Retrieving ATT&CK tactics and techniques in cyber threat reports

Valentine Legoy, Andreas Peter, Christin Seifert, and Marco Caselli

Abstract— **Threat intelligence sharing has been expanding during the last few years, leading cybersecurity professionals to have access to a large amount of open data. Among those, the tactics, techniques and procedures (TTPs) related to a cyber threat are particularly valuable but generally found in unstructured textual reports, so-called Cyber Threat Reports (CTRs). In this study, we evaluate different multi-label text classification models to retrieve TTPs from textual sources, based on the ATT&CK framework – an open knowledge base of adversarial tactics and techniques. We also review several post-processing approaches using the relationship between the various labels in order to improve the classification. Our final contribution is the creation of a tool able to extract ATT&CK tactics and techniques from a CTR to a structured format, which, with more data, could reach a macro-averaged $F_{0.5}$ score of 80% for the prediction of tactics and over 27.5% for the prediction of techniques.**

I. Introduction

Cyber threat intelligence (CTI) is the collection and organisation of specific information, which can be used to prevent, detect or mitigate cyber attacks. That information can either be: tactical, providing details about an attacker's methodology or the tools and tactics employed during an attack; or technical, that is to say specific indicators used by a threat [1]. While technical threat intelligence –i.e. indicators of compromise (IOCs)– are simple to obtain, it can be more difficult to have access to tactical threat intelligence –i.e. tactics, techniques, and procedures (TTPs). Tactics are also valuable, as knowing about it helps to develop a better defence against the related threat, which means that the adversaries have to expend more resources to be able to attack [2].

Threat information sharing resources have been expanding in recent years with the development of standards to securely and structurally distribute those (e.g. STIX [3], TAXII [4], CybOX [5]). To have access to this information, it is either possible to buy private feeds from specialized companies or to obtain open threat intelligence from different sources over the internet. More companies encourage the sharing of threat information by creating their sharing platform (e.g. IBM X Force [6], OTX AlienVault [7]), containing lists of IOCs and referencing cyber threat reports (CTRs), which have more precise information about the IOCs and TTPs.

Therefore, the information provided by these means can either be structured (e.g. IP blacklists) or unstructured (e.g. CTRs). Whereas structured data can be accessed simply by using web scrapers or parsers, unstructured data require either manual analysis or natural language processing (NLP) tools, to extract the most relevant information. It also appears that open structured intelligence is usually limited to lists of IOCs. That being the case, the valuable tactical information contained in CTRs is challenging to obtain. Without proper tools, security experts currently have to read and retrieve information from CTRs manually, which is a task that requires precision and time. Considering the number of CTRs published each day (i.e. in average 15 new text reports are created and shared each day on platforms such as IBM X Force and OTX AlienVault[1]), this is a cumbersome task for

Valentine Legoy was a student at the University of Twente, 7522 NB Enschede, Netherlands

Andreas Peter is with the Services and Cyber Security Group, University of Twente, 7522 NB Enschede, Netherlands (e-mail: a.peter@utwente.nl)

Christin Seifert is with the Data Science Group, University of Twente, 7522 NB Enschede, Netherlands (e-mail: c.seifert@utwente.nl)

Marco Caselli is with Siemens Corporate Technology, 81739 Munich, Germany (e-mail: marco.caselli@siemens.com)

[1]Number estimated based on the number of reports published on IBM X Force from January 2019 to March 2019, and on the number of reports published on AlienVault OTX from January 2019 to March 2019.

humans. Thus, there is currently a need for security experts to extract tactical information from CTRs to a structured format, which they could use more efficiently.

The goal of this thesis is to create a tool able to take as input the text of a given CTR, then process it to identify TTPs, and finally output those in a machine-readable format. We use the ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework [8], as it aims to describe every technique and tactic possibly used during a threat and is often used to describe attacks. Linking a report to specific tactics and techniques from the ATT&CK framework also allows the user to define prevention, detection and mitigation methods for the threat exposed in the text more quickly. We expect the tool to be used mainly on short reports about a precise threat, and not on more in-depth descriptions, nor annual reports containing descriptions of multiple threats. We can summarise our goal in one research question: *How can we automatically retrieve ATT&CK tactics and techniques from cyber threat reports?*

In summary, the contributions of this research are as follows:

- We define a multi-label text classification model adapted to the retrieval of ATT&CK tactics and techniques in CTRs, based on the limited annotated open data available.
- We introduce a post-processing approach based on the hierarchical structure of the ATT&CK framework, that we compare to diverse methods applied to similar problems.
- We describe rcATT: a tool that predicts ATT&CK tactics and techniques in a CTR and outputs them in a structured STIX format. rcATT contains a feedback system allowing users to add more data to the training dataset, which will improve the predictions over time.

II. Related work

Even though cyber threat intelligence sharing has progressed due to the development of new tools over the last few years, the research about the automated analysis of CTRs similar to our project is limited. The extraction of security concepts can be found in several research papers, either using rules and pattern detection, looking to retrieve vulnerabilities [9], [10], information related to Undercoffer's cybersecurity ontology [11], [12], and other generic concepts [13], [14] from various unstructured sources.

Several other studies have been conducted about the extraction of indicators of compromise from threat reports. If IOCs are interesting to know, they are easier to extract than other information, using regular expressions, or parsing the list at the end of advisories. Multiple open-source tools performing this kind of tasks are currently available (i.e., 22 repositories about "IOC extractions" on Github). Several of the tool described in these research papers start by extracting candidate IOCs using regular expressions, before using different classification techniques based on the semantic context of the candidate, to remove the false positive from the results: iACE by Liao *et al.* [15], iGen by Panwar *et al.* [16], and ChainSmith by Zhu *et al.* [17]. Zhou *et al.*'s work [18] uses a different approach to the previous papers by using a neural network inspired by the research of Lampe *et al.* [19] and Dernoncourt *et al.* [20], and avoid relying on field-specific language structure, or requiring a large amount of annotated data.

The identification of adversarial tools in unstructured data has also been the focus of some papers. Samtani *et al.* [21], [22] and Deliu *et al.* [23] both use topic modelling based on Latent Dirichlet Allocation to identify various types of tools (e.g. website exploits, SQL injections), as well as IOCs categories, for Deliu *et al.* [23]. They, however, rely on manual analysis and labelling of the results from the topic modelling. While Deliu *et al.* [23] work on texts, Samtani *et al.* [21], [22] apply their process to text and source code.

Only five research papers aim to retrieve TTPs in unstructured data, as we do. Zhu *et al.* [24] focused strictly on the identification of the behaviour of malware and the Android features it uses, in the development of their tool, FeatureSmith. For a given scientific paper, using a typed-dependency parser, FeatureSmith extracts behaviours as tuples of subject-verb-object, from which subject or object could be missing. A behaviour weight is defined, based on the mutual information between the different words and the word "Android". The tool then builds a semantic network having for nodes the different malware families, behaviours and features, and linking a malware and behaviours or linking behaviours and features. These edges are weighted by the number of times the nodes appear together, within three sentences of each other. These relations allow the production of an explanation between the three elements and their importance to detect malware. Albeit extremely specific to their topic, their approach can differentiate the behaviours associated with separate malware mentioned in the same text.

Ramnani *et al.* [25] also extract TTPs from advisories but to rank them by relevance. The extraction and identification are done using patterns designed with the help of Basilisk [26], which identifies patterns surrounding keywords. Basilisk takes as input candidate patterns, which in this case were retrieved from AutoSlog TS [27], and keywords, which were identified based on their frequency in a cybersecurity corpus. A semantic similarity score is attributed between each sentence and each pattern, to define the importance of the information [28], [29]. On a test set of eighteen CTRs, Ramnani *et al.* consistently obtain an averaged recall above 80%, but they also get an irregular precision across all reports. In order to improve these results, they decided on implementing a feedback system, to evaluate the validity of their patterns.

TTPDrill [30], developed by Husari *et al.*, is one of the closest tools to what we aim to create, as it extracts threat actions from a cyber threat report, in order to link them to specific ATT&CK and CAPEC [31] techniques and tactics. Husari *et al.*'s first step is the creation of a unique ontology [32] based on ATT&CK and CAPEC, linking a Lockheed Martin's kill chain phase [33] to a threat action, composed of an object, a precondition and an intent. TTPDrill works the following way. First, a scraper retrieve threat and blog articles archive. Then, the content is pre-processed by sanitizing and selecting relevant articles through binary Support Vector Machine (SVM) classification, based on the number of words, the security-related verbs density and security-related nouns density. Using a typed-dependency parser, the tool identifies candidate threat actions, composed of a subject, a verb and an object. It, then, filters out the candidates and retrieves the ATT&CK techniques by calculating the similarity score, using the BM25 score between the candidate threat actions and the threat actions present in the ontology. The final result of a report is a set of threat actions, with their technique, tactic, and kill chain phase output to the user in the STIX format. Husari *et al.* continued the development of technologies to extract threat action by creating ActionMiner [34]. Using Part-Of-Speech tagging, ActionMiner retrieves noun phrases and verbs, then iterates through a sentence from a verb to a noun phrase with which it could form a pair, which result in a candidate threat action. These verb-object pairs are filtered based on the entropy and mutual information defined from a corpus composed of many computer science-related articles.

The most recent research on this topic, presented by Ayoade *et al.* [35], also has for precise aim to retrieve ATT&CK tactics and techniques present in a threat report. The tool starts by extracting features from a CTR as TF-IDF weighted bag-of-words. Because of the diversity of the sources – they use three different datasets for training and testing: 169 articles from the ATT&CK website, 488 diverse CTRs and 17600 Symantec reports – Ayoade *et al.* decided to use a bias correction technique called covariate shift method, for which they tested several algorithms: KMM [36], KLIEP [37], and uLSIF [38]. Then, they classify the report, first for the ATT&CK tactics, then for the ATT&CK techniques using a confidence propagation score between those two classifications. The final step is the classification of the report into kill chain phases based on rules [33].

Husari *et al.* [30], authors of TTPDrill, and Ayoade *et al.* [35] defined different methodologies for the retrieval of ATT&CK tactics and techniques in CTRs. They both present good performances: overall 94% precision and 82% recall for TTPDrill tested on 50 reports and trained on 90 reports from Symantec; and 76% accuracy on average for the tactics predictions and up to 82% accuracy on average for the techniques predictions for Ayoade *et al.*'s work, trained and tested on their three datasets. However, Ayoade *et al.*, being published after Husari *et al.*'s paper, is highly critical of the approach from TTPDrill, finding a 29% accuracy, in average, on the tactics predictions, when using their datasets. Their claims encouraged us to research a text classification approach instead of an ontology-based information retrieval method. Unfortunately, we are not able to verify the results from any of these papers, as we do not have access to their datasets or their tools. Thus, the only comparison we will be able to make with their works is on their concept, which can be found in Section X.

III. ATT&CK FRAMEWORK

The ATT&CK (Adversarial Tactics, Techniques, and Common Knowledge) framework [8] is central in our work as it defines the different techniques and tactics we want to retrieve in CTRs. The main incentive to using this framework is the fact that it defines in such an extensive manner attacker behaviours while being open and frequently updated. It also has the advantage to have gained popularity in recent years for multiple purposes and to have been integrated into popular threat information sharing technologies [3], [39].

ATT&CK is a structure created for cybersecurity professionals to have a common taxonomy to describe adversary behaviours, which is divided by "technology domains":

- "Enterprise", which describes behaviours on Linux, macOS, and Windows.
- "Mobile", which focuses on Android and iOS.

Beyond these domains, it also documents behaviours for reconnaissance and weaponisation under the "PRE-ATT&CK" designation. As we are mostly interested in the action that an adversary could perform on an enterprise network, we focus, in our case, on Enterprise ATT&CK.

The framework has four core components:

- Tactics, which represent the "why" of a behaviour. In the case of Enterprise ATT&CK, there is a number of twelve: "Initial Access", "Execution", "Persistence", "Privilege Escalation", "Defense Evasion", "Credential Access", "Discovery", "Lateral Movement", "Collection", "Command and Control", "Exfiltration", and "Impact".
- Techniques, which are the "how" an adversary will perform an attack tactic. As of July 2019, there are 244 techniques. Each technique is associated with one or more tactics.
- Mitigations, which are any tools or processes that can avert the desired outcomes from the use of techniques. Each mitigation can be used for one or more techniques, and techniques can have none to several mitigations associated with them.
- Recorded adversary uses of techniques under two categories:
  - Groups, which are collections of related intrusion activities.
  - Softwares, which can be either tools or malware.

Privilege Escalation	**Defense Evasion**	**Credential Access**
Access Token Manipulation	Access Token Manipulation	Account Manipulation
Accessibility Features	Binary Padding	Bash History
AppCert DLLs	BITS Jobs	Brute Force
AppInit DLLs	Bypass User Account Control	Credential Dumping

Table 1: Examples of relationship in the ATT&CK matrix between tactics and techniques. Each column represents a different tactic, which is linked to several techniques, whose title is in each cell.

Tactics and Techniques are the key elements of the ATT&CK framework and constitute the labels in our classification. Table 1 presents examples of how some of these tactics relate to some techniques. For instance, an attacker would use the "Access Token Manipulation" technique from the "Privilege Escalation" and the "Defense Evasion" tactics to increase its permissions level and avoid being detected.

IV. DATA COLLECTION

In order to use a text classification approach to our problem, we need to collect enough data to train and test various models. The only data we could find already labelled was the content of the ATT&CK website, which can be found on the GitHub of MITRE [40], structured in the STIX 2.0 format (Structured Threat Information Expression) [3].

Figure 1 represents the organisation of the dataset for Enterprise ATT&CK. Each name be-

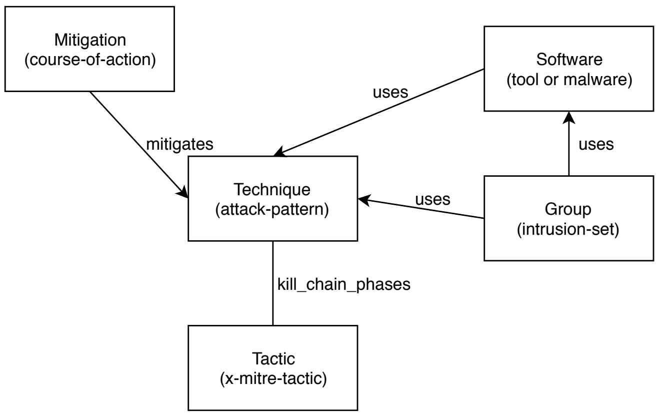


Fig. 1: Organisation of the ATT&CK dataset as provided on GitHub under a STIX format and as presented on the ATT&CK website.

tween brackets is the STIX object under which the particular ATT&CK object was saved. Every instance of "course-of-action", "malware", "intrusion-set", and "tool" are linked to one or several "attack-pattern" using the object "relation" of type "mitigates" or "uses". The techniques and tactics STIX object are linked through a "kill_chain_phases" references directly inside the techniques STIX objects to a tactic STIX object.

Each of these objects includes external references, which are usually threat reports or technical descriptions, that we can use as training data for our tool. We only use the ones which are directly connected to an "attack-pattern", either contained in an object "attack-pattern", or included in a "relationship" object linked to an "attack-pattern". We do not use any of the references from other objects, as we cannot guaranty that a text explaining a particular object will mention the techniques related to it. We also only collect the references with an URL, as references without URLs are usually not freely available on the internet and, thus, not representative of the texts in which we are interested. From these lists of URLs for each technique, we can retrieve the list of techniques for one document, and as all techniques are linked to specific tactics, the list of tactics for one report.

As a result, and once the links for non-accessible reports and webpages not presenting written reports (e.g. video, code, zip files) were removed, we have a total of 1490 different reports. The documents can be separated in two types: the ones written in HTML (1311 URLs) and the ones in PDF format (179 URLs). The documents in PDF format can easily be extracted using existing tools [41], allowing us to obtain a text format report quickly. However, the ones in HTML are more challenging to extract, as they are not all from the same source. Two hundred eighty-three different websites host these 1311 reports, which mean that we have potentially 283 different HTML architectures. To simplify the extraction, we decided on parsing only the text in paragraph HTML tags and copied manually the reports which could not be collected this way. Some of the reports extracted contained too much noise parsed at the same time as the report, which could hinder the classification. This problem is solved during the pre-processing of the text.

While each tactic has at least 80 reports in which it is included, it is not the case for the techniques. 29 out of the 244 techniques present in the ATT&CK framework are included in less than three reports (see Appendix XI-A), which we estimate as being too few for our task. Thus, we only work on 215 techniques out of the 244 present in the ATT&CK framework. Nonetheless, we obtained an imbalanced dataset from this collection, which has an impact on the classification of the reports and will be addressed in Section VII of this report.

V. Methodology overview

The definition of our methodology derives from the work already done on the retrieval of TTPs in CTRs and the data we could collect. We settled on using a text classification approach over an ontology-based retrieval due to Ayoade *et al.*'s paper [35], which stated having better results than Husari *et al.*'s work [30].

As mentioned before, the reports collected contain several techniques and tactics at the same time, meaning we are facing a multi-label classification problem. We are also aware, because of the analysis made during the data collection, that our dataset is imbalanced, which might impact our classification, thus adding a problem to our research. The ATT&CK framework being organised under a particular hierarchy, having tactics at a higher level and techniques at a lower level, we have a hierarchical classification problem. Because of this, we want to evaluate the classification performance by label types separately. While the label relationships have an impact during the multi-label classification, we also investigate the use of ground truth knowledge about these relationships in post-processing methods.

We define our main research question as "*How can we automatically retrieve ATT&CK tactics and techniques from cyber threat reports?*", and, based on all these problems, we address the following subquestions:

1) Which model would be the best to classify CTRs by tactics and techniques?
2) How can we solve the imbalanced dataset problem to improve the classification?
3) Can we improve classification using a post-processing method based on the labels relationships?

Each of these questions is studied in this report, detailing the approach and the results. To define which model would be the best to classify CTRs by tactics and techniques, we tested several multi-label classification models, either with classifiers being adapted to our multi-label problem, or using problem transformation methods from the Scikit-learn library [42]. All of them are compared when working with different text representation methods: term-frequency (TF) and term frequency-inverse document frequency (TF-IDF) weighting factors [43], and Word2Vec [44]. To answer the second question, we studied the impact of the increase in size and the random resampling of the dataset on the performance of our classification, observing the effect on the prediction of tactics and techniques, overall and for each of them separately. To finish, we defined several post-processing approaches using the tactics and techniques relationships to test on our model, based on literature about similar problems to ours and our observations of our initial results, and selected the best ones to implement in our tool.

In order to evaluate our work more precisely, all experiments are done using a five-fold cross-validation. We also defined metrics and a baseline, to observe the evolution of our work and establish if our model is learning.

A. Metrics

Based on our goal, we use as evaluation metrics the precision, the recall, and the $F_{0.5}$ score, with two different averaging methods: macro-averaging and micro-averaging [45]. The $F_{0.5}$ score is the weighted average between the precision Pr and the recall Re with an emphasize on the precision Pr. We choose this metric over the F_1 score because, in this application, the precision is more important than the recall, as when predicting the TTPs for a report, we want to be sure these TTPs are present in the CTR, even if there are only a few of them predicted.

$$F_{0.5}\left(y_t, y_p\right) = 1.25 \cdot \frac{Pr\left(y_t, y_p\right) \cdot Re\left(y_t, y_p\right)}{\left(0.25 \cdot Pr\left(y_t, y_p\right)\right) + Re\left(y_t, y_p\right)}$$

Even though all these metrics are essential, the most important metric for choosing a final model to use for our tool is the macro-averaged $F_{0.5}$ score, as we want to give equal importance to each label.

B. Baseline

To follow the improvements done during the evaluation of different models and validate if they are learning from the training set, we defined a naive baseline, done by attributing to the testing set instances the most frequent label in the training set. The baseline results can be found in Table 2.

Majority	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactics	48.72%	19.00%	37.10%	4.43%	9.09%	4.93%
Techniques	9.57%	2.51%	6.11%	0.05%	0.48%	0.06%

Table 2: Baseline for classification by tactics and techniques based on the most frequent label

VI. Multi-label classification of reports by tactics and techniques

In this section, we answer the question: "*Which model would be the best to classify CTRs by tactics and techniques?*", by defining how to do the pre-processing, which feature extraction method is the best and what classifier is best fitting our problem. We also address the problem of overfitting that we encountered in our evaluation.

A. *Pre-processing*

The first step before classifying our reports is pre-processing. It is the cleanup steps, as we have parsed some noise during the data collection, which can hinder the classification process. We studied different ways to do the pre-processing and kept the one improving the most the classification of the reports. We first remove any potential leftover HTML tags, non-word and whitespace characters. Then we created regular expressions to remove strings that could hinder the classification, such as Hashes, IP addresses, email addresses or URL. We then removed all stop words, using the list provided in Natural Language Toolkit (NLTK) [46], as well as using a stemmed form of the words from the text.

B. *Text representation*

We studied different ways to represent the text and obtain features to classify the reports. We tried basic extraction such as term-frequency (TF) and term frequency-inverse document frequency (TF-IDF) weighting factors [43], either based on bag-of-words or uni-grams, bi-grams and tri-grams. We also tested applying a maximum and a minimum frequency of appearance in the overall corpus. Because the number of features chosen would impact the fitting of the model, we studied selections approach and the number of features to use to improve the model.

Overall bags-of-words presented better results than grouping words. We also defined that selecting half of the words with the highest TF or TF-IDF scores, presented better results, than choosing a constant number of words for each report or not limiting the number of features at all.

We also decided on testing the word-embedding approach Word2Vec [44], as it has been frequently used in text classification over the recent years. We decided on training it on our current dataset, instead of using an existing pre-trained version as we found inconsistent results in an early trial [47]. We tested both averaging and summing word vectors as simple approaches to represent the text.

C. *Classifiers*

Several methods are possible when facing a multi-label classification problem: we can either use adapted algorithms or transform the problem. Ways to transform the problem are to use binary relevance, which trains a binary classifier for each label independently [48], a classifier chain, which is similar to binary relevance but uses the relation between labels [49], or label powerset, to transform it into a multi-class problem between all labels combination [50]. A label powerset model is difficult to apply to our case, as we have too many labels and not enough data to cover the entirety of the possible combination. However, we can use binary relevance and classifier chains with different types of classifiers.

As we primarily use Scikit-learn library [51], we decided on using the multi-label classifiers implemented in it, namely: multi-label K-Nearest Neighbours [52], the multi-label Decision Tree and Extra Tree classifiers, as well as the Extra Trees, Random Forest ensemble methods for multi-label classification [51]. Then we selected several algorithms to use in the classifier chains, and the binary relevance models, based on their uses in similar problems to ours (i.e. small and imbalanced dataset, text classification) presented in the library's documentation [51], general data science resources [53], [54] and various research papers. We tested several Linear models: Logistic Regression [55], Perceptron [56], Ridge [57]; but also Bernoulli Naive Bayes classifier [58], K-Nearest Neighbors [59] and Linear Support Vector Machine (Linear SVC) [60]. We also experimented with multiple tree-based models like Decision Tree [61] and Extra Tree classifiers [62], as well as ensemble methods like Random Forest [63], Extra Trees [64], AdaBoost Decision Tree [65], Bagging Decision Tree [66] and Gradient Boosting [67], [42]. Other classifiers from the library were tested but did not outperform our naive baseline; thus, they will not be presented in the results.

As we faced overfitting during our tests, in addition to reducing the number of features as mentioned previously, we tried regularising the selected models, fine-tuning the hyper-parameters of the tested classifiers and focusing on the simpler models. While these steps might have improved our classification in general, we privileged the best results on the testing set with overfitting, over worse results on the testing set and lower results

	Without resampling						With resampling					
	Micro			Macro			Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Majority Baseline	48.72%	19.00%	37.10%	4.43%	9.09%	4.93%	-	-	-	-	-	-
Term Frequency												
CC Adaboost DT	64.73%	50.84%	61.30%	60.87%	45.20%	56.45%	-	-	-	-	-	-
CC Bagging DT	67.19%	40.57%	59.38%	64.01%	33.46%	51.89%	-	-	-	-	-	-
CC Gradient T Boosting	71.84%	44.33%	63.61%	66.72%	37.42%	55.91%	-	-	-	-	-	-
CC Logistic Regression	63.81%	54.35%	61.61%	58.86%	47.78%	55.85%	-	-	-	-	-	-
CC Perceptron	56.34%	56.58%	56.35%	51.63%	50.79%	51.04%	-	-	-	-	-	-
CC Linear SVC	63.81%	51.02%	60.71%	58.89%	44.31%	54.64%	-	-	-	-	-	-
BR AdaBoost DT	62.30%	49.27%	59.08%	57.26%	42.37%	52.91%	49.77%	60.45%	47.08%	52.24%	65.22%	48.91%
BR Bagging DT	68.26%	42.36%	60.75%	63.71%	34.30%	51.54%	52.39%	58.59%	49.36%	54.32%	63.73%	50.48%
BR Gradient T Boosting	71.42%	48.19%	65.04%	66.35%	40.16%	56.78%	54.87%	**66.41%**	51.94%	57.66%	72.47%	53.92%
BR Logistic Regression	63.71%	54.42%	61.54%	58.74%	47.81%	55.76%	56.94%	61.56%	52.44%	58.66%	**66.74%**	53.90%
BR Ridge Classifier	61.85%	51.46%	59.39%	55.64%	45.31%	52.92%	49.93%	55.86%	45.48%	51.50%	58.97%	47.03%
BR Linear SVC	64.70%	51.55%	61.51%	59.20%	44.36%	54.89%	54.95%	57.69%	50.76%	56.45%	63.37%	51.83%
Term Frequency-Inverse Document Frequency												
CC Adaboost DT	61.42%	49.86%	58.59%	57.71%	44.09%	53.79%	-	-	-	-	-	-
CC Gradient T Boosting	71.15%	43.15%	62.52%	67.40%	36.29%	54.97%	-	-	-	-	-	-
CC Perceptron	62.06%	54.97%	60.28%	58.05%	49.26%	54.72%	-	-	-	-	-	-
CC Ridge Classifier	74.40%	41.27%	63.27%	**67.63%**	33.31%	51.74%	-	-	-	-	-	-
CC Linear SVC	71.63%	44.89%	63.41%	65.70%	36.76%	54.59%	-	-	-	-	-	-
BR AdaBoost DT	61.02%	51.02%	58.61%	56.61%	44.67%	53.19%	49.52%	58.46%	46.12%	51.84%	63.79%	47.86%
BR Bagging DT	66.88%	41.44%	59.39%	63.92%	34.95%	52.29%	53.15%	56.68%	50.06%	54.70%	61.94%	50.87%
BR Gradient T Boosting	70.13%	46.85%	63.66%	65.08%	38.94%	55.03%	54.30%	63.40%	50.79%	56.92%	70.53%	52.48%
BR Logistic Regression	71.04%	50.70%	65.61%	59.00%	40.53%	51.21%	59.25%	63.13%	***56.69%***	61.09%	69.80%	***57.14%***
BR Perceptron	65.20%	55.35%	62.80%	60.54%	48.29%	56.24%	57.26%	62.68%	53.97%	59.28%	69.11%	54.98%
BR Ridge Classifier	**72.40%**	48.90%	**65.83%**	66.57%	38.58%	53.32%	***59.47%***	62.28%	55.77%	***61.13%***	68.89%	56.59%
BR Linear SVC	65.64%	***64.69%***	65.38%	60.26%	***58.50%***	**59.47%**	57.71%	66.17%	53.88%	59.97%	71.18%	55.75%

Table 3: Best results from initial and resampling classifications for tactics prediction (with CC = Classifier Chain, BR = Binary Relevance, DT = Decision Tree, T = Tree).

on the training set.

The issue of separating classifications by tactics and techniques was also addressed during this phase of the research, especially as, in some cases, the best pre-processing and classifier hyper-parameters for the same classifier and text representation method, applied to the separated tactics and techniques predictions, would be different. Ultimately, in classification models were the correlations between labels is used (i.e. multi-label classifiers and classifier chains), we predicted the tactics and techniques together, with the best parameters possible for both label types. In the cases in which the label relationship did not matter (i.e. binary relevance), we split the classification by tactics and techniques, applying to each the best parameters possible for their category.

D. *Results and discussion*

The evaluation of the different models we tried to adapt to our problem is detailed in Appendix XI-B, and the best results can be found in Tables 3 for the tactics and 4 for the techniques.

When we compare the text representation methods, we can observe that, overall, models using Word2Vec either with a sum or an averaging of the word vectors, underperformed compared to the ones using a TF or TF-IDF weighting system. We can observe minor differences in the use of the average against the sum of the vector in the Word2Vec approach, but we cannot claim that one way is better than the other as the results depend on the classifier used.

Observing the results for the different classifiers, we can see that adapted algorithms tend to underperform compared to the classifier chains and the binary relevance models, for both the tactics and techniques predictions. Overall, the classification made with binary relevance instead of classifier chain performs slightly better, which means that the relationship between labels did not have as much impact as expected.

Some classifiers stand out among the other for each type of labels. For the classification by tactics, the AdaBoost Decision Tree, the Gradient Tree Boosting, Perceptron and the Linear SVC in classifier chains or binary relevance models have the best performance, indifferently using TF-IDF or TF. The Bagging Decision tree, the Ridge classifier and the Logistic Regression perform well when used in binary relevance. They also perform well in a classifier chain if Logistic Regression

	Without resampling						With resampling					
	Micro			Macro			Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Majority Baseline	9.57%	2.51%	6.11%	0.05%	0.48%	0.06%	-	-	-	-	-	-
Term Frequency												
CC Adaboost DT	36.64%	13.27%	27.05%	18.38%	8.98%	13.72%	-	-	-	-	-	-
CC Bagging DT	39.95%	5.83%	18.24%	18.50%	7.90%	12.32%	-	-	-	-	-	-
CC Ridge Classifier	14.82%	25.59%	16.16%	10.88%	23.40%	11.66%	-	-	-	-	-	-
CC Decision Tree	22.44%	17.94%	21.31%	18.64%	14.98%	16.46%	-	-	-	-	-	-
BR AdaBoost DT	35.60%	16.04%	28.56%	18.23%	9.74%	14.23%	26.44%	23.02%	22.79%	14.65%	16.16%	12.45%
BR Bagging DT	39.30%	7.63%	21.42%	16.92%	7.53%	11.88%	26.99%	12.79%	16.81%	12.34%	15.43%	10.75%
BR Gradient T Boosting	27.60%	12.67%	22.31%	20.25%	12.72%	16.26%	18.47%	20.32%	16.73%	14.80%	24.51%	13.96%
BR Ridge Classifier	14.96%	25.96%	16.33%	10.98%	23.50%	11.74%	11.40%	**31.83%**	12.83%	9.05%	**32.50%**	10.04%
BR Linear SVC	22.87%	19.22%	21.98%	15.36%	11.39%	13.41%	17.85%	24.59%	17.97%	12.85%	17.01%	11.95%
BR Decision Tree	23.06%	18.53%	21.94%	18.00%	14.76%	16.18%	17.46%	24.90%	17.11%	14.70%	21.66%	13.61%
Term Frequency-Inverse Document Frequency												
CC Adaboost DT	37.06%	13.06%	26.98%	17.70%	8.44%	13.19%	-	-	-	-	-	-
CC Decision Tree	23.18%	18.47%	22.02%	18.83%	14.85%	16.77%	-	-	-	-	-	-
BR AdaBoost DT	35.04%	14.77%	27.41%	17.23%	9.05%	13.36%	27.53%	18.12%	22.32%	14.55%	13.00%	11.85%
BR Gradient T Boosting	25.85%	11.60%	20.72%	18.83%	12.09%	15.21%	19.45%	16.76%	17.10%	15.33%	21.69%	14.38%
Logistic Regression	**52.82%**	3.66%	14.33%	7.86%	2.76%	5.18%	***42.05%***	4.98%	12.41%	7.53%	4.79%	5.64%
BR Perceptron	30.45%	18.26%	26.82%	21.89%	14.61%	18.32%	25.16%	22.56%	23.03%	19.50%	21.29%	17.32%
BR Linear SVC	37.18%	***29.79%***	**35.02%**	**28.84%**	***22.67%***	**25.06%**	31.16%	32.86%	***29.37%***	***26.27%***	28.05%	***22.87%***
BR Decision Tree	20.72%	18.31%	20.15%	16.88%	14.57%	15.55%	17.44%	22.91%	17.02%	15.45%	20.21%	14.21%

Table 4: Best results from initial and resampling classifications for techniques prediction (with CC = Classifier Chain, BR = Binary relevance, DT = Decision Tree, T = Tree).

and the Bagging Decision Tree classify text using TF weighting, and for the Ridge classifier, using TF-IDF. Similar models work as well for the techniques prediction, in addition to Decision Tree classifier in a binary relevance or classifier chain model.

For the tactics prediction and more importantly, for the classification by techniques, the binary relevance Linear SVC with a TF-IDF weighted text representation stands out compared to the other models. By testing this model on the training set, we can observe a significant difference with the application to the testing set, with a macro-averaged $F_{0.5}$ score of 95.75% for the tactics prediction and 89.67% for the techniques prediction, compared to 59.47% and 25.06% for the tactics prediction and techniques prediction on the testing set. This demonstrates the overfitting problem mentioned previously, similarly observed with several models, which we could not improve, despite limiting the number of features, regularising the models, and fine-tuning the hyper-parameters of the classifiers.

VII. Imbalanced learning and impact of the size of the dataset on the classification

One of the problems we face with our dataset is the fact that it is imbalanced, as some tactics and techniques have more data than others. The first solutions we tested to solve this problem were to use algorithms which tend to perform well on an imbalanced dataset, penalise the classification models, and assign a weight to the different label, during the initial tests we presented in the previous section.

Another possibility to overcome the imbalanced problem, as well as the overfitting problem, would be to add more data. Thus, in this section, we study the possible impact of the augmentation of the dataset on the performance of the classification. We also study the resampling of the dataset, that we tested on different models, as it is also a solution against an imbalanced dataset.

Another possibility would have been to group some of the labels together, which would make sense as some techniques are more precise in their definition than other, and differentiation between techniques and sub-techniques is possible in the context of the ATT&CK framework [68]. However, such a solution would require, in our case, to manually re-label our dataset, which we have decided to not do by lack of time. This redefinition following the idea of techniques and sub-techniques could also increase the imbalance if we were to use the current dataset solely.

A. Evolution of the performance of the classifier with the increase of data

Like for the overfitting problem, the ideal solution to an imbalanced dataset would be to add more

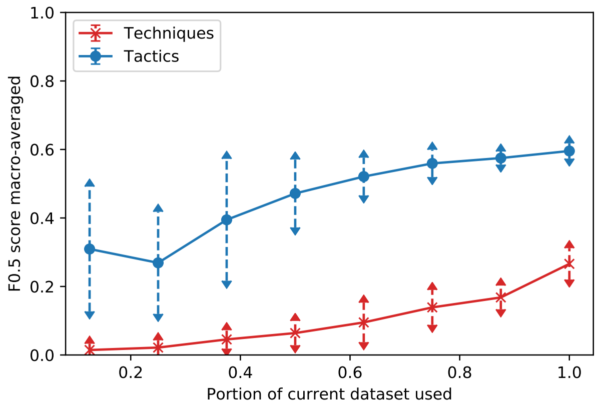


Fig. 2: Evolution of $F_{0.5}$ score (macro-averaged) for different dataset size fo tactics and techniques predictions.

data to our training set. However, because of lack of time and the fact that we do not know how much more data we would need, we did not try this. Nonetheless, we want to confirm the hypothesis that more data could improve the classification. To do so, we take differents percentage of our dataset and evaluate our best performing model on it. More precisely, we took one-eighth, one-fourth, three-eighth, one-half, five-eighth, three-fourth, and seven-eighth of our dataset randomly five separate times and tested on those the TF-IDF and binary relevance Linear SVC model, which was the best performing model in the previous section, using five-fold cross-validation.

Figure 2 presents the results from these tests. We can observe an improvement when the dataset increase in size. While the increase is slow for the tactics, the improvement for the techniques prediction is more significant.

Based on these results, we can expect that by adding more data to our set, we could improve our results, and possibly limit the overfitting and the imbalance. As we cannot add more data ourselves to the dataset, we can implement in our tool a feedback function for the user to correct the false results from the classifications and save the corrected results to the training set.

B. *Resampling the imbalanced dataset*

One of the solutions to improve the classification over an imbalanced dataset is to resample the data to balance the set. Many solutions exist to do so, but the implementations fitting multi-label problems are rare. In our case, we decided to randomly resample each label, trying out different ratios and undersampling or oversampling. We only test our resampling on the models with binary relevance, associated with the TF and TF-IDF feature extractions, mainly for simplicity of implementation and because the binary relevance models perform overall better than multi-label classifiers and faster than classifier chains. Since we use binary relevance, we split the classification by tactics and techniques and attributed different sampling ratio. In the case of the tactics, the resampling method was a combination of over- and undersampling for all tactics to have in their training set with 400 positive reports and 400 negative reports. For the techniques, the best resampling was also a balance between oversampling and undersampling with 125 positive reports and 500 negative reports. However, since only seven techniques have over 125 reports, we are mostly oversampling the techniques data.

The resampling on the tactics prediction resulted in an overall increase of the recall, but also a decrease in precision and, thus, in $F_{0.5}$ score (see Table 3 or Appendix XI-C.1 for the detailed results). Only a few models improve because of the resampling: only ExtraTrees and RandomForest benefit from it. The Logistic Regression model does also increase its $F_{0.5}$ macro-averaged score when paired with TF-IDF, which is the maximum obtained in the resampling evaluation. However, it is not enough to outdo the Linear SVC model without resampling, which is found to be the best performing model in the previous section.

The resampling on the techniques prediction performs similarly (see Table 4 or Appendix XI-C.2 for the detailed results). The recall increases, while the precision and the $F_{0.5}$ score decrease. The only improvements in performance are for models which performed very poorly in the initial classification, and they do not overtop the original non-resampled best performing model from the previous section.

The observation that the resampling does not improve the classification seems incoherent with the finding that an increase in the size of the dataset improves the classification. A possible explanation is that there are differences in the impact of the dataset size for each label.

C. Relationship between labels prediction and size of dataset

We want once again to verify the correlation between the number of reports and classification performance, but, this time, focusing on the impact of the different sampling size for each label with our binary relevance Linear SVC model, as the results from our previous tests seemed to contradict each other.

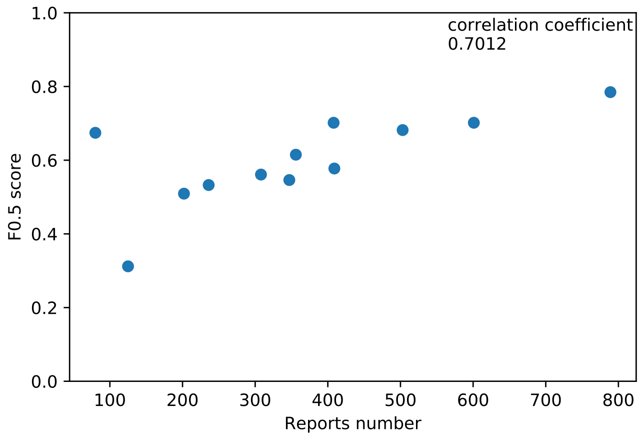


Fig. 3: Correlation analysis between the $F_{0.5}$ score and the number of reports by tactics.

By plotting the $F_{0.5}$ score for each tactic based on the number of reports by tactic, we obtain Figure 3. We observe a higher $F_{0.5}$ score for the tactics which have more reports, compared to the ones with less of them. We can use the Pearson correlation coefficient to evaluate the linear correlation between these two variables [69]. This coefficient is of 0.7012, indicating a positive linear relation between the prediction of the tactics and the number of reports labelled as containing them in the dataset.

We do a similar plotting with the techniques, taking the $F_{0.5}$ score resulting from the binary relevance Linear SVC and the number of reports for each technique to obtain Figure 4. We observe a large concentration of $F_{0.5}$ scores, going from very low to almost perfect, for the techniques with a low number of reports. In this case, the correlation coefficient is 0.084, which demonstrates

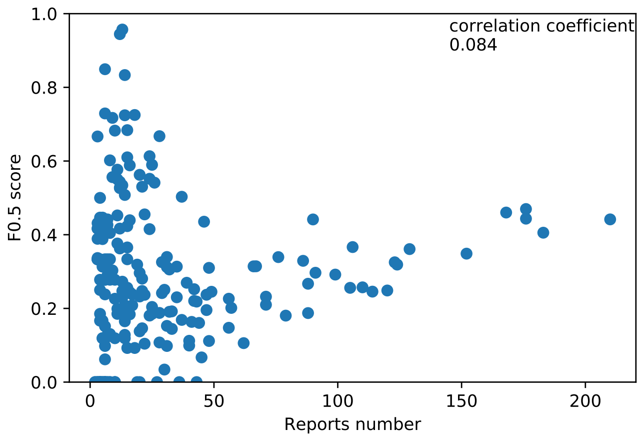


Fig. 4: Correlation analysis between the $F_{0.5}$ score and the number of reports by techniques.

no relationship between the performance of the techniques prediction and the number of reports by techniques.

As we were able to establish a relationship between the number of reports and the prediction of a tactic, but we could not establish a relationship between the techniques related to these tactics and the number of reports, we want to observe the change in techniques prediction performance with different sampling. Observing the Figure 5, which was build using the $F_{0.5}$ score of each techniques prediction from TA005 with different resampling ratios on our Linear SVC model (see Appendix XI-C.3 for complete results over all techniques predictions), we can observe that some techniques have different performances depending on the different sampling size and ratios. However, for some techniques, the $F_{0.5}$ score is barely changing, meaning that whatever the size of the training set for those, they are difficult to predict. We do not have enough data to establish a relationship between an increase of performance and the increase of data for each technique, even for the techniques which are impacted by the resampling. However, we did observe that some techniques are impacted negatively by an oversampling too large, while a majority of others seems impacted positively.

The conclusion from this analysis of the impact of the dataset size would be that it is possible to improve the classification by adding more data to the training set. Based on the positive linear correlation between the $F_{0.5}$ score and the number

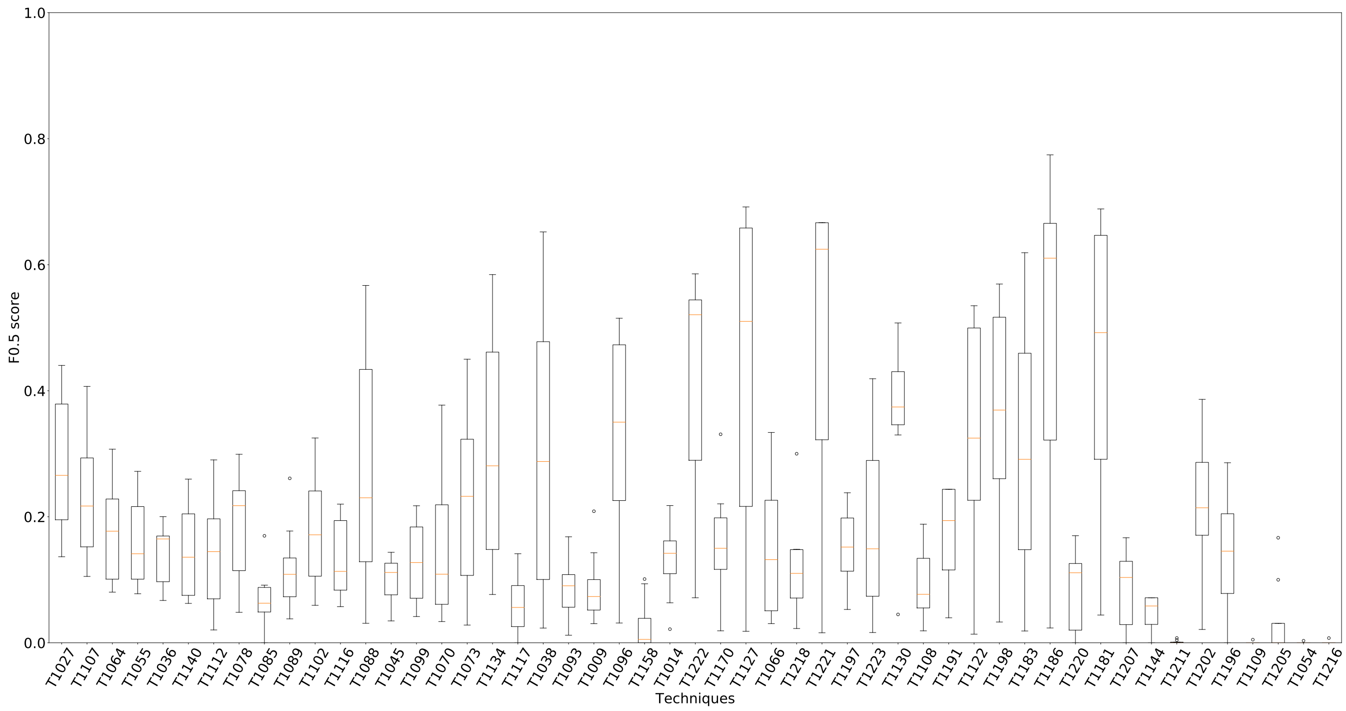


Fig. 5: Impact of training sampling size on $F_{0.5}$ score of each technique from TA0005 prediction, with techniques sorted by decreasing number of CTRs in the original dataset. This graph is a representative sample of the complete graph over all techniques presented in Appendix XI-C.3.

of reports by tactics for the tactics prediction and the fact that, currently, the tactics with the most reports has a $F_{0.5}$ score of 80%, we can expect that, by adding more reports to our dataset, especially for the tactics with fewer reports, we would be able to obtain a macro-averaged $F_{0.5}$ score of at least 80% for the tactics prediction. However, while the increase of data would impact the tactics prediction and some of the predictions of the techniques that compose them, it might not improve the prediction of some techniques. These techniques can be considered as difficult to predict, but since it is not due to the quantity of data, it might be due to the dataset quality. A low-quality dataset would also explain why we observed some techniques prediction to be worse when adding more data, or why there is an overall lack of improvement when performing the resampling in Section VII-B.

VIII. Applying post-processing methods to the multi-label classification

The binary relevance Linear SVC model, with TF-IDF weighted bag-of-words, we considered the best at the end of section VI, is still considered our best performing model, despite trials at resampling the dataset for each label. From the previous section, we also defined that some of the techniques are more difficult to classify than others, as the quantity of data in the training set barely impacts their prediction. As we are currently limited in our results, we want to use the relation between the different labels to try to improve our predictions. We already tested this influence on the classification by using multi-label classifiers and classifier chains. However, our best results were obtained on a model which did not consider it. Thus we want to use the ground truth knowledge to improve the classification by studying different post-processing approaches to add to our current model.

A. *Approaches studied*

1) Using relationships between tactics and techniques

Because tactics and techniques are directly related to the ATT&CK framework, we can use different simple post-processing methods to try to improve the classification based on these relationships.

a) Direct rules from techniques to tactics

The first approach is to define the tactics of a report based on the techniques classification. Because each technique belongs to one or more tactics, we can consider rules such as: if we predict a technique Te_1 in a report and Te_1 is part of the tactic Ta_1, then Ta_1 is in the report.

b) Tactics as features

Since the techniques prediction underperformed compared to the tactics prediction, we want to use the results from the classification by tactics to improve the prediction of the techniques. Our second method is to use the results of the tactics prediction as features for the classification by techniques. It is close to the classifier chain method, but it solely uses the tactics prediction to influence the techniques prediction. Additionally, this method is not purely a post-processing approach; however, it fits in our goal of improving the classification of a part of the labels using the other part.

c) Confidence propagation

As a third approach, we want to use the method described in Ayoade *et al.*'s paper [35], itself inspired by Wu *et al.*'s work [70]. The idea behind it is to use the confidence score of each technique and the confidence score of the associated tactics to create boosting factors. Each boosting factors are multiplied to each associated tactic confidence score, and this ensemble is added to the pre-existing technique confidence score. Based on the associated tactics confidence score, the techniques' confidence score increases or decreases, which impacts the final predictions. As we use Scikit-learn's Linear SVC [60], instead of using the confidence scores of the classifier, we use the decision function scores, as normalising them before the application of this method (which is not automatically done in the library) worsen the performance of the model.

d) Hanging node

The fourth approach is based on the observation that for 30% of the techniques predicted for a report, not all related tactics were predicted, meaning that either the techniques or the tactics were incorrectly predicted. The analysis of the distribution of the frequency of the confidences shows that the false predictions tend to be closer to the threshold distinguishing a positive from a negative label than the true predictions, especially for the tactics predictions (see Appendix XI-D). The fourth post-processing method would, thus, be to use the confidence scores resulting from each type of classification to add tactics or remove techniques.

As presented in Algorithm 1, for a connected pair of technique and tactic, if the technique is

```
Input:
$R$ /* report */
$\{Ta_i^R, ...Te_k^R\}$ /* all tactics and
    techniques predicted as being present
    in the $R$ */
$Te_x$ /* one of the techniques */
$Ta_y$ /* one of the tactics */
$p(Te_x \in R)$ /* the probability of $Te_x$
    being predicted for $R$ */
$p(Ta_y \in R)$ /* the probability of $Ta_y$
    being predicted for $R$ */
Output:
$\{Ta_i^R, ...Te_k^R\}$ /* updated ensemble of
    tactics and techniques being present in
    the $R$ */
Data:
$th \in \mathbb{R}$ /* classification threshold */
$a, b, c, d \in \mathbb{R}$ /* defined thresholds */
begin
    if $Te_x \rightarrow Ta_y$ then
        if $p(Te_x \in R) > a > th$ and
          $b < p(Ta_y \in R) < th$ then
            $\{Ta_i^R, ...Te_k^R\} += Ta_y$ /* adding
                $Ta_y$ to the ensemble of
                tactics and techniques being
                present in the $R$ */
        if $th < p(Te_x \in R) < c$ and
          $p(Ta_y \in R) < d < th$ then
            $\{Ta_i^R, ...Te_k^R\} -= Te_x$
                /* removing $Te_x$ to the
                ensemble of tactics and
                techniques being present in
                the $R$ */
```

Algorithm 1. Hanging node approach

predicted with a high confidence score, while the tactic is not predicted, but with a confidence score close to the classification threshold th, then we can add the tactic to the tactics and techniques present in the report. On the contrary, if the techniques are declared present, with a confidence score near the classification threshold th and the tactic is not predicted as present in the report, with a low confidence score, then we can remove the techniques from the labels predicted. The thresholds a, b, c, d are defined after testing different values and comparing the improvement in classification performance.

Other similar connections between techniques and tactics are difficult to implement, as several techniques have similar tactics. Adding a technique if a related tactic is also present in the report would be misguided, as the high probability could be due to another technique. Similarly, removing a tactic, if a related technique is absent, would be injudicious, for the same reason.

With the current dataset, we use for classification threshold $th = 0.5$ and the thresholds $a = 0.55, b = 0.05, c = 0.95, d = 0.30$, which allow the highest macro-averaged $F_{0.5}$ score. As we use the Linear SVC from Scikit-learn [60], we have defined to the confidence score by scaling the decision function scores using min-max scaling with $min = -1$ and $max = 1$ [71].

2) *Using relationships between techniques*

Based on the same data as the one we used to build the dataset, we can retrieve joint probabilities between techniques, using their common appearances in the same malware, tool or group. Using these probabilities, we decided to test three separate methods to improve the techniques predictions.

a) Rare association rules

The first approach follows the work of Benites and Sapozhnikova [72], based on which we selection association rules between techniques. The first step is to calculate the Kulczynski measure [73] for each pair of techniques. These values are forming a curve from which we can determine the variance. If this variance is low, the threshold to decide on the pairing rules based on their Kulczynski measure is defined based on the median of differences between neighbour values. If this variance is high, it is based on the average of the values slightly lower than the mean of this curve.

b) Steiner tree association rules

The second approach in this category has been described by Soni *et al.* [74], in which they formulate the label coherence as a Steiner Tree Approximation problem [75]. Once the techniques prediction was performed, for each report, we create a directed tree [76] in which edges have for weight the conditional probabilities between the two nodes and the direction is given by the following criterion:

$$\begin{cases} Te_i \rightarrow Te_j, & p(Te_i|Te_j) \leq p(Te_j|Te_i), \\ Te_i \leftarrow Te_j, & otherwise. \end{cases}$$

Then we use Edmond's algorithm [77] to be left with a reduced tree and a limited number of connection between techniques. Based on the predictions from the classification, we find in the graph the techniques which descend from the predicted techniques with the K highest weights. In our case, we define $K = 15$.

c) Knapsack

The third approach also comes from the paper from Soni *et al.* [74] for which they consider the label assignment as a resource allocation problem. In this case, we solve the 0-1 Knapsack problem [78], in which a label is included in the prediction if its conditional probability based on labels predicted to increase the overall log-likelihood.

B. Results and discussion

The final results for each approach are presented in Table 5.

We can observe that for the prediction of tactics, the independent classification is the best performing. It is to be expected as the current independent classification by techniques does not have higher performance, meaning that every post-processing depending on it would potentially lower the results.

However, if the techniques prediction were to become perfect, both post-processing approaches would help improve the tactics classification. The approach *1.a. direct rules from techniques to tactics* would have a perfect performance, as it is purely rule-based. However, it is unlikely that the techniques prediction would improve without the tactic prediction improving, so the results from

Approaches	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactics						
Inde.	**65.64% ±3.76%**	64.69% ±3%	**65.38% ±2.87%**	**60.26% ±3.2%**	58.50% ±3.68%	**59.47% ±2.29%**
1.a.	63.32% ±6.42%	52.34% ±5.08%	60.45% ±3.8%	57.49% ±5.41%	47.94% ±5.87%	54.59% ±3%
1.d.	59.60% ±2.89%	**68.04% ±3.06%**	61.08% ±3.42%	54.41% ±2.97%	**61.28% ±3.12%**	55.42% ±3.2%
Techniques						
Inde.	**37.18% ±6.75%**	29.79% ±5.91%	**35.02% ±5.32%**	28.84% ±6.9%	22.67% ±5.94%	25.06% ±6.09%
1.b.	31.55% ±5%	35.17% ±6%	31.97% ±4.31%	24.70% ±6.29%	22.74% ±6.3%	22.38% ±5.72%
1.c.	33.07% ±7.11%	**38.17% ±5.41%**	33.69% ±6.2%	28.14% ±6.72%	**28.19% ±4.99%**	26.06% ±5.69%
1.d.	32.19% ±6.05%	29.27% ±6.2%	31.34% ±5.23%	**32.35% ±6.68%**	22.21% ±4.89%	**27.52% ±6.03%**
2.a.	33.70% ±6.79%	36.26% ±5.16%	33.89% ±5.76%	28.08% ±6.8%	25.18% ±5.22%	25.20% ±5.78%
2.b.	37.06% ±6.77%	29.79% ±5.99%	34.93% ±5.34%	28.84% ±6.94%	22.67% ±5.91%	25.06% ±6.11%
2.c.	33.98% ±5.92%	33.88% ±6%	33.68% ±%5.02	28.38% ±6.88%	23.60% ±5.9%	24.80% ±6.06%

Table 5: Comparison of post-processing approaches, in which "Inde." serves as the new baseline as the best performing model without any post-processing.

Approaches	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactics						
Inde.	65.64% ±3.76%	64.69% ±3%	65.38% ±2.87%	60.26% ±3.2%	58.50% ±3.68%	59.47% ±2.29%
1.a.	**100% ±0%**	**100% ±0%**	**100% ±0%**	**100% ±0%**	**100% ±0%**	**100% ±0%**
1.d.	68.09% ±4.02%	72.37% ±2.97%	68.84% ±3.24%	63.32% ±3.1%	66.29% ±4%	63.54% ±2.08%
Techniques						
Inde.	37.18% ±6.75%	29.79% ±5.91%	35.02% ±5.32%	28.84% ±6.9%	22.67% ±5.94%	25.06% ±6.09%
1.b.	**51.54% ±5.01%**	**55.90% ±4.23%**	**52.24% ±4.4%**	**41.05% ±3.94%**	**39.04% ±5.13%**	**38.35% ±3.16%**
1.c.	36.51% ±8%	48.75% ±4.13%	38.14% ±7.54%	34.21% ±7.43%	37.12% ±4.13%	32.36% ±6.24%
1.d.	38.40% ±7%	28.39% ±6.14%	35.44% ±5.42%	29.14% ±7.1%	22.40% ±6.22%	25.23% ±6.37%

Table 6: Comparison of post-processing approaches, in which "Inde." serves as the new baseline as the best performing model without any post-processing, with the prediction of tactics using perfect techniques classification and the prediction of techniques using perfect tactics classification.

Table 6 would probably be better in the case of approach *1.d. hanging node* and the independent classification. However, Table 6 proves that approach *1.d. hanging node*, in theory, helps to improve the prediction, despite not improving the results on the current dataset.

In the case of the techniques prediction, the independent classification presented a relatively low performance. The use of the approaches *1.c. confidence propagation* and *1.d. hanging node* is a possible improvement. In all cases, except for the approach *1.d. hanging node*, the $F_{0.5}$ score changes mainly due to the lowering of the precision and the increase of the recall, due to the addition of techniques to the predicted set. Thus the $F_{0.5}$ score increasing in *1.d. hanging node* is especially interesting, as it is due to the decrease of false-positive and, thus, the increase of the precision.

The approaches relying on the relationship between techniques are close to the classification without post-processing. We can conclude from these results that they are not adapted to our problem, or to the data we currently have. The approach *2.a. rare association rules* could probably fit better in a hierarchical environment with known conditional probabilities. It is also possible that the ground truth we used is incomplete, as, even though it is based on data collected and analysed by experts, it is representing only a sample of the malware, tools and campaigns of the past years.

If the tactics prediction were to become perfect, without the prediction of techniques changing, the approach *1.b. tactics as features* would have the highest evaluation for all metrics. Once again, we cannot choose an approach based on this test and the results from Table 6, since the techniques prediction would probably improve at the same time as the tactics prediction. However, because all approaches are better than the independent classification, we can conclude that they are helpful for the prediction. However, the fact that the approach *1.d. hanging node* is barely higher than the independent classification compared to approach *1.c. confidence propagation*, while it was the opposite on the current dataset, can let us question how *1.d. hanging node* and *1.c. confidence propagation* would behave if the dataset was modified and if improvements in the classification were made. *1.d. Hanging node* also presents worse results when the tactics prediction is perfect, meaning that it could rely on wrong prediction of the tactics to luckily find improving results for the techniques predictions. Thus, it would be more likely for the *1.c. confidence propagation* approach to present better results than the *1.d. hanging node* approach with an improved model.

While the *1.d. hanging node* and *1.c. confidence propagation* approaches are very different in their strategy, by observing their behaviours on the dataset (see Appendix XI-E.1), we can see that they tend to target different techniques. *1.d. Hanging node* seems to affect very little, but efficiently, the results from the independent classification and only modifies the results of the techniques with the highest number of reports. *1.c. Confidence propagation* targets any techniques and modify many predictions, which would potentially improve the classification for the hard-to-predict techniques. However, from our observation, this approach is right almost as often as it is wrong, thus worsening the prediction of some labels and improving the prediction of others. Replotting similar graphs as the ones in Section VII-C, to observe the correlation between the number of reports and the model performance for each technique, with the post-processing methods applied, we do not see major differences (see Appendix XI-E.2). The correlation coefficient is slightly higher for the *1.c. confidence propagation*, but still not high enough to demonstrate a linear correlation. As for the independent classification, there is no proof augmenting the size of the training set will positively impact the predictions of the techniques using these methods. Thus, we can conclude similarly to the Section VII. Some techniques might be harder to predict than others, and the quality of our dataset might not be good enough to predict some of the techniques, even when using a post-processing method.

The final choice of post-processing method for our classification is difficult as we cannot predict exactly how they would react if the dataset were to evolve. Based on the analysis of the results, it is better to not use any post-processing method on the tactics classification. In the case of the techniques prediction, the choice to be made is between the approaches *1.c. confidence propagation* and *1.d. hanging node*. In the current implementation of the tool, we use the approach *1.d. hanging node* as it is the one performing the best on the current dataset. However, since the user of our tool can retrain the model with new data, the approach *1.c. confidence propagation* is also implemented, and they are compared at each retraining to select the best method with the current dataset.

IX. rcATT: Tool Implementation

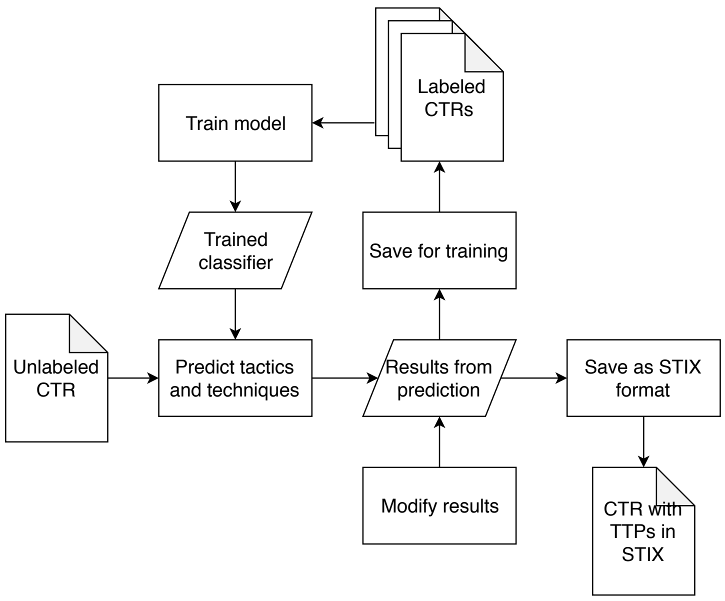
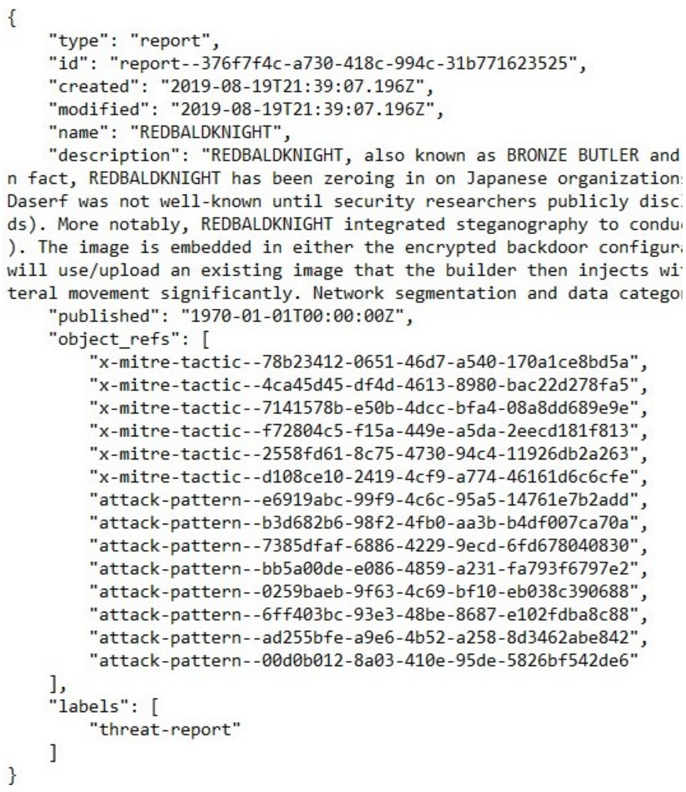


Fig. 6: Organisation of rcATT.

The goal of our research was to implement a tool, so our classification model could be used easily. To do so, we created a simple tool: rcATT, for “reports classification by adversarial tactics and techniques”. Based on the conclusions from the previous sections, the classification is done using a

TF-IDF weighted bag-of-words text representation and a binary relevance Linear SVC. It is followed in post-processing by either the hanging node approach or the confidence propagation approach, depending on the current training set.

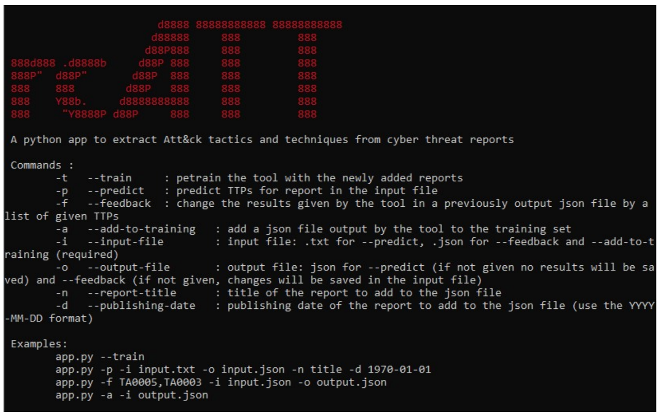


```
{
    "type": "report",
    "id": "report--376f7f4c-a730-418c-994c-31b771623525",
    "created": "2019-08-19T21:39:07.196Z",
    "modified": "2019-08-19T21:39:07.196Z",
    "name": "REDBALDKNIGHT",
    "description": "REDBALDKNIGHT, also known as BRONZE BUTLER and
n fact, REDBALDKNIGHT has been zeroing in on Japanese organization
Daserf was not well-known until security researchers publicly disc
ds). More notably, REDBALDKNIGHT integrated steganography to condu
). The image is embedded in either the encrypted backdoor configur
will use/upload an existing image that the builder then injects wi
teral movement significantly. Network segmentation and data catego
    "published": "1970-01-01T00:00:00Z",
    "object_refs": [
        "x-mitre-tactic--78b23412-0651-46d7-a540-170a1ce8bd5a",
        "x-mitre-tactic--4ca45d45-df4d-4613-8980-bac22d278fa5",
        "x-mitre-tactic--7141578b-e50b-4dcc-bfa4-08a8dd689e9e",
        "x-mitre-tactic--f72804c5-f15a-449e-a5da-2eecd181f813",
        "x-mitre-tactic--2558fd61-8c75-4730-94c4-11926db2a263",
        "x-mitre-tactic--d108ce10-2419-4cf9-a774-46161d6c6cfe",
        "attack-pattern--e6919abc-99f9-4c6c-95a5-14761e7b2add",
        "attack-pattern--b3d682b6-98f2-4fb0-aa3b-b4df007ca70a",
        "attack-pattern--7385dfaf-6886-4229-9ecd-6fd678040830",
        "attack-pattern--bb5a00de-e086-4859-a231-fa793f6797e2",
        "attack-pattern--0259baeb-9f63-4c69-bf10-eb038c390688",
        "attack-pattern--6ff403bc-93e3-48be-8687-e102fdba8c88",
        "attack-pattern--ad255bfe-a9e6-4b52-a258-8d3462abe842",
        "attack-pattern--00d0b012-8a03-410e-95de-5826bf542de6"
    ],
    "labels": [
        "threat-report"
    ]
}
```

Fig. 7: Example of a report in STIX format output by rcATT.

Figure 6 illustrates the functioning of the tool. From a cyber threat report, in a text format, entered by the user, our trained model predicts different tactics and techniques. The user can view the results and the confidence score associated with each prediction, either based on the Min-Max scaling of the decision function score from the classifier or the reevaluation score, based on the post-processing method. In case the user does not agree with the results, they can change them. Regardless, the results can be saved with the other labelled cyber threat reports to be used to train the classifier again. The retraining of the classifier has to be activated manually by the user, as automated retraining might slow down the tools in unwanted moments and avoids training when no new data has been added to the training set. In addition to retraining the classifier with the new data, this functionality also defines which post-processing method improves the most the predictions, as well as which hanging node thresholds are the best, with this dataset.

The results can also be saved in a JSON file in STIX format, as seen in Figure 7, so it can easily be accessed for other uses. The predictions are exported in a "report" object for which the "name" and "published" date are left for the user to precise (or are given default values). The "description" is the report given by the user for the prediction, and the results of the prediction, or the changes made by the user, are saved in the "object_refs", under the identification number from the STIX version of the ATT&CK framework [40].



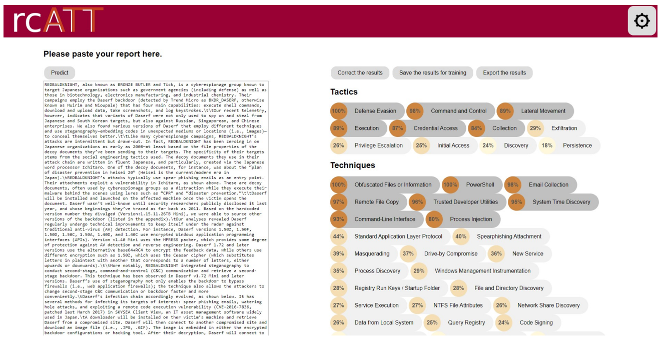
```
                                    d8888 88888888888 88888888888
                                   d88888     888         888
                                  d88P888     888         888
888d888 .d8888b                  d88P 888     888         888
888P"   d88P"                   d88P  888     888         888
888     888                    d88P   888     888         888
888     Y88b.                 d8888888888     888         888
888      "Y8888P             d88P     888     888         888

 A python app to extract Att&ck tactics and techniques from cyber threat reports

 Commands :
        -t    --train      : petrain the tool with the newly added reports
        -p    --predict    : predict TTPs for report in the input file
        -f    --feedback   : change the results given by the tool in a previously output json file by a
list of given TTPs
        -a    --add-to-training    : add a json file output by the tool to the training set
        -i    --input-file         : input file: .txt for --predict, .json for --feedback and --add-to-t
raining (required)
        -o    --output-file        : output file: json for --predict (if not given no results will be sa
ved) and --feedback (if not given, changes will be saved in the input file)
        -n    --report-title       : title of the report to add to the json file
        -d    --publishing-date    : publishing date of the report to add to the json file (use the YYYY
-MM-DD format)

 Examples:
        app.py --train
        app.py -p -i input.txt -o input.json -n title -d 1970-01-01
        app.py -f TA0005,TA0003 -i input.json -o output.json
        app.py -a -i output.json
```

Fig. 8: rcATT commmand-line interface.



[illegible]

Tactics

100% Defense Evasion · 98% Command and Control · 89% Lateral Movement
89% Execution · 87% Credential Access · 84% Collection · 29% Exfiltration
26% Privilege Escalation · 25% Initial Access · 24% Discovery · 18% Persistence

Techniques

100% Obfuscated Files or Information · 100% PowerShell · 98% Email Collection
97% Remote File Copy · 96% Trusted Developer Utilities · 95% System Time Discovery
93% Command-Line Interface · 80% Process Injection
44% Standard Application Layer Protocol · 40% Spearphishing Attachment
39% Masquerading · 37% Drive-by Compromise · 36% New Service
35% Process Discovery · 29% Windows Management Instrumentation
28% Registry Run Keys / Startup Folder · 28% File and Directory Discovery
27% Service Execution · 27% NTFS File Attributes · 26% Network Share Discovery
26% Data from Local System · 25% Query Registry · 24% Code Signing

Fig. 9: rcATT graphical interface.

It has an interface in command-line and a graphical interface, having each their perks and drawbacks, despite the same functionalities. The command-line version is practical to use when the user might want to predict TTPs from a large number of CTRs (see Figure 8). However, if the user wants to change the results of the classification and save it to the training set, as a way to give feedback about the classifier, and increase the training data, the graphical interface version is easier to use (see Figure 9).

X. Comparison with similar work

We have mentioned several similar studies in the related work section of this paper, and the ones closest to our implementation are TTPDrill [30] and Ayoade *et al.*'s work [35]. However, there are many differences between our work and theirs. First of all, we use an approach different from TTPDrill [30], as we use text classification. However, we create a tool and output our results in a structured manner, as TTPDrill [30], but with a feedback function to continuously improve the tool. Though Ayoade *et al.* [35] does not create a tool, our classification approach is similar to theirs, as we separate the classification between tactics and techniques and use a post-processing approach to improve the predictions. Although, we do not use a bias correction technique, preferring to use a training set with multiple sources, as opposed to their unique source sets, and the chosen classifiers and post-processing approaches are different from theirs.

However, it is difficult to compare these previous works to our tool as we do not have access to their code or dataset, and the description given in their paper is too limited to be implemented precisely. Solely the tool Unfetter Insight, available on Github [79], can be compared to ours in concept and results. This tool has been developed to detect possible ATT&CK techniques, or tactics, in reports in TXT, PDF or HTML format. Based on our understanding of the code, as no paper is available to describe their approach, the detection is done by the Babelfish software created by W. Kinsman in 2017, which learns topics from a wiki-like corpus and retrieve the presence of those topics in a document. The tool creates convolutions from the text, from which are only kept the convolutions having a majority of common vocabulary like the one present in the corpus. Based on the term frequency of the vocabulary in the convolutions left, the techniques are predicted using a binary relevance Multinomial Naive Bayes classifier, and a list of techniques, or tactics, is output for the user to read.

Tools	Micro			Macro		
	Precision	Recall	$F_{0.5}$	Precision	Recall	$F_{0.5}$
Tactics						
rcATT	**79.31%**	12.20%	**37.75%**	**81.73%**	8.21%	**23.23%**
Unfetter insight	19.09%	**26.76%**	20.25%	19.17%	**23.14%**	19.28%
Techniques						
rcATT	**72.22%**	**2.07%**	**9.30%**	**20.60%**	**4.33%**	**10.11%**
Unfetter insight	3.54%	1.11%	2.46%	1.31%	0.64%	0.74%

Table 7: Comparison between rcATT and Unfetter insight ATT&CK tactics and techniques prediction function, using the same training and testing sets.

As it is the only tool we have access to, we decided on comparing it with our tool. However, the premises of the models are not the same, since we base our classification on a training set of labelled reports, and they use a dictionary-like learning set. We could not add our training set without compromising the concept behind Babelfish itself; thus, we decided to use only their training set to test both approaches. It is to be noted that their wiki-like corpus is, in fact, the content of the ATT&CK website with the techniques definitions, which was already part of our dataset.

Using the same training set and testing set, we obtain the results presented in Table 7. Compared to our tool, Unfetter Insight tactics and techniques predictions module underperforms, mainly because of the higher precision of our tool. Basing the classification purely on the definition of the technique is partly at fault, as even our tool underperforms compared to our original training set.

XI. Conclusions, limitations and future work

The goal of our research was the creation of a tool to retrieve tactics and techniques from the ATT&CK framework. Based on Ayoade *et al.* [35] and Husari *et al.*'s works [30] on the same topic, we decided on approaching the problem using text classification, which led us to investigate which classification model would be the best for our problem. After comparing different classifiers, problem transformations to fit our multi-label problem, and text representations, we defined that the best model for our problem would use a TF-IDF weighted bag-of-words as text representation and a binary relevance Linear SVC. If this model is the best performing among the ones tested, it is overfitting and working on an imbalanced dataset. We thus looked for ways to avoid this imbalance and studied the impact of the dataset size on the different predictions. The resampling methods we tried did not improve our classification, although we established that increasing the training data would improve the performance of our classifier for tactics prediction. By adding more data, we would be able to reach an 80% macro-averaged $F_{0.5}$ score. Studying the $F_{0.5}$ score for each technique based on the change of training set size, we established that some techniques were much harder than others to predict and would not improve much if more data were added to their classification. We invoke the possibility that we are working on a low-quality dataset, at least for some of the techniques. In order to improve the results, we studied various ways to link the labels in post-processing methods based on ground truth knowledge. We defined that two approaches, *hanging node* and *confidence propagation*, in which the tactics prediction influence the techniques prediction, are able to improve the reports classification by techniques. However, as we believe their performance will evolve depending on the training dataset, we decide on implementing both in our tool: rcATT.

While we were able to find a model fitting our problem and to create a tool out of it, our work has some limitations. The main issue we faced from the beginning was the limited amount of data we could collect, as we did not have the resources to label enough reports for this project manually. As mentioned in this report, a problem with the dataset that we collected is the fact that we are not able to guaranty that the included reports are of good enough quality to predict all labels.

Limitations in resources and time restricted us in testing different text representation models, like Google's BERT [80], OpenAI's GPT [81] and GPT-2 [82], and other similar methods [83]. Likewise, we limited the test of classifiers to the ones included in Scikit-learn [42], when more recent text classification solutions are available (e.g. neural networks, which are frequently applied to problem similar to ours [84]). We also limited ourselves to try only one resampling method, while others are available [85], but more difficult to apply to our problem. More importantly, because of our constraint in time, we were not able to test all combinations of classifiers, text representation, post-processing and resampling method.

All these limitations are ground for future work by testing these different approaches. It would also be possible to change of general approach and use a similar information retrieval method to the one described in Husari *et al.*'s TTPDrill [30], in order to verify their results and potentially improve their method. Another possibility would be to mix both ideas and to retrieve the threat actions, based on verb-object pairs, from the reports and use it to classify the reports by techniques.

The ATT&CK framework is also bound to change. Modifications of the organisation of the framework are already being presented [68]. There will be in the future three levels: tactics, techniques and sub-techniques. Tactics would seemingly stay the same as the already existing ones, but more general techniques would be created, and more precise sub-techniques would be associated with them. This format would probably facilitate a similar approach to TTPDrill [30], but could also improve a multi-label classification method such as our work. As we have seen, the classification by tactics performs better than by techniques, potentially because they are more general.

REFERENCES

[1] D. Chismon and M. Ruks, "Threat intelligence: Collecting, analysing, evaluating," *MWR InfoSecurity Ltd*, 2015.

[2] D. Bianco, "The pyramid of pain," *Enterprise Detection & Response*, 2013.

[3] S. Barnum, "Standardizing cyber threat intelligence information with the structured threat information expression (stix)," *Mitre Corporation*, vol. 11, pp. 1–22, 2012.

[4] J. Connolly, M. Davidson, and C. Schmidt, "The trusted automated exchange of indicator information (taxii)," *The MITRE Corporation*, pp. 1–20, 2014.

[5] S. Barnum, R. Martin, B. Worrell, and I. Kirillov, "The cybox language specification," *draft, The MITRE Corporation*, 2012.

[6] "Ibm x-force exchange," 2019. [Online]. Available: https://exchange.xforce.ibmcloud.com

[7] "Alienvault open threat exchange," 2019. [Online]. Available: https://otx.alienvault.com

[8] "Mitre att&ck," 2019. [Online]. Available: https://attack.mitre.org/

[9] V. Mulwad, W. Li, A. Joshi, T. Finin, and K. Viswanathan, "Extracting information about security vulnerabilities from web text," in *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 03*. IEEE Computer Society, 2011, pp. 257–260.

[10] M. Almukaynizi, E. Marin, E. Nunes, P. Shakarian, G. I. Simari, D. Kapoor, and T. Siedlecki, "Darkmention: A deployed system to predict enterprise-targeted external cyberattacks," in *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2018, pp. 31–36.

[11] J. Undercoffer, A. Joshi, and J. Pinkston, "Modeling computer attacks: An ontology for intrusion detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 113–135.

[12] S. More, M. Matthews, A. Joshi, and T. Finin, "A knowledge-based approach to intrusion detection modeling," in *2012 IEEE Symposium on Security and Privacy Workshops*. IEEE, 2012, pp. 75–81.

[13] N. McNeil, R. A. Bridges, M. D. Iannacone, B. Czejdo, N. Perez, and J. R. Goodall, "Pace: Pattern accurate computationally efficient bootstrapping for timely discovery of cyber-security concepts," in *2013 12th International Conference on Machine Learning and Applications*, vol. 2. IEEE, 2013, pp. 60–65.

[14] C. L. Jones, R. A. Bridges, K. M. Huffer, and J. R. Goodall, "Towards a relation extraction framework for cyber-security concepts," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*. ACM, 2015, p. 11.

[15] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "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*. ACM, 2016, pp. 755–766.

[16] A. Panwar, "igen: Toward automatic generation and analysis of indicators of compromise (iocs) using convolutional neural network," *Arizona state university, Arizona*, 2017.

[17] Z. Zhu and T. Dumitras, "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, pp. 458–472.

[18] S. Zhou, Z. Long, L. Tan, and H. Guo, "Automatic identification of indicators of compromise using neural-based sequence labelling," *CoRR*, vol. abs/1810.10156, 2018. [Online]. Available: http://arxiv.org/abs/1810.10156

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," *arXiv preprint arXiv:1603.01360*, 2016.

[20] F. Dernoncourt, J. Y. Lee, O. Uzuner, and P. Szolovits, "De-identification of patient notes with recurrent neural networks," *Journal of the American Medical Informatics Association*, vol. 24, no. 3, pp. 596–606, 2017.

[21] S. Samtani, R. Chinn, H. Chen, and J. F. Nunamaker Jr, "Exploring emerging hacker assets and key hackers for proactive cyber threat intelligence," *Journal of Management Information Systems*, vol. 34, no. 4, pp. 1023–1053, 2017.

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

[23] I. Deliu, C. Leichter, and K. Franke, "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, pp. 5008–5013.

[24] Z. Zhu and T. Dumitraş, "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*. ACM, 2016, pp. 767–778.

[25] R. R. Ramnani, K. Shivaram, S. Sengupta *et al.*, "Semi-automated information extraction from unstructured threat advisories," in *Proceedings of the 10th Innovations in Software Engineering Conference*. ACM, 2017, pp. 181–187.

[26] M. Thelen and E. Riloff, "A bootstrapping method for learning semantic lexicons using extraction pattern contexts," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics, 2002, pp. 214–221.

[27] E. Riloff, "Automatically generating extraction patterns from untagged text," in *Proceedings of the national conference on artificial intelligence*, 1996, pp. 1044–1049.

[28] M. M. Deza and E. Deza, "Encyclopedia of distances," in *Encyclopedia of Distances*. Springer, 2009, pp. 1–583.

[29] D. Lin *et al.*, "An information-theoretic definition of similarity." in *Icml*, vol. 98, no. 1998. Citeseer, 1998, pp. 296–304.

[30] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 2017, pp. 103–115.

[31] "Mitre capec," 2019. [Online]. Available: https://capec.mitre.org/

[32] V. Mavroeidis and S. Bromander, "Cyber threat intelligence model: an evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence," in *2017 European Intelligence and Security Informatics Conference (EISIC)*. IEEE, 2017, pp. 91–98.

[33] L. Martin, "https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html," 2019.

[34] G. Husari, X. Niu, B. Chu, and E. Al-Shaer, "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, pp. 1–6.

[35] G. Ayoade, S. Chandra, L. Khan, K. Hamlen, and B. Thuraisingham, "Automated threat report classification over multi-source data," in *2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2018, pp. 236–245.

[36] J. Huang, A. Gretton, K. Borgwardt, B. Schölkopf, and A. J. Smola, "Correcting sample selection bias by unlabeled data," in *Advances in neural information processing systems*, 2007, pp. 601–608.

[37] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe, "Direct importance estimation for covariate shift adaptation," *Annals of the Institute of Statistical Mathematics*, vol. 60, no. 4, pp. 699–746, 2008.

[38] T. Kanamori, S. Hido, and M. Sugiyama, "A least-squares approach to direct importance estimation," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1391–1445, 2009.

[39] "Misp galaxy clusters," 2019. [Online]. Available: https://www.misp-project.org/galaxy.html

[40] MITRE, "Att&ck framework github," 2019. [Online]. Available: https://github.com/mitre/cti

[41] H. Bast and C. Korzen, "A benchmark and evaluation for text extraction from pdf," in *Proceedings of the 17th ACM/IEEE Joint Conference on Digital Libraries*. IEEE Press, 2017, pp. 99–108.

[42] "api reference scikit-learn 0.21.3 documentation," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

[43] C. D. Manning, P. Raghavan, and H. Schütze, "Scoring, term weighting and the vector space model," *Introduction to information retrieval*, vol. 100, pp. 2–4, 2008.

[44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[45] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 667–685.

[46] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *In Proceedings of the ACL Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics. Philadelphia: Association for Computational Linguistics*. [Online]. Available: https://www.nltk.org/

[47] 2013. [Online]. Available: https://code.google.com/archive/p/word2vec/

[48] O. Luaces, J. Díez, J. Barranquero, J. J. del Coz, and A. Bahamonde, "Binary relevance efficacy for multilabel classification," *Progress in Artificial Intelligence*, vol. 1, no. 4, pp. 303–313, 2012.

[49] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine learning*, vol. 85, no. 3, p. 333, 2011.

[50] N. SpolaôR, E. A. Cherman, M. C. Monard, and H. D. Lee, "A comparison of multi-label feature selection methods using the problem transformation approach," *Electronic Notes in Theoretical Computer Science*, vol. 292, pp. 135–151, 2013.

[51] Scikit-learn.org, "https://scikit-learn.org/stable/modules/multiclass.html," 2019.

[52] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[53] "Towards data science," 2019. [Online]. Available: https://towardsdatascience.com/

[54] "Kaggle.com," 2019. [Online]. Available: https://www.kaggle.com/

[55] "Logistic regression - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

[56] "Perceptron - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html#sklearn.linear_model.Perceptron

[57] "Ridge classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.RidgeClassifier.html#sklearn.linear_model.RidgeClassifier

[58] "Bernoulli nb - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.BernoulliNB.html#sklearn.naive_bayes.BernoulliNB

[59] "Kneighbors classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

[60] "Linear svc - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html#sklearn.svm.LinearSVC

[61] "Decision tree classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

[62] "Extra tree classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeClassifier.html#sklearn.tree.ExtraTreeClassifier

[63] "Random forest classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

[64] "Extra trees classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier

[65] "Adaboost classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html#sklearn.ensemble.AdaBoostClassifier

[66] "Bagging classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html#sklearn.ensemble.BaggingClassifier

[67] "Gradient boosting classifier - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier

[68] B. Strom, "Att&ck sub-techniques preview," 2019. [Online]. Available: https://medium.com/mitre-attack/attack-sub-techniques-preview-b79ff0ba669a

[69] "pearson correlation coefficient - wikipedia," 2019. [Online]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

[70] Y. Wu, B. L. Tseng, and J. R. Smith, "Ontology-based multi-classification learning for video concept detection," in *2004 IEEE International Conference on Multimedia and Expo (ICME) (IEEE Cat. No.04TH8763)*, vol. 2, June 2004, pp. 1003–1006 Vol.2.

[71] "Minmax-scaler - scikit-learn," 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[72] F. Benites and E. Sapozhnikova, "Improving multi-label classification by means of cross-ontology association rules," *bioinformatics*, vol. 2, no. 9, p. 20, 2015.

[73] S. Kulczynski, "Die pflanzenassoziationen der pieninen," in *Bulletin International de l'Academie Polonaise des Sciences et des Lettres, Classe des Sciences Mathematiques et Naturelles B*, 1927, pp. 57–203.

[74] A. Soni, A. Pappu, J. C.-m. Ni, and T. Chevalier, "Post-processing techniques for improving predictions of multilabel learning approaches," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2017, pp. 61–66.

[75] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees," *SIAM Journal on Applied Mathematics*, vol. 16, no. 1, pp. 1–29, 1968.

[76] K. Mehlhorn, "A faster approximation algorithm for the steiner problem in graphs," *Information Processing Letters*, vol. 27, no. 3, pp. 125–128, 1988.

[77] J. Edmonds, "Optimum branchings," *Journal of Research of the national Bureau of Standards B*, vol. 71, no. 4, pp. 233–240, 1967.

[78] S. Martello, "Knapsack problems: algorithms and computer implementations," *Wiley-Interscience series in discrete mathematics and optimiza tion*, 1990.

[79] Unfetter, "Unfetter insight," 2018. [Online]. Available: https://github.com/unfetter-discover/unfetter-insight

[80] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[81] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[82] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI Blog*, vol. 1, no. 8, 2019.

[83] “Hugging face’s transformers,” 2019. [Online]. Available: https://github.com/huggingface/transformers

[84] 2019. [Online]. Available: https://github.com/Tencent/NeuralNLP-NeuralClassifier

[85] G. Lemaître, F. Nogueira, and C. K. Aridas, “Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning,” *Journal of Machine Learning Research*, vol. 18, no. 17, pp. 1–5, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-365.html

Appendix

A. Analysis of the number of reports by tactics and techniques

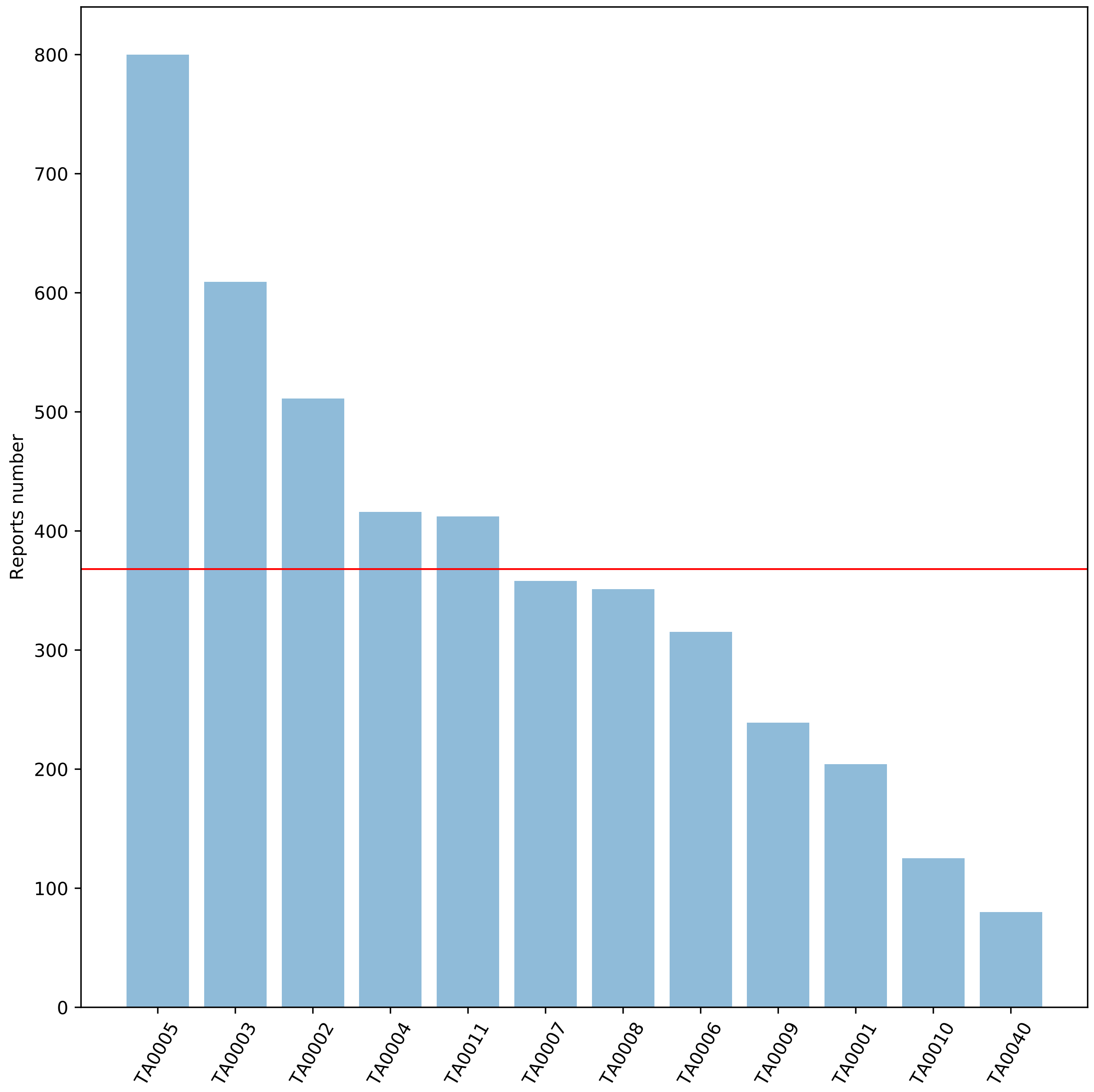


Fig. 1: Number of reports for each tactics in ATT&CK's Github dataset

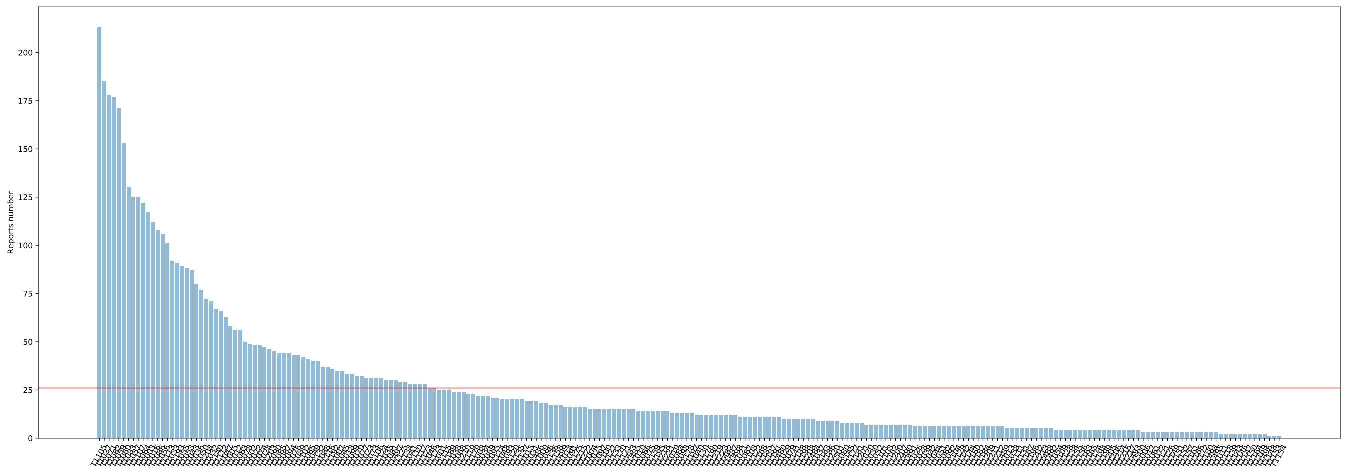


Fig. 2: Number of reports for each techniques in ATT&CK's Github dataset

B. *Results from initial classification by tactics and techniques*

1) *Evaluation of tactics predictions by model*

	Micro			Macro		
	Precision	Recall	F0.5	Precision	Recall	F0.5
TF						
Classifier Chain						
AdaBoost	64.73%	50.84%	61.30%	60.87%	45.20%	56.45%
Bagging	67.19%	40.57%	59.38%	64.01%	33.46%	51.89%
ExtraTrees	62.78%	27.87%	50.05%	54.90%	20.89%	37.82%
Gradient Boosting	71.84%	44.33%	63.61%	66.72%	37.42%	55.91%
RandomForest	65.18%	29.58%	52.35%	56.35%	21.96%	38.65%
Logistic Regression	63.81%	54.35%	61.61%	58.86%	47.78%	55.85%
Perceptron	56.34%	56.58%	56.35%	51.63%	50.79%	51.04%
RidgeClassifier	53.21%	52.66%	53.05%	47.79%	48.27%	47.70%
BernoulliNB	**80.20%**	6.13%	23.25%	52.02%	3.73%	12.36%
KNN	57.92%	38.67%	52.54%	52.54%	30.40%	43.06%
LinearSVC	63.81%	51.02%	60.71%	58.89%	44.31%	54.64%
DecisionTree	55.28%	48.88%	53.80%	50.80%	44.57%	49.13%
ExtraTree	51.68%	38.15%	48.20%	44.29%	30.71%	39.59%
Binary Relevance						
AdaBoost	62.30%	49.27%	59.08%	57.26%	42.37%	52.91%
Bagging	68.26%	42.36%	60.75%	63.71%	34.30%	51.54%
ExtraTrees	63.20%	30.33%	51.81%	56.74%	22.58%	39.61%
Gradient Boosting	71.42%	48.19%	65.04%	66.35%	40.16%	56.78%
RandomForest	63.30%	29.63%	51.32%	53.08%	21.76%	37.39%
Logistic Regression	63.71%	54.42%	61.54%	58.74%	47.81%	55.76%
Perceptron	55.99%	57.45%	56.25%	51.45%	52.36%	51.38%
RidgeClassifier	61.85%	51.46%	59.39%	55.64%	45.31%	52.92%
BernoulliNB	56.36%	32.98%	49.20%	47.43%	27.13%	40.51%
KNN	57.93%	38.66%	52.55%	52.55%	30.41%	43.06%
LinearSVC	64.70%	51.55%	61.51%	59.20%	44.36%	54.89%
DecisionTree	54.85%	49.20%	53.57%	50.59%	44.77%	49.10%
ExtraTree	51.43%	37.91%	47.96%	44.70%	30.46%	39.77%
Adapted algorithms						
ExtraTrees	63.83%	26.90%	49.90%	54.95%	18.61%	33.62%
RandomForest	64.03%	27.09%	50.00%	49.46%	18.40%	32.32%
KNN	57.93%	38.66%	52.55%	52.55%	30.41%	43.06%
DecisionTree	48.29%	46.83%	47.97%	41.75%	40.38%	41.38%
ExtraTree	44.07%	39.98%	43.15%	36.72%	32.96%	35.70%
TFIDF						
Classifier Chain						
AdaBoost	61.42%	49.86%	58.59%	57.71%	44.09%	53.79%
Bagging	68.62%	34.73%	56.97%	64.51%	28.47%	47.85%
ExtraTrees	66.03%	25.24%	49.77%	54.42%	17.68%	33.33%
Gradient Boosting	71.15%	43.15%	62.52%	67.40%	36.29%	54.97%

RandomForest	62.71%	27.73%	49.93%	53.95%	19.92%	35.29%
Logistic Regression	74.37%	18.65%	43.35%	43.55%	12.48%	24.93%
Perceptron	62.06%	54.97%	60.28%	58.05%	49.26%	54.72%
RidgeClassifier	74.40%	41.27%	63.27%	**67.63%**	33.31%	51.74%
BernoulliNB	55.71%	38.41%	50.92%	47.54%	32.05%	42.67%
KNN	56.86%	36.69%	51.12%	53.10%	30.89%	44.76%
LinearSVC	71.63%	44.89%	63.41%	65.70%	36.76%	54.59%
DecisionTree	48.66%	46.38%	48.15%	43.26%	40.75%	42.58%
ExtraTree	44.60%	37.21%	42.77%	38.73%	31.18%	36.59%
Binary Relevance						
AdaBoost	61.02%	51.02%	58.61%	56.61%	44.67%	53.19%
Bagging	66.88%	41.44%	59.39%	63.92%	34.95%	52.29%
ExtraTrees	64.22%	27.17%	50.29%	56.84%	19.69%	35.94%
Gradient Boosting	70.13%	46.85%	63.66%	65.08%	38.94%	55.03%
RandomForest	65.57%	27.81%	51.32%	57.89%	19.58%	35.67%
Logistic Regression	71.04%	50.70%	65.61%	59.00%	40.53%	51.21%
Perceptron	65.20%	55.35%	62.80%	60.54%	48.29%	56.24%
RidgeClassifier	72.40%	48.90%	**65.83%**	66.57%	38.58%	53.32%
BernoulliNB	55.78%	38.41%	50.97%	47.60%	32.00%	42.67%
KNN	62.45%	45.24%	57.89%	57.82%	36.75%	49.63%
LinearSVC	65.64%	**64.69%**	65.38%	60.26%	58.50%	**59.47%**
DecisionTree	49.33%	47.16%	48.80%	45.02%	43.01%	44.43%
ExtraTree	45.31%	38.61%	43.74%	39.62%	32.50%	37.63%
Adapted algorithms						
ExtraTrees	65.21%	23.40%	47.69%	56.20%	15.72%	30.71%
RandomForest	63.99%	24.45%	48.07%	47.71%	16.03%	29.17%
KNN	62.45%	45.24%	57.89%	57.82%	36.75%	49.63%
DecisionTree	46.75%	42.85%	45.87%	40.84%	36.94%	39.84%
ExtraTree	41.79%	36.64%	40.62%	35.46%	30.63%	34.13%
w2v avg						
Classifier Chain						
AdaBoost	58.59%	44.21%	54.98%	52.69%	36.95%	47.34%
Bagging	62.47%	34.95%	53.75%	54.41%	28.18%	44.15%
ExtraTrees	64.21%	31.81%	52.97%	56.22%	24.33%	41.17%
Gradient Boosting	63.33%	42.22%	57.58%	55.10%	33.38%	47.52%
RandomForest	65.17%	32.87%	54.35%	54.26%	24.39%	40.27%
Logistic Regression	62.80%	34.15%	53.78%	55.27%	26.87%	42.63%
Perceptron	47.15%	50.29%	47.75%	44.62%	48.35%	43.38%
RidgeClassifier	67.80%	33.90%	56.19%	58.81%	25.66%	41.44%
BernoulliNB	42.00%	62.59%	44.92%	39.46%	62.38%	41.82%
KNN	53.40%	51.78%	52.93%	48.15%	43.84%	45.79%
LinearSVC	62.88%	40.82%	56.75%	59.98%	34.04%	49.66%
DecisionTree	45.16%	47.04%	45.48%	38.94%	40.94%	39.11%

ExtraTree	43.52%	43.22%	43.41%	37.84%	37.75%	37.57%
Binary Relevance						
AdaBoost	57.97%	46.24%	55.11%	50.77%	38.54%	46.70%
Bagging	62.30%	37.63%	54.99%	54.08%	29.42%	44.03%
ExtraTrees	63.73%	36.23%	55.20%	55.97%	27.79%	42.62%
Gradient Boosting	64.53%	44.77%	59.23%	55.65%	35.07%	46.78%
RandomForest	64.90%	36.36%	56.03%	58.19%	27.65%	43.52%
Logistic Regression	66.86%	41.94%	59.68%	61.02%	31.95%	46.17%
Perceptron	54.09%	42.20%	50.58%	53.29%	36.27%	40.79%
RidgeClassifier	68.06%	39.26%	59.26%	60.04%	28.61%	42.06%
BernoulliNB	43.35%	62.57%	46.14%	40.10%	61.73%	42.42%
KNN	58.08%	51.48%	56.58%	51.80%	41.86%	47.45%
LinearSVC	64.85%	44.91%	59.49%	56.57%	35.35%	47.81%
DecisionTree	46.08%	46.52%	46.13%	40.20%	40.90%	40.07%
ExtraTree	44.22%	46.17%	44.56%	37.29%	39.00%	37.49%
Adapted algorithms						
ExtraTrees	63.11%	35.16%	54.30%	51.38%	26.32%	39.88%
RandomForest	65.44%	35.46%	55.81%	56.18%	26.21%	40.60%
KNN	57.33%	51.03%	55.92%	52.11%	41.78%	47.43%
DecisionTree	44.68%	46.18%	44.93%	38.79%	40.74%	38.96%
ExtraTree	42.24%	45.75%	42.86%	36.52%	39.83%	36.98%
w2v sum						
Classifier Chain						
AdaBoost	60.26%	43.49%	55.82%	54.81%	36.87%	48.68%
Bagging	62.71%	36.65%	54.65%	53.82%	29.42%	44.40%
ExtraTrees	61.29%	29.94%	50.54%	54.15%	22.87%	39.94%
Gradient Boosting	60.21%	40.00%	54.69%	50.19%	30.78%	43.26%
RandomForest	65.57%	34.24%	55.23%	62.47%	27.04%	45.09%
Logistic Regression	54.83%	50.41%	53.89%	49.18%	42.97%	47.26%
Perceptron	48.97%	38.95%	46.57%	50.73%	30.88%	33.63%
RidgeClassifier	63.90%	26.90%	49.87%	55.71%	21.61%	40.35%
BernoulliNB	42.32%	64.00%	45.32%	39.79%	**63.86%**	42.24%
KNN	57.61%	48.22%	55.39%	54.82%	39.33%	47.90%
LinearSVC	48.11%	40.23%	46.30%	43.50%	35.29%	38.58%
DecisionTree	45.20%	47.68%	45.62%	39.67%	42.39%	40.00%
ExtraTree	41.28%	42.75%	41.52%	35.04%	36.56%	35.17%
Binary Relevance						
AdaBoost	59.75%	45.73%	56.15%	52.96%	38.41%	48.22%
Bagging	61.58%	38.37%	54.84%	52.94%	30.24%	43.90%
ExtraTrees	62.91%	36.36%	54.79%	54.29%	28.15%	43.02%
Gradient Boosting	64.54%	45.72%	59.53%	55.37%	35.96%	47.76%
RandomForest	62.79%	36.37%	54.71%	53.92%	28.10%	43.03%
Logistic Regression	58.71%	48.01%	56.13%	52.07%	42.03%	49.31%

Perceptron	47.35%	50.01%	47.64%	45.91%	42.11%	39.88%
RidgeClassifier	63.90%	33.99%	54.19%	54.90%	26.00%	42.53%
BernoulliNB	43.75%	62.37%	46.48%	40.41%	61.01%	42.68%
KNN	57.70%	48.23%	55.45%	54.14%	39.26%	47.62%
LinearSVC	41.97%	49.21%	42.94%	40.80%	46.08%	38.60%
DecisionTree	45.93%	48.47%	46.36%	40.43%	43.18%	40.72%
ExtraTree	42.99%	44.57%	43.25%	36.92%	38.51%	37.11%
Adapted algorithms						
ExtraTrees	62.54%	36.64%	54.68%	55.45%	28.01%	42.16%
RandomForest	63.59%	36.04%	55.03%	55.85%	27.21%	41.70%
KNN	57.58%	48.08%	55.33%	53.64%	38.95%	47.25%
DecisionTree	45.23%	46.24%	45.31%	38.65%	39.73%	38.63%
ExtraTree	42.90%	44.97%	43.20%	37.01%	39.14%	37.24%

2) *Evaluation of techniques predictions by model*

	Micro			Macro		
	Precision	Recall	F0.5	Precision	Recall	F0.5
TF						
Classifier Chain						
AdaBoost	36.64%	13.27%	27.05%	18.38%	8.98%	13.72%
Bagging	39.95%	5.83%	18.24%	18.50%	7.90%	12.32%
ExtraTrees	28.13%	1.89%	7.38%	6.68%	1.19%	2.84%
Gradient Boosting	29.13%	3.46%	11.81%	8.16%	2.61%	5.69%
RandomForest	27.54%	1.52%	6.20%	3.98%	0.51%	1.46%
Logistic Regression	19.64%	23.45%	20.26%	6.40%	19.53%	7.31%
RidgeClassifier	14.82%	25.59%	16.16%	10.88%	23.40%	11.66%
BernoulliNB	11.70%	19.64%	12.70%	4.23%	7.36%	4.51%
KNN	15.68%	4.99%	11.26%	4.34%	1.89%	2.87%
LinearSVC	14.07%	12.63%	13.71%	4.74%	4.66%	4.47%
DecisionTree	22.44%	17.94%	21.31%	18.64%	14.98%	16.46%
ExtraTree	13.89%	10.58%	13.04%	7.94%	6.07%	6.90%
Binary Relevance						
AdaBoost	35.60%	16.04%	28.56%	18.23%	9.74%	14.23%
Bagging	39.30%	7.63%	21.42%	16.92%	7.53%	11.88%
ExtraTrees	29.41%	2.59%	9.52%	6.93%	1.35%	3.35%
Gradient Boosting	27.60%	12.67%	22.31%	20.25%	12.72%	16.26%
RandomForest	30.37%	2.06%	8.07%	3.90%	0.53%	1.55%
Logistic Regression	30.92%	6.52%	17.54%	9.53%	3.77%	6.32%
Perceptron	17.43%	20.12%	17.88%	11.17%	12.27%	10.69%
RidgeClassifier	14.96%	25.96%	16.33%	10.98%	23.50%	11.74%
BernoulliNB	12.57%	19.06%	13.46%	4.02%	6.62%	4.27%
KNN	37.21%	5.33%	16.88%	6.73%	1.67%	3.87%
LinearSVC	22.87%	19.22%	21.98%	15.36%	11.39%	13.41%
DecisionTree	23.06%	18.53%	21.94%	18.00%	14.76%	16.18%
ExtraTree	14.38%	11.20%	13.59%	6.91%	5.79%	6.18%
Adapted algorithms						
ExtraTrees	31.06%	1.95%	7.64%	6.11%	1.08%	2.74%
RandomForest	34.56%	1.54%	6.49%	5.97%	0.99%	2.94%
KNN	37.21%	5.33%	16.88%	6.73%	1.67%	3.87%
DecisionTree	16.23%	14.06%	15.71%	9.99%	8.27%	9.00%
ExtraTree	12.86%	10.10%	12.19%	6.33%	4.95%	5.55%
TFIDF						
Classifier Chain						
AdaBoost	37.06%	13.06%	26.98%	17.70%	8.44%	13.19%
Bagging	46.77%	5.30%	17.91%	15.17%	5.39%	9.32%
ExtraTrees	25.12%	1.12%	4.74%	4.45%	0.59%	1.60%
Gradient Boosting	29.33%	3.46%	12.02%	5.48%	1.96%	4.16%
RandomForest	23.46%	0.73%	3.22%	2.66%	0.32%	0.88%

Logistic Regression	30.42%	0.78%	3.48%	3.89%	1.31%	2.94%
RidgeClassifier	**82.45%**	1.74%	8.00%	7.65%	3.27%	5.50%
BernoulliNB	11.56%	19.20%	12.54%	4.33%	7.23%	4.56%
KNN	14.58%	1.93%	5.96%	4.18%	1.59%	2.52%
LinearSVC	1.44%	15.04%	16.60%	28.57%	10.46%	5.28%
DecisionTree	23.18%	18.47%	22.02%	18.83%	14.85%	16.77%
ExtraTree	13.30%	9.62%	12.33%	6.68%	5.61%	6.06%
Binary Relevance						
AdaBoost	35.04%	14.77%	27.41%	17.23%	9.05%	13.36%
Bagging	45.58%	6.26%	20.07%	16.12%	6.50%	10.58%
ExtraTrees	31.62%	1.80%	7.30%	6.13%	0.97%	2.56%
Gradient Boosting	25.85%	11.60%	20.72%	18.83%	12.09%	15.21%
RandomForest	33.15%	1.42%	5.98%	4.23%	0.50%	1.45%
Logistic Regression	52.82%	3.66%	14.33%	7.86%	2.76%	5.18%
Perceptron	30.45%	18.26%	26.82%	21.89%	14.61%	18.32%
RidgeClassifier	66.85%	3.70%	15.13%	11.48%	4.08%	7.34%
BernoulliNB	12.57%	19.06%	13.46%	4.02%	6.62%	4.27%
KNN	36.04%	2.59%	9.88%	6.13%	1.34%	3.17%
LinearSVC	37.18%	29.79%	**35.02%**	**28.84%**	22.67%	**25.06%**
DecisionTree	20.72%	18.31%	20.15%	16.88%	14.57%	15.55%
ExtraTree	13.60%	9.62%	12.52%	7.37%	5.57%	6.48%
Adapted algorithms						
ExtraTrees	24.47%	1.06%	4.46%	3.64%	0.57%	1.52%
RandomForest	30.64%	1.23%	5.31%	5.03%	0.49%	1.74%
KNN	36.04%	2.59%	9.88%	6.13%	1.34%	3.17%
DecisionTree	14.11%	12.19%	13.65%	8.14%	7.19%	7.44%
ExtraTree	12.54%	9.83%	11.85%	5.70%	4.68%	5.15%
w2v avg						
Classifier Chain						
AdaBoost	29.88%	9.29%	20.67%	9.35%	4.05%	6.70%
Bagging	27.07%	2.70%	9.53%	4.36%	1.16%	2.41%
ExtraTrees	36.45%	1.65%	6.89%	3.87%	0.61%	1.60%
Gradient Boosting	34.51%	7.49%	19.98%	3.61%	0.54%	1.67%
RandomForest	39.05%	2.06%	8.42%	2.99%	0.46%	1.20%
Logistic Regression	38.14%	1.86%	7.94%	3.15%	0.12%	0.07%
RidgeClassifier	35.00%	0.14%	0.67%	0.58%	0.04%	0.15%
BernoulliNB	3.38%	64.05%	4.17%	3.31%	40.41%	3.97%
KNN	35.65%	3.62%	12.20%	5.19%	1.84%	3.15%
LinearSVC	36.21%	3.33%	11.70%	7.58%	3.60%	5.35%
DecisionTree	10.47%	14.60%	11.08%	4.80%	6.09%	4.77%
ExtraTree	11.01%	12.84%	11.33%	4.63%	5.19%	4.46%
Binary Relevance						
AdaBoost	26.88%	11.79%	21.36%	9.15%	4.82%	7.31%

Bagging	30.20%	4.89%	14.81%	5.13%	1.49%	2.99%
ExtraTrees	35.89%	3.67%	12.89%	4.55%	1.10%	2.41%
Gradient Boosting	20.36%	7.55%	15.16%	6.57%	2.81%	4.35%
RandomForest	38.86%	3.70%	13.33%	3.89%	0.83%	2.01%
Logistic Regression	50.63%	3.83%	14.52%	3.64%	0.83%	1.87%
Perceptron	17.43%	15.88%	17.03%	8.33%	7.80%	6.37%
RidgeClassifier	51.89%	0.87%	4.02%	1.23%	0.14%	0.43%
BernoulliNB	10.54%	47.77%	12.47%	5.21%	22.33%	6.06%
KNN	36.87%	9.91%	23.74%	5.78%	2.61%	4.22%
LinearSVC	49.01%	6.33%	20.70%	8.57%	3.27%	5.64%
DecisionTree	13.35%	14.83%	13.61%	5.12%	5.50%	4.98%
ExtraTree	12.75%	14.62%	13.08%	4.92%	5.95%	4.87%
Adapted algorithms						
ExtraTrees	37.19%	3.32%	12.15%	4.59%	0.90%	2.25%
RandomForest	37.65%	2.85%	10.89%	3.03%	0.59%	1.40%
KNN	36.67%	9.89%	23.65%	5.49%	2.58%	4.11%
DecisionTree	12.31%	14.03%	12.60%	4.86%	5.54%	4.73%
ExtraTree	12.53%	14.41%	12.85%	5.17%	5.75%	5.05%
w2v sum						
Classifier Chain						
AdaBoost	27.88%	7.34%	17.68%	8.77%	3.47%	6.14%
Bagging	23.89%	2.50%	8.73%	3.50%	0.68%	1.67%
ExtraTrees	20.06%	1.02%	4.15%	2.19%	0.32%	0.90%
Gradient Boosting	19.01%	1.94%	6.92%	1.86%	1.01%	1.61%
RandomForest	36.44%	2.19%	8.74%	3.06%	0.41%	1.21%
Logistic Regression	29.47%	4.85%	14.57%	2.86%	2.15%	2.69%
RidgeClassifier	23.28%	3.18%	9.94%	5.43%	1.27%	2.86%
BernoulliNB	3.51%	**65.27%**	4.32%	3.38%	**40.81%**	4.05%
KNN	38.71%	9.12%	23.33%	5.17%	1.88%	3.36%
LinearSVC	13.56%	8.16%	11.94%	4.95%	3.76%	4.71%
DecisionTree	10.54%	14.60%	11.13%	4.37%	5.60%	4.32%
ExtraTree	11.05%	14.19%	11.54%	4.34%	5.31%	4.30%
Binary Relevance						
AdaBoost	28.34%	11.69%	21.93%	9.63%	4.50%	7.30%
Bagging	28.74%	5.44%	15.27%	4.85%	1.34%	2.76%
ExtraTrees	32.49%	4.11%	13.46%	4.49%	0.98%	2.29%
Gradient Boosting	21.14%	8.41%	16.14%	7.38%	2.96%	5.00%
RandomForest	33.04%	4.16%	13.69%	3.76%	0.92%	2.07%
Logistic Regression	20.49%	18.60%	19.98%	11.48%	10.53%	10.62%
Perceptron	16.39%	18.52%	16.77%	6.27%	7.96%	5.65%
RidgeClassifier	30.28%	5.70%	15.72%	6.46%	1.89%	3.91%
BernoulliNB	9.98%	49.20%	11.86%	5.10%	23.51%	5.93%
KNN	38.25%	8.98%	23.00%	5.10%	1.86%	3.35%

LinearSVC	11.64%	22.82%	12.87%	8.30%	15.49%	8.42%
DecisionTree	13.43%	15.53%	13.77%	4.88%	5.94%	4.84%
ExtraTree	11.94%	14.43%	12.34%	4.74%	5.74%	4.71%
Adapted algorithms						
ExtraTrees	33.40%	3.77%	12.85%	3.55%	0.78%	1.85%
RandomForest	37.56%	3.55%	12.68%	3.29%	0.68%	1.64%
KNN	38.58%	9.09%	23.27%	5.60%	1.90%	3.49%
DecisionTree	13.39%	14.23%	13.51%	5.63%	5.73%	5.42%
ExtraTree	11.93%	13.13%	12.12%	5.02%	5.27%	4.79%

C. Imbalanced dataset analysis

1) Random resampling evaluation for tactics

	Micro			Macro		
	Precision	Recall	F0.5	Precision	Recall	F0.5
TF						
Binary Relevance						
AdaBoost	49.77%	60.45%	47.08%	52.24%	65.22%	48.91%
Bagging	52.39%	58.59%	49.36%	54.32%	63.73%	50.48%
ExtraTrees	52.71%	42.85%	48.02%	52.37%	51.14%	45.36%
Gradient Boosting	54.87%	**66.41%**	51.94%	57.66%	**72.47%**	53.92%
RandomForest	49.84%	50.23%	47.36%	51.23%	57.74%	46.35%
Logistic Regression	56.94%	61.56%	52.44%	58.66%	66.74%	53.90%
Perceptron	50.28%	60.76%	45.45%	52.73%	65.79%	47.61%
RidgeClassifier	49.93%	55.86%	45.48%	51.50%	58.97%	47.03%
BernoulliNB	49.82%	47.92%	44.88%	50.68%	54.48%	45.01%
KNN	43.26%	57.69%	39.63%	46.11%	63.42%	41.76%
LinearSVC	54.95%	57.69%	50.76%	56.45%	63.37%	51.83%
DecisionTree	44.84%	57.91%	41.24%	47.35%	60.99%	43.56%
ExtraTree	38.84%	46.45%	34.17%	40.97%	52.44%	35.88%
TFIDF						
Binary Relevance						
AdaBoost	49.52%	58.46%	46.12%	51.84%	63.79%	47.86%
Bagging	53.15%	56.68%	50.06%	54.70%	61.94%	50.87%
ExtraTrees	51.47%	40.53%	47.17%	51.10%	49.72%	42.96%
Gradient Boosting	54.30%	63.40%	50.79%	56.92%	70.53%	52.48%
RandomForest	48.77%	47.65%	45.17%	49.96%	55.36%	44.17%
Logistic Regression	59.25%	63.13%	**56.69%**	61.09%	69.80%	**57.14%**
Perceptron	57.26%	62.68%	53.97%	59.28%	69.11%	54.98%
RidgeClassifier	**59.47%**	62.28%	55.77%	**61.13%**	68.89%	56.59%
BernoulliNB	49.35%	53.00%	44.74%	51.13%	59.78%	45.70%
KNN	43.50%	49.52%	45.71%	45.73%	57.55%	43.62%
LinearSVC	57.71%	66.17%	53.88%	59.97%	71.18%	55.75%
DecisionTree	43.59%	55.21%	39.88%	45.98%	58.92%	42.01%
ExtraTree	38.74%	46.25%	33.89%	40.88%	52.46%	35.64%

2) Random resampling evaluation for techniques

	Micro			Macro		
	Precision	Recall	F0.5	Precision	Recall	F0.5
TF						
Binary Relevance						
AdaBoost	26.44%	23.02%	22.79%	14.65%	16.16%	12.45%
Bagging	26.99%	12.79%	16.81%	12.34%	15.43%	10.75%
ExtraTrees	21.58%	5.21%	8.50%	5.61%	3.63%	3.71%
Gradient Boosting	18.47%	20.32%	16.73%	14.80%	24.51%	13.96%
RandomForest	21.12%	4.66%	7.09%	3.21%	1.61%	1.77%
Logistic Regression	25.21%	8.42%	14.75%	8.31%	5.55%	5.74%
Perceptron	14.58%	24.30%	14.79%	10.01%	16.24%	9.44%
RidgeClassifier	11.40%	31.83%	12.83%	9.05%	**32.50%**	10.04%
BernoulliNB	9.78%	**37.86%**	11.95%	3.62%	15.60%	4.40%
KNN	25.22%	9.90%	12.78%	5.06%	3.88%	3.52%
LinearSVC	17.85%	24.59%	17.97%	12.85%	17.01%	11.95%
DecisionTree	17.46%	24.90%	17.11%	14.70%	21.66%	13.61%
ExtraTree	10.26%	17.46%	10.30%	5.59%	11.25%	5.49%
TFIDF						
Binary Relevance						
AdaBoost	27.53%	18.12%	22.32%	14.55%	13.00%	11.85%
Bagging	34.84%	9.06%	17.13%	12.70%	11.51%	10.01%
ExtraTrees	24.43%	3.18%	6.98%	5.23%	2.46%	3.15%
Gradient Boosting	19.45%	16.76%	17.10%	15.33%	21.69%	14.38%
RandomForest	23.98%	2.69%	5.43%	3.44%	1.40%	1.82%
Logistic Regression	42.05%	4.98%	12.41%	7.53%	4.79%	5.64%
Perceptron	25.16%	22.56%	23.03%	19.50%	21.29%	17.32%
RidgeClassifier	**52.10%**	5.18%	12.91%	9.65%	7.40%	7.58%
BernoulliNB	11.05%	27.40%	12.53%	4.04%	11.71%	4.56%
KNN	23.94%	2.96%	7.45%	4.51%	2.00%	2.66%
LinearSVC	31.16%	32.86%	**29.37%**	**26.27%**	28.05%	**22.87%**
DecisionTree	17.44%	22.91%	17.02%	15.45%	20.21%	14.21%
ExtraTree	11.24%	12.91%	10.43%	6.87%	9.12%	6.12%

3) Study of the impact of different sampling on the classification by techniques

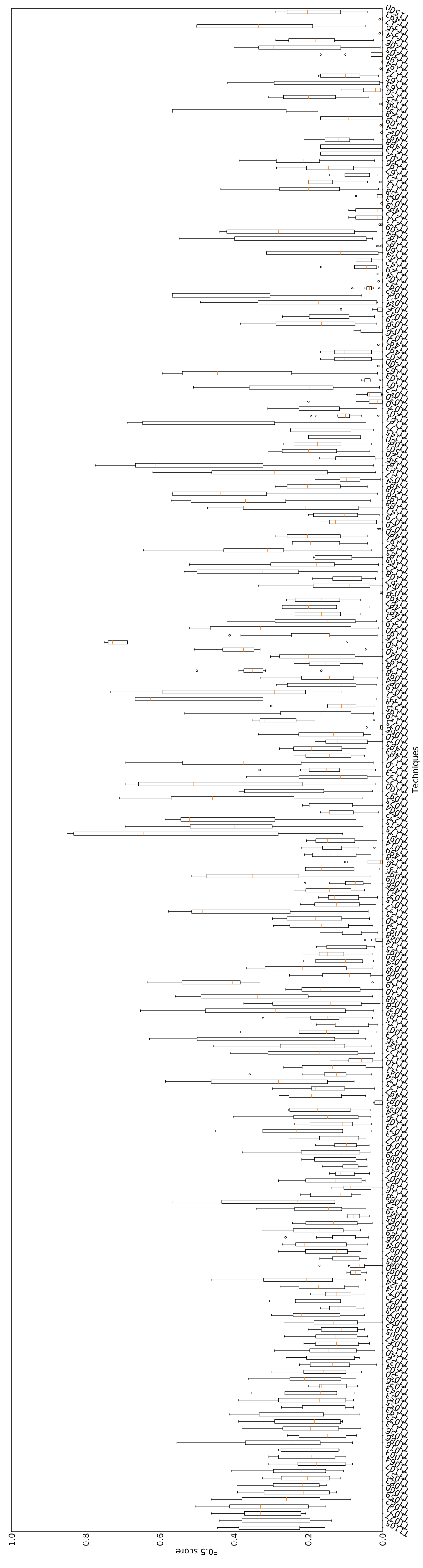


Fig. 1: Impact of training sampling size on F0.5 score of each technique, with techniques sorted by decreasing number of CTRs in the original dataset.

D. Frequency of confidence scores for false and true predictions

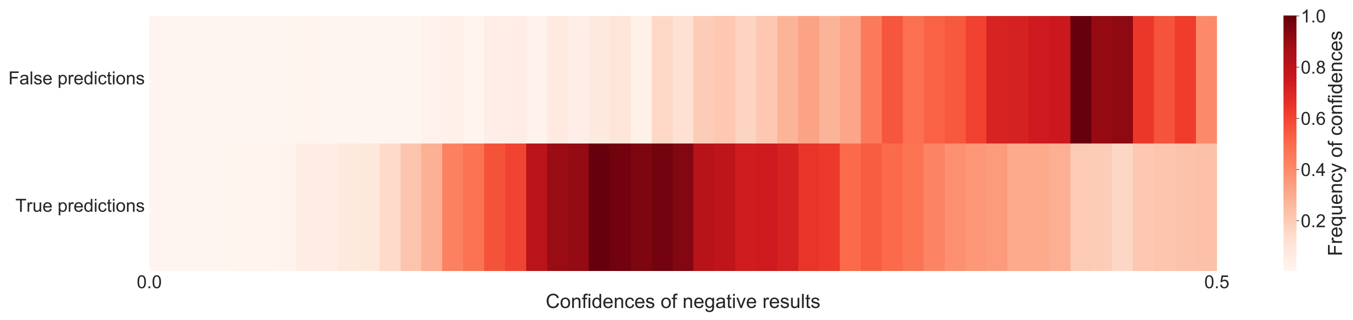


Fig. 1: Frequency of confidence scores for false and true negative tactics prediction, using the TF-IDF weighted, binary relevance Linear SVC model.

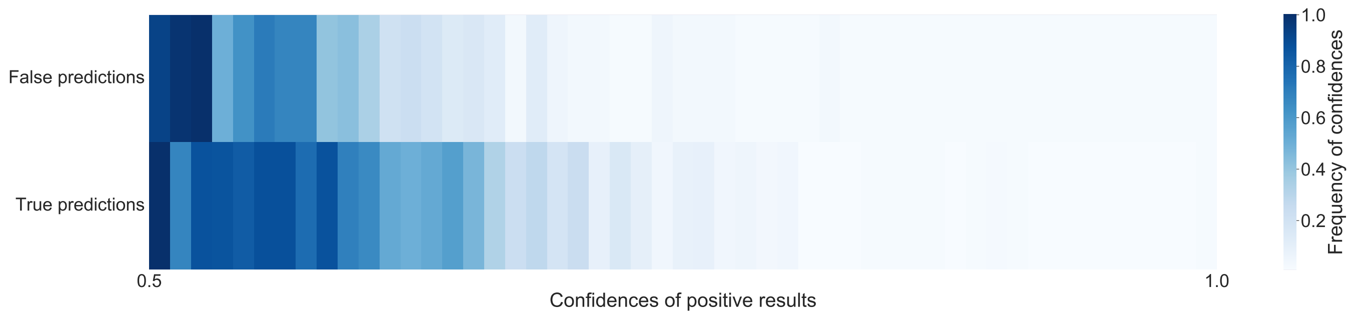


Fig. 2: Frequency of confidence scores for false and true positive tactics prediction, using the TF-IDF weighted, binary relevance Linear SVC model.

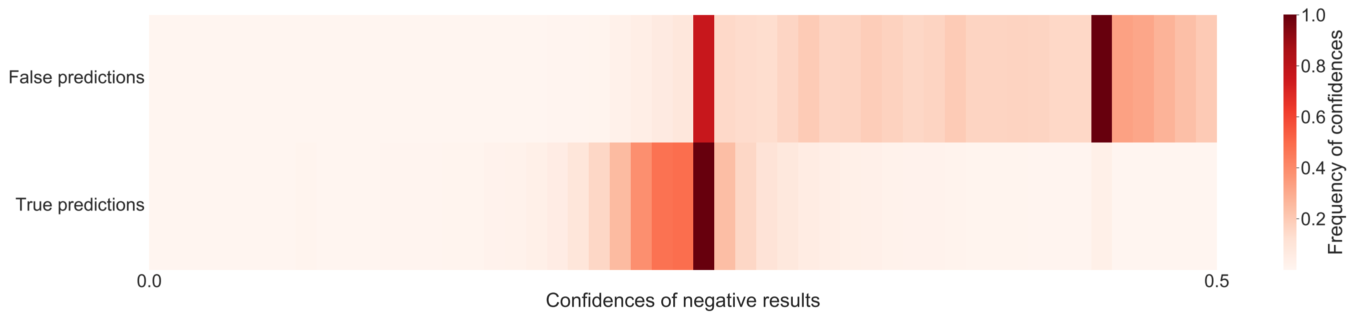


Fig. 3: Frequency of confidence scores for false and true negative techniques prediction, using the TF-IDF weighted, binary relevance Linear SVC model.

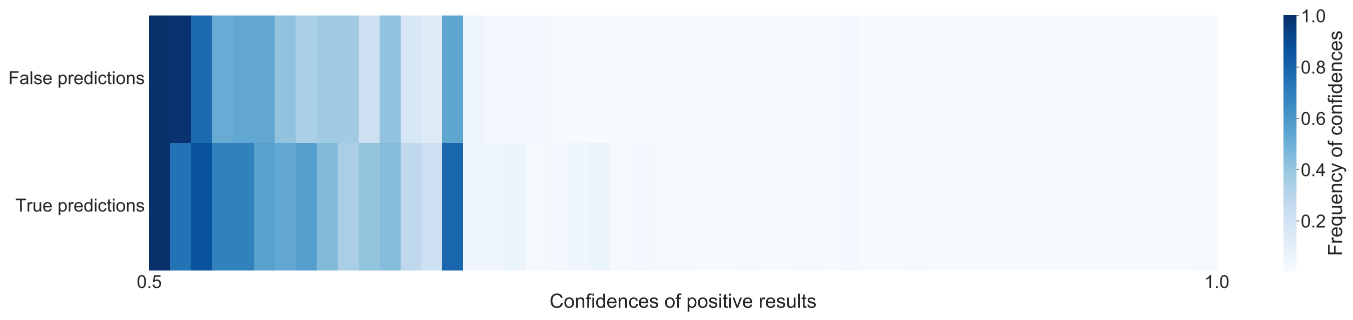


Fig. 4: Frequency of confidence scores for false and true positive techniques prediction, using the TF-IDF weighted, binary relevance Linear SVC model.

E. Impact of post-processing method on dataset

1) Samples showing the behaviours of the hanging node and confidence propagation approach on the dataset

	Independent				Hanging node			
Tactics	TP	FN	TN	FP	TP	FN	TN	FP
TA0005	147	46	173	38	154	39	153	58
TA0003	107	42	211	44	114	35	190	65
TA0002	74	41	252	37	79	36	242	47
TA0004	64	53	263	24	70	47	241	46
TA0008	37	52	291	24	44	45	278	37
TA0011	62	26	295	21	67	21	282	34
TA0007	50	31	295	28	53	28	284	39
TA0006	30	45	306	23	33	42	277	52
TA0009	20	27	331	26	24	23	316	41
TA0001	14	35	344	11	21	28	337	18
TA0010	5	25	362	12	8	22	351	23
TA0040	10	16	432	8	14	12	424	16

Fig. 1: Sample over one fold of the change in prediction between the independent classification of tactics and the one using the hanging node post-processing.

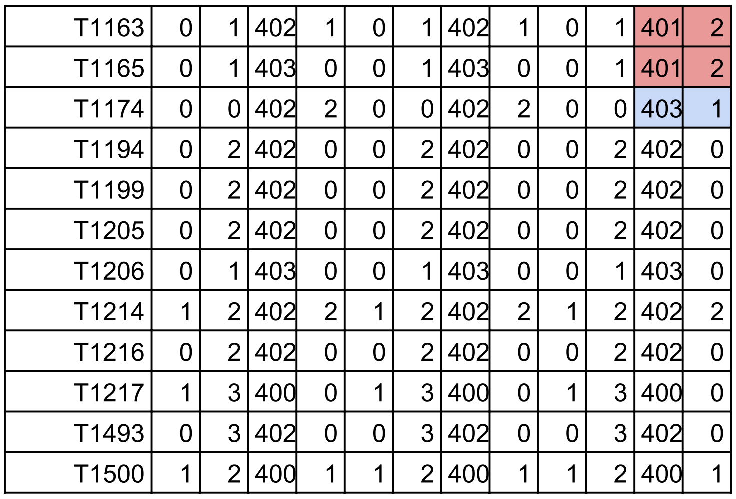
	Independent				Hanging node				Conf. propag.			
Techniques	TP	FN	TN	FP	TP	FN	TN	FP	TP	FN	TN	FP
T1105	23	23	332	26	23	23	335	23	28	18	331	27
T1027	21	9	336	38	21	9	337	37	24	6	323	51
T1071	20	20	342	22	20	20	343	21	26	14	335	29
T1082	25	13	331	35	25	13	331	35	27	11	333	33
T1059	21	14	337	32	21	14	337	32	21	14	335	34
T1060	15	19	336	34	15	19	339	31	16	18	327	43
T1083	12	14	349	29	11	15	350	28	15	11	341	37
T1057	12	16	345	31	11	17	345	31	15	13	336	40
T1107	12	12	341	39	11	13	343	37	16	8	327	53
T1064	5	20	360	19	5	20	360	19	12	13	349	30
T1003	8	24	347	25	7	25	347	25	8	24	352	20
T1016	14	13	351	26	13	14	352	25	16	11	348	29
T1086	9	20	360	15	9	20	360	15	13	16	355	20
T1056	7	14	357	26	7	14	357	26	6	15	364	19
T1113	5	16	362	21	5	16	362	21	7	14	358	25
T1033	12	13	354	25	12	13	354	25	14	11	351	28
T1193	10	9	373	12	10	9	373	12	9	10	374	11
T1055	5	25	357	17	5	25	357	17	7	23	345	29
T1053	4	26	359	15	4	26	359	15	11	19	349	25
T1043	5	7	379	13	5	7	379	13	7	5	366	26
T1036	3	11	371	19	3	11	371	19	4	10	358	32
T1050	5	18	372	9	5	18	372	9	7	16	364	17
T1204	4	7	378	15	4	7	378	15	4	7	370	23
T1032	5	14	369	16	5	14	369	16	8	11	365	20
T1140	4	11	379	10	4	11	379	10	8	7	370	19
T1112	6	12	374	12	6	12	374	12	10	8	366	20
T1047	2	11	379	12	2	11	379	12	4	9	377	14
T1005	3	13	370	18	3	13	370	18	3	13	369	19
T1012	4	12	376	12	4	12	376	12	7	9	368	20
T1063	2	11	383	8	2	11	383	8	7	6	380	11
T1078	2	7	391	4	2	7	391	4	2	7	390	5
T1002	2	11	378	13	2	11	378	13	2	11	380	11
T1022	1	11	381	11	1	11	381	11	1	11	383	9
T1074	6	6	377	15	6	6	377	15	9	3	376	16
T1024	4	14	381	5	4	14	381	5	7	11	379	7
T1203	3	4	389	8	3	4	389	8	3	4	385	12
T1090	1	6	385	12	1	6	385	12	2	5	382	15
T1085	0	8	391	5	0	8	391	5	4	4	381	15
T1087	2	13	382	7	2	13	382	7	4	11	375	14

T1094	1	5	389	9	1	5	389	9	2	4	380	18
T1076	3	5	388	8	3	5	388	8	3	5	386	10
T1089	3	7	385	9	3	7	385	9	3	7	377	17
T1102	2	5	388	9	2	5	388	9	3	4	374	23
T1065	0	6	385	13	0	6	385	13	0	6	385	13
T1049	2	8	387	7	2	8	387	7	1	9	378	16
T1132	2	7	395	0	2	7	395	0	5	4	389	6
T1088	1	11	392	0	1	11	392	0	4	8	388	4
T1116	3	5	386	10	3	5	386	10	3	5	381	15
T1192	0	8	395	1	0	8	395	1	0	8	395	1
T1007	2	6	391	5	2	6	391	5	3	5	388	8
T1045	3	4	395	2	3	4	395	2	4	3	391	6
T1018	1	6	392	5	1	6	392	5	2	5	389	8
T1099	1	7	390	6	1	7	390	6	2	6	389	7
T1070	2	6	389	7	2	6	389	7	2	6	383	13
T1077	2	8	391	3	2	8	391	3	2	8	392	2
T1023	1	8	389	6	1	8	389	6	2	7	388	7
T1073	2	4	391	7	2	4	391	7	4	2	385	13
T1106	1	7	396	0	1	7	396	0	2	6	394	2
T1124	1	8	394	1	1	8	394	1	2	7	393	2
T1035	1	8	393	2	1	8	393	2	4	5	388	7
T1081	1	4	389	10	1	4	389	10	1	4	389	10
T1497	2	7	393	2	2	7	393	2	3	6	388	7
T1125	2	7	390	5	2	7	390	5	2	7	391	4
T1134	1	9	390	4	1	9	390	4	4	6	391	3
T1041	0	3	397	4	0	3	397	4	0	3	397	4
T1110	2	0	398	4	2	0	398	4	2	0	398	4
T1117	0	4	399	1	0	4	399	1	2	2	394	6
T1123	2	9	393	0	2	9	393	0	2	9	393	0
T1046	3	7	393	1	3	7	393	1	2	8	393	1
T1173	3	1	397	3	3	1	397	3	3	1	394	6
T1001	0	6	397	1	0	6	397	1	1	5	393	5
T1115	1	7	396	0	1	7	396	0	1	7	396	0
T1189	1	5	395	3	1	5	395	3	1	5	396	2
T1038	3	7	392	2	3	7	392	2	4	6	389	5
T1068	2	8	392	2	2	8	392	2	3	7	391	3
T1210	2	6	396	0	2	6	396	0	2	6	396	0
T1119	2	6	393	3	2	6	393	3	2	6	392	4
T1179	2	1	401	0	2	1	401	0	2	1	399	2
T1008	0	5	388	11	0	5	388	11	0	5	385	14
T1034	0	8	390	6	0	8	390	6	2	6	391	5

T1069	1	6	394	3	1	6	394	3	2	5	393	4
T1095	0	3	392	9	0	3	392	9	0	3	392	9
T1114	1	8	395	0	1	8	395	0	1	8	395	0
T1048	0	6	392	6	0	6	392	6	0	6	393	5
T1093	1	7	395	1	1	7	395	1	2	6	388	8
T1120	1	6	394	3	1	6	394	3	3	4	392	5
T1135	2	8	394	0	2	8	394	0	2	8	393	1
T1137	3	1	399	1	3	1	399	1	3	1	399	1
T1015	1	5	396	2	1	5	396	2	1	5	395	3
T1031	0	6	388	10	0	6	388	10	0	6	388	10
T1486	1	5	388	10	1	5	388	10	1	5	388	10
T1009	2	2	394	6	2	2	394	6	2	2	385	15
T1096	6	2	396	0	6	2	396	0	6	2	395	1
T1136	1	3	398	2	1	3	398	2	2	2	398	2
T1158	0	2	399	3	0	2	399	3	1	1	399	3
T1499	1	3	396	4	1	3	396	4	2	2	396	4
T1014	1	2	394	7	1	2	394	7	1	2	396	5
T1091	1	7	396	0	1	7	396	0	1	7	395	1
T1175	2	3	396	3	2	3	396	3	3	2	396	3
T1215	1	5	395	3	1	5	395	3	1	5	394	4
T1222	3	2	399	0	3	2	399	0	4	1	399	0
T1004	0	4	400	0	0	4	400	0	1	3	400	0
T1025	1	6	396	1	1	6	396	1	1	6	395	2
T1097	2	2	400	0	2	2	400	0	2	2	400	0
T1100	1	2	399	2	1	2	399	2	1	2	399	2
T1127	2	2	400	0	2	2	400	0	2	2	399	1
T1133	1	0	402	1	1	0	402	1	1	0	402	1
T1170	0	5	396	3	0	5	396	3	0	5	395	4
T1171	0	5	398	1	0	5	398	1	0	5	398	1
T1483	1	4	396	3	1	4	396	3	1	4	395	4
T1485	0	5	397	2	0	5	397	2	0	5	397	2
T1010	2	5	386	11	2	5	386	11	3	4	386	11
T1066	1	4	398	1	1	4	398	1	1	4	396	3
T1075	1	3	400	0	1	3	400	0	1	3	400	0
T1159	1	2	397	4	1	2	397	4	1	2	398	3
T1195	2	2	399	1	2	2	399	1	2	2	399	1
T1218	1	1	400	2	1	1	400	2	1	1	399	3
T1221	3	3	398	0	3	3	398	0	4	2	398	0
T1019	1	0	403	0	1	0	403	0	1	0	403	0
T1084	1	1	400	2	1	1	400	2	1	1	399	3
T1098	1	2	392	9	1	2	392	9	1	2	394	7

T1178	3	5	393	3	3	5	393	3	3	5	395	1
T1197	3	1	391	9	3	1	391	9	4	0	391	9
T1040	2	2	399	1	2	2	399	1	1	3	400	0
T1130	1	1	401	1	1	1	401	1	1	1	400	2
T1176	3	0	401	0	3	0	401	0	3	0	401	0
T1190	1	2	400	1	1	2	400	1	1	2	400	1
T1219	1	1	401	1	1	1	401	1	1	1	401	1
T1223	0	2	402	0	0	2	402	0	0	2	402	0
T1482	2	1	400	1	2	1	400	1	2	1	400	1
T1487	1	1	400	2	1	1	400	2	1	1	400	2
T1498	1	1	402	0	1	1	402	0	1	1	402	0
T1021	1	1	402	0	1	1	402	0	1	1	402	0
T1067	1	2	398	3	1	2	398	3	1	2	398	3
T1108	1	4	397	2	1	4	397	2	1	4	397	2
T1122	3	4	397	0	3	4	397	0	4	3	397	0
T1168	1	2	401	0	1	2	401	0	1	2	401	0
T1185	0	3	400	1	0	3	400	1	0	3	400	1
T1187	1	5	397	1	1	5	397	1	2	4	397	1
T1191	0	1	398	5	0	1	398	5	0	1	395	8
T1480	4	3	397	0	4	3	397	0	5	2	397	0
T1029	0	0	397	7	0	0	397	7	0	0	398	6
T1079	0	4	399	1	0	4	399	1	0	4	400	0
T1141	0	3	398	3	0	3	398	3	0	3	400	1
T1188	0	3	400	1	0	3	400	1	0	3	401	0
T1198	2	1	400	1	2	1	400	1	2	1	400	1
T1208	0	3	400	1	0	3	400	1	1	2	399	2
T1484	0	3	396	5	0	3	396	5	0	3	398	3
T1037	0	5	399	0	0	5	399	0	0	5	399	0
T1183	1	2	400	1	1	2	400	1	1	2	398	3
T1186	2	5	396	1	2	5	396	1	4	3	397	0
T1220	0	4	399	1	0	4	399	1	1	3	394	6
T1501	1	2	401	0	1	2	401	0	1	2	399	2
T1080	1	5	398	0	1	5	398	0	1	5	398	0
T1145	0	1	402	1	0	1	402	1	0	1	402	1
T1177	1	2	393	8	1	2	393	8	1	2	394	7
T1181	2	2	399	1	2	2	399	1	2	2	397	3
T1201	0	4	400	0	0	4	400	0	0	4	400	0
T1020	2	3	399	0	2	3	399	0	2	3	399	0
T1030	1	0	403	0	1	0	403	0	1	0	403	0
T1052	0	1	402	1	0	1	402	1	0	1	402	1
T1101	0	3	401	0	0	3	401	0	0	3	401	0

T1103	0	4	394	6	0	4	394	6	1	3	394	6
T1162	3	0	397	4	3	0	397	4	3	0	398	3
T1200	0	1	403	0	0	1	403	0	0	1	403	0
T1207	0	1	402	1	0	1	402	1	0	1	401	2
T1490	0	1	400	3	0	1	400	3	0	1	400	3
T1491	0	1	403	0	0	1	403	0	0	1	402	1
T1026	0	0	399	5	0	0	399	5	0	0	398	6
T1028	0	0	398	6	0	0	398	6	0	0	399	5
T1039	1	5	398	0	1	5	398	0	1	5	398	0
T1042	1	4	393	6	1	4	393	6	2	3	393	6
T1044	2	4	398	0	2	4	398	0	2	4	398	0
T1051	0	0	404	0	0	0	404	0	0	0	404	0
T1062	1	0	403	0	1	0	403	0	1	0	403	0
T1092	0	2	402	0	0	2	402	0	0	2	402	0
T1104	0	2	402	0	0	2	402	0	0	2	402	0
T1129	0	2	401	1	0	2	401	1	0	2	401	1
T1142	0	4	400	0	0	4	400	0	0	4	400	0
T1144	0	3	400	1	0	3	400	1	0	3	399	2
T1160	0	1	402	1	0	1	402	1	0	1	401	2
T1182	0	3	401	0	0	3	401	0	0	3	401	0
T1184	0	4	400	0	0	4	400	0	0	4	400	0
T1209	0	3	398	3	0	3	398	3	0	3	397	4
T1211	0	3	399	2	0	3	399	2	0	3	400	1
T1212	0	2	398	4	0	2	398	4	0	2	400	2
T1489	1	1	399	3	1	1	399	3	1	1	400	2
T1013	0	1	401	2	0	1	401	2	1	0	400	3
T1058	0	2	396	6	0	2	396	6	0	2	397	5
T1111	1	4	398	1	1	4	398	1	1	4	399	0
T1131	1	1	400	2	1	1	400	2	1	1	400	2
T1167	0	3	397	4	0	3	397	4	0	3	400	1
T1196	0	2	400	2	0	2	400	2	0	2	400	2
T1202	2	3	396	3	2	3	396	3	2	3	395	4
T1213	0	0	403	1	0	0	403	1	0	0	404	0
T1488	1	2	400	3	1	2	400	3	1	2	400	3
T1492	3	2	395	3	3	2	395	3	3	2	394	4
T1054	1	0	402	1	1	0	402	1	0	1	403	0
T1109	0	2	402	0	0	2	402	0	0	2	402	0
T1128	0	3	401	0	0	3	401	0	0	3	400	1
T1138	0	0	404	0	0	0	404	0	0	0	403	1
T1155	0	1	403	0	0	1	403	0	0	1	403	0
T1156	0	3	401	1	0	3	401	1	0	3	400	2



T1163	0	1	402	1	0	1	402	1	0	1	401	2
T1165	0	1	403	0	0	1	403	0	0	1	401	2
T1174	0	0	402	2	0	0	402	2	0	0	403	1
T1194	0	2	402	0	0	2	402	0	0	2	402	0
T1199	0	2	402	0	0	2	402	0	0	2	402	0
T1205	0	2	402	0	0	2	402	0	0	2	402	0
T1206	0	1	403	0	0	1	403	0	0	1	403	0
T1214	1	2	402	2	1	2	402	2	1	2	402	2
T1216	0	2	402	0	0	2	402	0	0	2	402	0
T1217	1	3	400	0	1	3	400	0	1	3	400	0
T1493	0	3	402	0	0	3	402	0	0	3	402	0
T1500	1	2	400	1	1	2	400	1	1	2	400	1

Fig. 2: Sample over one fold of the change in prediction between the independent classification of techniques and the one using the hanging node post-processing.

2) Analysis of the correlation between techniques predictions and number of reports by techniques

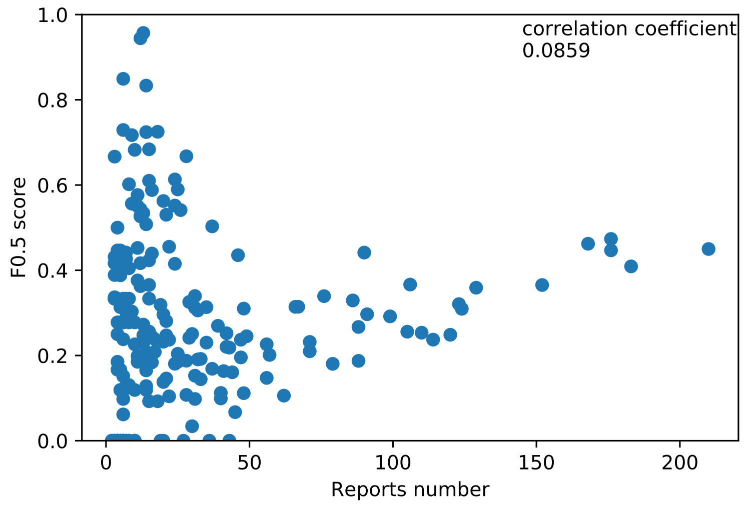


Fig. 3: Correlation between the $F_{0.5}$ score and the number of reports by techniques with the hanging node approach applied.

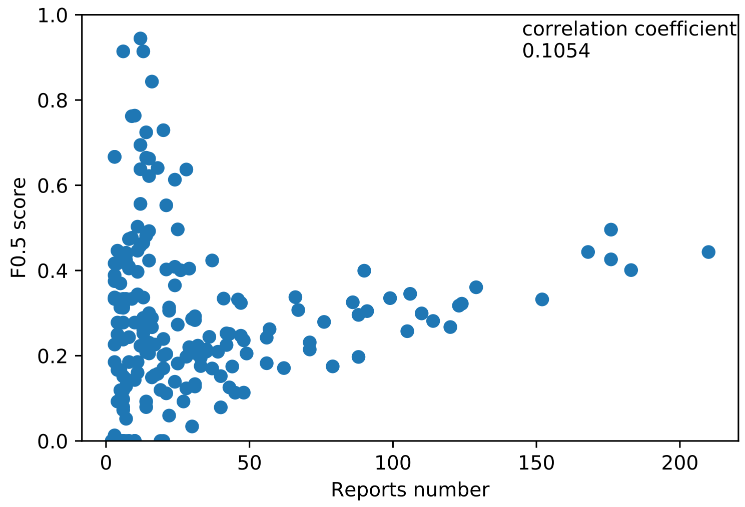


Fig. 4: Correlation between the $F_{0.5}$ score and the number of reports by techniques with the confidence propagation approach applied.