, Discovery (selected by number of techniques predicted). Given a report (which can be found under "stuxnettext.txt" on Github) 2aCTI is able to predict the tactics Initial Access (macro f -score 54%; 2 techniques identified by MITRE), Persistence (macro f -score 48%; 5 techniques identified by MITRE) and Discovery (macro f -score 75.53%; 9 techniques identified by MITRE). It fails to predict, however, the tactics Privilege escalation (7 techniques identified by MITRE) and Defence Evasion (11 techniques identified by MITRE). It also fails to predict any techniques with a confidence score higher than the defined threshold. Several reasons can serve to explain these results: the report given may not be completely descriptive of the Stuxnet attack and may fail to mention some crucial elements for classification. It may also be that the performance of the technique classification in 2aCTI reduces the probability of a tactic being classified following the hanging node algorithm. These points serve to feed areas of improvements described below.

⁴For ease of comparison, we used the same classification thresholds of 0.5 as Legoy et al. (2).

Chapter 6

Conclusion

6.1 Summary of contributions:

The aim of this thesis was twofold: to understand how best to extract TTPs from CTI reports using MITRE's ATT&CK knowledge base and to create a system to facilitate the analysis of CTI reports for the cybersecurity professional.

Legoy et al.'s rcATT (2) - but also the works of Ayoade et al. (3) and Husari et al. (1) - gave the impetus to this thesis and, following their efforts, the research question expounded in the introduction was approached by turning to Machine Learning models for text classification.

To improve rcATT's performance, the first contribution this thesis sought to make was to solve the problem of data availability highlighted in much of the relevant literature. With access to Legoy et al.'s dataset, this thesis turned to web-scraping and preprocessing techniques for data enhancement. In doing so, we were able to increase the number of labelled sample in our dataset by roughly 800 labelled samples which proved crucial when comparing our classifier performance to that of Legoy et al.

The second contribution of this thesis is the breadth of its analysis and the scope of experiments conducted in Chapter 4. While related work, in our view, has tended to focus on more simpler models and word embedding techniques, we have tested and evaluated both linear, non-linear and neural network-based approaches and found promising results for the use of more complex models for text classification tasks. We have extended existing work by experimenting with various feature extraction techniques such as GloVe, word2vec and doc2vec, in both their pre-trained and trained version on cybersecurity vocabulary.

Third, with the exception of rcATT, few works reviewed have developed a tool to facilitate CTI classification as we have. The aim when designing 2aCTI was to provide a visually compelling matrix akin to that of MITRE's Enterprise framework. By using an intuitive design and providing descriptions for all tactics and techniques,

we believe that 2aCTI grants the cybersecurity researcher with an accessible and detailed overview of tactics and techniques either used by a specific malware or mentioned in a CTI report. Last, using FastAPI to develop our Python back-end and Vue.js for our front-end, we designed the entire software pipeline and web app from scratch which allowed us to design each feature with the purpose of achieving our intended aim.

Finally, we achieved the aim of this thesis defined at the outset by improving the performance of CTI classification. Specifically compared to rcATT, 2aCTI improved the classification of tactics by 11 points by testing with multi-layer perceptrons and of techniques by 20 points by using TF-IDF with a Support Vector Machine. The lessons learned for future text classification tasks of CTI reports are thus the following:

1. The data collection step of any task should not be taken lightly and, whenever possible, researchers should seek to collect more labelled data. We give significant credit to our data collection strategy for improving our classifier's performance.
2. The MITRE ATT&CK matrix is a good framework to build classifiers around but suffers from inevitable imbalance due to the large number of techniques.
3. The scope of analysis should not be limited to simple models as we have seen that more complex neural network-based models can outperform linear classifiers.
4. It is more important to train word embeddings on larger corpuses than on domain-specific vocabulary.

These points bring us to the areas of improvement which this thesis could have benefited from.

6.2 Limitations:

Limitations of this thesis fall into one of two issue categories: those regarding the system's implementation with its possible improvement using state-of-the art language models and those that are inherent to the text classification task at hand due to the nature of the ATT&CK framework.

Starting with the first, a number of revisions could improve our work. First, the side effect of using web scraping techniques for data collection is that it inevitably creates noise in the dataset. Indeed, as each page scraped is structured differently, we had to devise a general scraping strategy rather than treating each web page individually. Had we not been constrained by time, this thesis would have benefited from a more rigorous data cleaning process. We believe that while time-consuming, this could have a significant effect on classifier performance.

Next, similarly to rcATT, our dataset suffers from imbalance which meant that certain

tactics or techniques could not be included in our prediction models. In particular, the technique model had to absorb sub-technique classification due to an imbalanced ratio between the number of techniques (385 subtechniques divided in 191 techniques) and the number of reports available. While related work had to adopt a similar strategy, this could mean that our system may not provide the same level of granularity as granted by the Enterprise framework.

A further limitation of our experiments is that while a grid search was implemented for neural network-based models, hyperparameter tuning for other models was ultimately a manual endeavour. A more thorough analysis to find the best fit for each model could have been done. Likewise, the implementation of more complex deep learning models could have improved the performance of our tactic classifiers.

Next, regarding the second issue category, it should be noted that as the ATT&CK framework is “based on real-world observations” (82), it is bound to undergo some modifications. New tactics and techniques as well as frameworks are continually added to guarantee ATT&CK’s relevance. The Disarm Foundation, for instance, is currently creating a framework for disinformation which follows ATT&CK’s structure¹. While it is true that our web scraping efforts ensure that the data collected at the time of this thesis is up to date, this cannot be guaranteed in the future. Last, the MITRE organisation admits that the ATT&CK cannot “enumerate all possibilities for the types of actions and behaviors documented as part of its adversary mode” (see Disclaimers in Appendix B) and therefore they may be some undisclosed TTPs not taken into account by our system.

Taking these limitations into account, future improvements could cover the following:

6.3 Future work

First, by leveraging on the dataset provided in this project’s repository, future work could start by seeking to correct data imbalance to improve the classification performance. This could be achieved by collecting more labelled data, either by additional web-scraping or — perhaps more interestingly — through data augmentation. While popular in computer vision, strategies to artificially increase the diversity of a training set such as backtranslation (which consists in translating a document into a foreign language before translating it back into the original language) have recently been explored in the field of Natural Language Processing (83). By doing so, other state-of-the art language models such as BERT or ELMo could be tested. Additionally, the promising results from our neural network-based experiments, notably for tactic classification, should encourage future research to apply more complex architectures such as recurrent and convolutional neural network as well as deep learning models.

¹<https://www.disarm.foundation/>

Second, it would be valuable to gain access to a larger corpus of CTI reports to train the word embeddings on. This suggestion for improvement takes a cue from Lau and Baldwin(78)'s review of doc2vec which found that the algorithm yielded strong results when trained over a large corpus. Indeed, we believe that the reason why our trained embeddings did not perform as expected is because of limited data for training.

Third, future research could focus specifically on testing new avenues for long document classification. One route could be to apply recent work which has drawn inspiration from audio segmentation used for speech recognition to classify lengthy legal documents (65). In their experiments, the authors suggested segmenting large documents and testing the relevance of each segment to the overall classification (65). Instead of turning to doc2vec which operates at the document-level, the result of each segment embedding is then combined to build a single embedding (65). Similar strategies to improve the performance of long documents applied to our classification task would make an important contribution to the field.

Finally as an important future contribution, future research could answer recent calls by Mandiant Threat Intelligence, jointly with the MITRE organisation, to push for the integration of both Enterprise and Industrial Control Systems (ICS) matrices provided by MITRE (84). The justification behind this call is that the boundary between IT and ICS is increasingly fading, with threat actors moving across bound technology domains more freely (84). Doing so would grant the cybersecurity analyst with a more holistic vision for the analysis of new adversary behaviours. While we aspired to fulfill this aim at the inception of this thesis, we were ultimately constrained by data availability but given the threat posed by ICS attacks, MITRE's ICS framework will most likely expand in the next few years. Future work should therefore take care to collect ICS-related data before embarking on this task. We have provided an initial dataset and web scraping code on our Github to assist in this effort.

6.4 Practical, ethical and professional considerations

We conclude this thesis by reviewing the ethical and legal considerations which have guided this thesis.

This project relies heavily of MITRE's ATT&CK open source data. The framework is freely available to be modified and enhanced, as we have done in gathering more data and distilling its content². While our dataset suffers from imbalance - with some tactics or techniques more frequently used and thus referenced in more reports - we took advantage of MITRE's license to obtain more labelled samples and mitigate, as much as possible, against risks of bias induced by imbalance. Overall, any commer-

²For the terms of use, see
<https://attack.mitre.org/resources/terms-of-use/>

ical, research or development use of MITRE's ATT&CK data is authorised, provided that its license is reproduced, as we have done (see Appendix B).

Having said this, this thesis faces important ethical considerations regarding the **ethics and legality of web scraping practices** which are still considered a “grey area” (53) in the field. To our knowledge, there exist few legislature that addresses the issue of web scraping specifically. Examples include the policy paper published in March 2021 by the Office for National Statistics which defines the contours of web scraping for the purpose of data collection and analysis³. Likewise, in the European Union, Eurostat devised a policy draft outlining good practices for webscraping⁴. Principles of ethical webscraping include but are not limited to:

- Minimising the number of requests and burden on website owners
- Only scraping websites where it is legal
- Respecting the ‘robots.txt’ robots exclusion protocols
- Limiting scraping for the purpose of creating new value to the data (56)
- Always follow the terms of use and any copyright documents on website

With these principles in mind, this thesis took care to ensure that we only sent requests to the MITRE ATT&CK website and any external links when necessary. We have also emphasised that this project is heavily reliant on the MITRE ATT&CK organisation to avoid any copyright issues.

In line with the ethics checklist section 4 (see Appendix A), this project does not collect any personal data, as defined by the General Data Protection Regulation (GDPR)⁵. While it involves the merging of previously collected data, our project does not deal with any personal or sensitive information.

Regarding section 8, we understand “civilian application” to mean non-military activities in line with the European Commission's definition⁶ (69). Our project is primarily addressed to civilians, researchers - including MITRE ATT&CK collaborators - and cybersecurity professionals and there is no intention for it to be used in military applications.

Finally, with regard to professional implications, our system can be leveraged by

³For the ONS see:

<https://www.ons.gov.uk/aboutus/transparencyandgovernance/datastrategy/datapolicies/>

⁴For Eurostat see:

<https://ec.europa.eu/eurostat/cros/content/WPC>

⁵For the UK's GDPR guidelines, see

<https://ico.org.uk/for-organisations/guide-to-data-protection/guide-to-the-general-data-protection-regulation-gdpr/what-is-personal-data/what-is-personal-data/>

⁶For the EU's research guidance, see

<https://ec.europa.eu/research/participants/portal4/doc/call/h2020/h2020-bes-2015/1645164-explanatory>

cybersecurity professionals and researchers alike to identify recurring tactics and techniques. Blue teams can use our system to identify attack patterns (3); Red teams can use the information of past TTPs used against similar organisations to emulate attacker behaviours and evaluate their organisation's defence strategy (3). Last, our system can also be used as a means to update MITRE's central repository and enhance it with more threat reports.

After careful examination, we consider our project to be in line with the BCS Code of Conduct, IET Rules of Conduct and Engineering Council Statement of Ethical Principles. As our aim is to classify publicly available documents according to an open-source framework, our view is that our project does not confront any other ethical concerns. Nonetheless, we recognise that as our system relies on Machine Learning models, questions pertaining to potential job displacement may be a source of concern. As it stands, we consider our system to be of use for research purposes and, if implemented in a professional setting, used as a means to enhance the cybersecurity analysts' work, not replace it.

Appendices

Appendix A

Appendix A - Ethics Checklist:

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?	X	
Does your project involve the use of human embryos?	X	
Does your project involve the use of human foetal tissues / cells?	X	
Section 2: HUMANS		
Does your project involve human participants?	X	
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?	X	
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?	X	
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?	X	
Does it involve processing of genetic information?	X	
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.	X	
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve merging existing data sets?	X	
Section 5: ANIMALS		
Does your project involve animals?	X	
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?	X	
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?	X	
Could the situation in the country put the individuals taking part in the project at risk?	X	
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?	X	
Does your project deal with endangered fauna and/or flora /protected areas?	X	
Does your project involve the use of elements that may cause harm to humans, including project staff?	X	
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?	X	
Section 8: DUAL USE		
Does your project have the potential for military applications?	X	
Does your project have an exclusive civilian application focus?	X	
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?	X	
Does your project affect current standards in military ethics –e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?	X	
Section 9: MISUSE		
		Does your project have the potential for malevolent/criminal/terrorist abuse?
		X
		Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?
		X
		Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if misapplied?
		X
		Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related projects?
		X
		SECTION 10: LEGAL ISSUES
		Will your project use or produce software for which there are copyright licensing implications?
		X
		Will your project use or produce goods or information for which there are data protection, or other legal implications?
		X
		SECTION 11: OTHER ETHICS ISSUES
		Are there any other ethics issues that should be taken into consideration?
		X

Appendix B

Appendix B - MITRE's ATT&CK license:



[Home](#) > [Resources](#) > [Terms of Use](#)

Terms of Use

LICENSE

The MITRE Corporation (MITRE) hereby grants you a non-exclusive, royalty-free license to use ATT&CK® for research, development, and commercial purposes. Any copy you make for such purposes is authorized provided that you reproduce MITRE's copyright designation and this license in any such copy.

"© 2021 The MITRE Corporation. This work is reproduced and distributed with the permission of The MITRE Corporation."

DISCLAIMERS

MITRE does not claim ATT&CK enumerates all possibilities for the types of actions and behaviors documented as part of its adversary model and framework of techniques. Using the information contained within ATT&CK to address or cover full categories of techniques will not guarantee full defensive coverage as there may be undisclosed techniques or variations on existing techniques not documented by ATT&CK.

ALL DOCUMENTS AND THE INFORMATION CONTAINED THEREIN ARE PROVIDED ON AN "AS IS" BASIS AND THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE MITRE CORPORATION, ITS BOARD OF TRUSTEES, OFFICERS, AGENTS, AND EMPLOYEES, DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION THEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Bibliography

- [1] Husari G, Al-Shaer E, Ahmed M, Chu B, Niu X. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: Proceedings of the 33rd annual computer security applications conference; 2017. p. 103-15. pages i, 1, 4, 15, 16, 17, 19, 20, 53
- [2] Legoy V, Caselli M, Seifert C, Peter A. Automated retrieval of att&ck tactics and techniques for cyber threat reports. arXiv preprint arXiv:200414322. 2020. pages i, 1, 2, 3, 4, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 32, 35, 36, 45, 47, 52, 53
- [3] Ayoade G, Chandra S, Khan L, Hamlen K, Thuraisingham B. Automated threat report classification over multi-source data. In: 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC). IEEE; 2018. p. 236-45. pages i, 1, 2, 4, 16, 18, 19, 20, 53, 58
- [4] NCSC. Russia behind cyber attack with Europe-wide impact an hour before Ukraine invasion. May 2022. Available from: <https://www.ncsc.gov.uk/news/russia-behind-cyber-attack-with-europe-wide-impact-hour-before-ukraine-invasion>. pages 1
- [5] ATT&CK M. MITRE ATT&CK Getting Started. Available from: <https://attack.mitre.org/resources/getting-started/>. pages 2
- [6] Alexander O, Belisle M, Steele J. MITRE ATT&CK for Industrial Control Systems: Design and Philosophy. The MITRE Corporation: Bedford, MA, USA. 2020. pages 2
- [7] Orchilles J. Cyber Kill Chain, MITRE ATT&CK and Purple Team. SANS Blog. Available from: <https://www.sans.org/blog/cyber-kill-chain-mitre-attack-purple-team/>. pages 2
- [8] Basra J, Kaushik T. MITRE ATT&CK® as a Framework for Cloud Threat Investigation. Center for Long-Term Cybersecurity (CLTC): Berkeley, Italy. 2020. pages 2
- [9] Trellix. What is the MITRE ATT&CK Framework?; 2022. Available from: <https://www.trellix.com/en-us/security-awareness/cybersecurity/what-is-mitre-attack-framework.html>. pages 2, 3

-
- [10] Georgiadou A, Mouzakitis S, Askounis D. Assessing mitre att&ck risk using a cyber-security culture framework. *Sensors*. 2021;21(9):3267. pages 2, 3
- [11] Classroom C. CyCraft Classroom: MITRE ATTCK vs. Cyber Kill Chain vs. Diamond Model; 2022. Available from: <https://medium.com/cycraft/cycraft-classroom-mitre-att-ck-vs-cyber-kill-chain-vs-diamond-model-1cc8fa49a20f>. pages 3
- [12] Centre NCS. CNI Hub;. Available from: <https://www.ncsc.gov.uk/section/private-sector-cni/cni>. pages 3
- [13] Zhu Z, Dumitras T. Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports. In: 2018 IEEE European symposium on security and privacy (EuroS&P). IEEE; 2018. p. 458-72. pages 3, 4, 15, 16, 17, 18, 19, 20
- [14] Vajjala S, Majumder B, Gupta A, Surana H. Practical natural language processing: a comprehensive guide to building real-world NLP systems. O'Reilly Media; 2020. pages 4, 8, 9, 10, 27, 36, 37, 40
- [15] House of Lords HoC. Joint Committee on the National Security Strategy of the UK's Critical National Infrastructure. Available from: <https://committees.parliament.uk/committee/111/national-security-strategy-joint-committee/publications/>. pages 4
- [16] Liao X, Yuan K, Wang X, Li Z, Xing L, Beyah R. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the 2016 ACM SIGSAC conference on computer and communications security; 2016. p. 755-66. pages 4, 15, 16, 19, 20
- [17] Ikonomakis M, Kotsiantis S, Tampakas V. Text classification using machine learning techniques. *WSEAS transactions on computers*. 2005;4(8):966-74. pages 7
- [18] Zhou ZHTbSL. Machine Learning. In: Machine learning. Springer; 2016. . pages 7
- [19] Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D. Text classification algorithms: A survey. *Information*. 2019;10(4):150. pages 7, 8, 9, 10, 11, 12, 16, 36, 40, 41, 42
- [20] Zhang ML, Zhou ZH. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*. 2013;26(8):1819-37. pages 7, 11, 30, 41
- [21] Jurafsky D, Martin JH. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition;. pages 7

- [22] Maglogiannis IG. Emerging artificial intelligence applications in computer engineering: real word ai systems with applications in ehealth, hci, information retrieval and pervasive technologies. vol. 160. Ios Press; 2007. pages 7
- [23] Bird S, Klein E, Loper E. Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."; 2009. pages 8, 9, 19
- [24] Bruce P, Bruce A, Gedeck P. Practical statistics for data scientists: 50+ essential concepts using R and Python. O'Reilly Media; 2020. pages 8, 11, 32
- [25] Eisenstein J. Introduction to natural language processing. MIT press; 2019. pages 9, 10, 43
- [26] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013. pages 10
- [27] Weeds J, Weir D. A general framework for distributional similarity. In: Proceedings of the 2003 conference on Empirical methods in natural language processing; 2003. p. 81-8. pages 10
- [28] Le Q, Mikolov T. Distributed representations of sentences and documents. In: International conference on machine learning. PMLR; 2014. p. 1188-96. pages 10, 41
- [29] Narkhede S. Understanding Confusion Matrix. Medium. 2018. Available from: <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>. pages 12
- [30] Behzadan V, Aguirre C, Bose A, Hsu W. Corpus and deep learning classifier for collection of cyber threat indicators in twitter stream. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE; 2018. p. 5002-7. pages 13
- [31] Deliu I, Leichter C, Franke K. Collecting cyber threat intelligence from hacker forums via a two-stage, hybrid process using support vector machines and latent dirichlet allocation. In: 2018 IEEE International Conference on Big Data (Big Data). IEEE; 2018. p. 5008-13. pages 13
- [32] Neil L, Mittal S, Joshi A. Mining threat intelligence about open-source projects and libraries from code repository issues and bug reports. In: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI). IEEE; 2018. p. 7-12. pages 13
- [33] Rahman MR, Mahdavi-Hezaveh R, Williams L. A Literature Review on Mining Cyberthreat Intelligence from Unstructured Texts. In: 2020 International Conference on Data Mining Workshops (ICDMW). IEEE; 2020. p. 516-25. pages 13, 14

- [34] Zhu Z, Dumitraş T. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security; 2016. p. 767-78. pages 14, 16, 17, 19, 25
- [35] Husari G, Niu X, Chu B, Al-Shaer E. Using entropy and mutual information to extract threat actions from cyber threat intelligence. In: 2018 IEEE international conference on intelligence and security informatics (ISI). IEEE; 2018. p. 1-6. pages 14, 16, 19
- [36] Satvat K, Gjomo R, Venkatakrishnan V. EXTRACTOR: Extracting attack behavior from threat reports. In: 2021 IEEE European Symposium on Security and Privacy (EuroS&P). IEEE; 2021. p. 598-615. pages 14, 16, 17, 19, 20
- [37] Ramnani RR, Shivaram K, Sengupta S. Semi-automated information extraction from unstructured threat advisories. In: Proceedings of the 10th Innovations in Software Engineering Conference; 2017. p. 181-7. pages 14, 16, 19
- [38] Lee C, Lee GG. Information gain and divergence-based feature selection for machine learning-based text categorization. Information processing & management. 2006;42(1):155-65. pages 14
- [39] Long Z, Tan L, Zhou S, He C, Liu X. Collecting indicators of compromise from unstructured text of cybersecurity articles using neural-based sequence labelling. In: 2019 international joint conference on neural networks (IJCNN). IEEE; 2019. p. 1-8. pages 15, 43
- [40] Azevedo R, Medeiros I, Bessani A. PURE: Generating quality threat intelligence by clustering and correlating OSINT. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (Trust-Com/BigDataSE). IEEE; 2019. p. 483-90. pages 15
- [41] Facebook. Facebook ThreatExchange;. Available from: <https://developers.facebook.com/programs/threatexchange/>. pages 15
- [42] Vault A. Alien Vault OTX; 2016. Available from: <https://otx.alienvault.com/>. pages 15
- [43] Liu J, Yan J, Jiang J, He Y, Wang X, Jiang Z, et al. TriCTI: an actionable cyber threat intelligence discovery system via trigger-enhanced neural network. Cybersecurity. 2022;5(1):1-16. pages 16, 18
- [44] Legoy V, Caselli M, Seifert C, Peter A. rcATT Github Repository. GitHub; 2020. <https://github.com/vlegoy/rcATT#readme>. pages 17
- [45] Uysal AK, Gunal S. The impact of preprocessing on text classification. Information processing & management. 2014;50(1):104-12. pages 17, 32

-
- [46] Toman M, Tesar R, Jezek K. Influence of normalization on text classification. *Proceedings of InSciT*. 2006;4:354-8. pages 17, 33
- [47] Pomikálek J, Rehurek R. The Influence of preprocessing parameters on text categorization. *International Journal of Industrial and Manufacturing Engineering*. 2007;1(9):504-7. pages 17
- [48] Scikit-Learn. Binary Relevance;. Available from: http://scikit.ml/api/skmultilearn.problem_transform.br.html. pages 18
- [49] Scikit-Learn. Classifier Chain;. Available from: http://scikit.ml/api/skmultilearn.problem_transform.cc.html. pages 18
- [50] Géron A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* " O'Reilly Media, Inc."; 2019. pages 18
- [51] TRICTI. GitHub; 2022. <https://github.com/lingren0/TriCTI>. pages 18
- [52] Cyber Monitor. GitHub; 2022. https://github.com/CyberMonitor/APT_CyberCriminal_Campagin_Collections. pages 18
- [53] Krotov V, Silva L. Legality and ethics of web scraping. 2018. pages 23, 24, 57
- [54] Breuss M. *Beautiful Soup: Build a Web Scraper With Python*. Available from: <https://realpython.com/beautiful-soup-web-scraper-python/>. pages 23
- [55] Brewer R, Westlake B, Hart T, Arauza O. The ethics of web crawling and web scraping in cybercrime research: Navigating issues of consent, privacy, and other potential harms associated with automated data collection. In: *Researching cybercrimes*. Springer; 2021. p. 435-56. pages 24
- [56] Densmore J. *Ethics in Web Scraping*. 2017. pages 25, 57
- [57] Gomaa WH, Fahmy AA, et al. A survey of text similarity approaches. *international journal of Computer Applications*. 2013;68(13):13-8. pages 25
- [58] Singh R, Singh S. Text similarity measures in news articles by vector space model using NLP. *Journal of The Institution of Engineers (India): Series B*. 2021;102(2):329-38. pages 25
- [59] Park K, Hong JS, Kim W. A methodology combining cosine similarity with classifier for text classification. *Applied Artificial Intelligence*. 2020;34(5):396-411. pages 25
- [60] Ravichandran D, Pantel P, Hovy E. Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*; 2005. p. 622-9. pages 25
-

- [61] Zhang ML, Li YK, Liu XY, Geng X. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*. 2018;12(2):191-202. pages 30, 31
- [62] Read J, Pfahringer B, Holmes G, Frank E. Classifier chains for multi-label classification. *Machine learning*. 2011;85(3):333-59. pages 30, 31, 45
- [63] Godbole S, Sarawagi S. Discriminative methods for multi-labeled classification. In: *Pacific-Asia conference on knowledge discovery and data mining*. Springer; 2004. p. 22-30. pages 30
- [64] Ji S, Tang L, Yu S, Ye J. Extracting shared subspace for multi-label classification. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*; 2008. p. 381-9. pages 30
- [65] Wan L, Papageorgiou G, Seddon M, Bernardoni M. Long-length legal document classification. *arXiv preprint arXiv:191206905*. 2019. pages 32, 56
- [66] Günel S, Ergin S, Gülmezoğlu MB, Gerek ÖN. On feature extraction for spam e-mail detection. In: *International Workshop on Multimedia Content Representation, Classification and Security*. Springer; 2006. p. 635-42. pages 32
- [67] Feng G, Guo J, Jing BY, Hao L. A Bayesian feature selection paradigm for text classification. *Information Processing & Management*. 2012;48(2):283-302. pages 32
- [68] Toman M, Tesar R, Jezek K. Influence of word normalization on text classification. *Proceedings of InSciT*. 2006;4:354-8. pages 33
- [69] Lim C. An Evaluation of Machine Learning Approaches to Natural Language Processing for Legal Text Classification. Imperial College London, MSc Computing thesis. 2019:1-120. pages 34, 39, 40, 57
- [70] Mladenović D, Brank J, Grobelnik M, Milic-Frayling N. Feature selection using linear classifier weights: interaction with classification models. In: *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*; 2004. p. 234-41. pages 35
- [71] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. 2011;12:2825-30. pages 37
- [72] Krishna D. A Look at the Maths Behind Linear Classification. Medium. 2020. Available from: <https://towardsdatascience.com/a-look-at-the-maths-behind-linear-classification-166e99a9e5fb>. pages 37
- [73] Mohri M, Rostamizadeh A, Talwalkar A. *Foundations of machine learning*. MIT press; 2018. pages 37, 40, 41
- [74] Kim SB, Han KS, Rim HC, Myaeng SH. Some effective techniques for naive bayes text classification. *IEEE transactions on knowledge and data engineering*. 2006;18(11):1457-66. pages 37, 38, 39

- [75] Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction. vol. 2. Springer; 2009. pages 40, 41
- [76] Joachims T. Text categorization with support vector machines: Learning with many relevant features. In: European conference on machine learning. Springer; 1998. p. 137-42. pages 40
- [77] Kamath CN, Bukhari SS, Dengel A. Comparative study between traditional machine learning and deep learning approaches for text classification. In: Proceedings of the ACM Symposium on Document Engineering 2018; 2018. p. 1-11. pages 40
- [78] Lau JH, Baldwin T. An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:160705368. 2016. pages 41, 56
- [79] Deliu I, Leichter C, Franke K. Extracting cyber threat intelligence from hacker forums: Support vector machines versus convolutional neural networks. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE; 2017. p. 3648-56. pages 43
- [80] Cui JL, Qiu S, Jiang My, Pei ZL, Lu Yn. Text classification based on ReLU activation function of SAE algorithm. In: International Symposium on Neural Networks. Springer; 2017. p. 44-50. pages 43
- [81] Japkowicz N, Stephen S. The class imbalance problem: A systematic study. Intelligent data analysis. 2002;6(5):429-49. pages 45
- [82] MITRE. MITRE ATT&CK Framework; 2022. Available from: <https://attack.mitre.org/>. pages 55
- [83] Feng SY, Gangal V, Wei J, Chandar S, Vosoughi S, Mitamura T, et al. A survey of data augmentation approaches for NLP. arXiv preprint arXiv:210503075. 2021. pages 55
- [84] Zafra D K AOBNAG Lunden K. In Pursuit of a Gestalt Visualization: Merging MITRE ATT&CK® for Enterprise and ICS to Communicate Adversary Behaviors;. Available from: <https://medium.com/mitre-attack/in-pursuit-of-a-gestalt-visualization-merging-mitre-att-ck-for-enterprise-and-ics-to-communicate-3523daa7b580>. pages 56