# Design of a Novel Information System for Semi-Automated Management of Cybersecurity in Industrial Control Systems

KIMIA AMERI, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, USA
MICHAEL HEMPEL, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, USA
HAMID SHARIF*, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, USA
JUAN LOPEZ JR., Oak Ridge National Laboratory, USA
KALYAN PERUMALLA†, Oak Ridge National Laboratory, USA

There is an urgent need in many critical infrastructure sectors, including the energy sector, for attaining detailed insights into cybersecurity features and compliance with cybersecurity requirements related to their Operational Technology (OT) deployments. Frequent feature changes of OT devices interfere with this need, posing a great risk to customers. One effective way to address this challenge is via a semi-automated cyber-physical security assurance approach, which enables verification and validation of the OT device cybersecurity claims against actual capabilities, both pre- and post-deployment. To realize this approach, this paper presents new methodology and algorithms to automatically identify cybersecurity-related claims expressed in natural language form in ICS device documents. We developed an identification process that employs natural language processing (NLP) techniques with the goal of semi-automated vetting of detected claims against their device implementation. We also present our novel NLP components for verifying feature claims against relevant cybersecurity requirements. The verification pipeline includes components such as automated vendor identification, device document curation, feature claim identification utilizing sentiment analysis for conflict resolution, and reporting of features that are claimed to be supported or indicated as unsupported. Our novel matching engine represents the first automated information system available in the cybersecurity domain that directly aids the generation of ICS compliance reports.

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Lexical semantics**; **Machine learning algorithms**; • **Information systems** → *Information retrieval.*

Additional Key Words and Phrases: Cybersecurity, Vetting system, Industrial Control Systems, Natural Language Processing, CYVET

**ACM Reference Format:**
Kimia Ameri, Michael Hempel, Hamid Sharif, Juan Lopez Jr., and Kalyan Perumalla. 2022. Design of a Novel Information System for Semi-Automated Management of Cybersecurity in Industrial Control Systems . 1, 1 (July 2022), 36 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*Corresponding Author
†This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Authors' addresses: Kimia Ameri, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA, kameri2@unl.edu; Michael Hempel, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA, mhempel@unl.edu; Hamid Sharif, Dept. of Electrical & Computer Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA, hsharif@unl.edu; Juan Lopez Jr., Oak Ridge National Laboratory, Oak Ridge, TN, USA, lopezj@ornl.gov; Kalyan Perumalla, Oak Ridge National Laboratory, Oak Ridge, TN, USA, perumallaks@ornl.gov.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2022 Association for Computing Machinery.
Manuscript submitted to ACM

## 1 INTRODUCTION

Cybersecurity auditing plays an increasingly important role in Operational Technology (OT). Many critical infrastructure industries, including the energy sector, rely on OT and Industrial Control Systems (ICS), and therefore a robust and reliable cybersecurity solution is needed for OT deployments. However, ICS vendors always add new features to their products to incentivize reasons for system upgrades. Often, these changes are driven by vendors, while customers may not be aware of these feature changes' full impact on their cybersecurity posture and regulatory compliance. Furthermore, the difference between vendor-provided cybersecurity feature claims and the customer's expectation for OT cybersecurity can be significant in industries such as the energy sector [62]. Due to these changes, the workload for dynamic verification shifts from the energy sector to the customer, and consequently is creating a significant cybersecurity risk. As a final consideration, the number of research studies into cybersecurity audits for OT is very limited. Therefore, there is an urgent need for a solution to audit whether vendor-supplied features (VSF) adhere to cybersecurity requirements as well as standards.

Cybersecurity requirements (CR) are standardized and codified by industry organizations and standards bodies in human-readable formats. A vendor-supplied feature (VSF) can either (1) align and match with requirements (CRs) or satisfy those requirements, (2) go beyond the related requirements, or (3) contradict and violate related requirements [62]. In order for operators to determine whether OT devices pose a threat to cybersecurity by weakening security postures, they must interpret vendor claims regarding supported features, evaluate the features of interest manually to see how well they match vendor features and reconcile those features with industry needs and requirements. Many industry requirements and their complexity must be taken into account as well as the wide variety of devices and their documentation, as well as the associated assessment of Installation Qualification (IQ), Operational Qualification (OQ), and Performance Qualification (PQ). Traditionally, this work requires specialized expertise and poses a great risk to cybersecurity assurance. Therefore, the result is subjective to human error and must be repeated periodically [62].

Achieving confidence that the cybersecurity posture of critical infrastructure industries such as the energy sector meets or exceeds the requirements of that industry in order to stay ahead of cybersecurity risks is a vital, yet extremely challenging objective. In order to aid this process we are developing a Cyber-Physical Security Assurance Framework for a Semi-Automated Vetting system (CYVET) to address this challenge. CYVET addresses directly the need to improve the current industry capabilities for operational technology cybersecurity and associated control system infrastructure validation and verification [62]. Figure. 1 shows the overall information flow of the CYVET system.

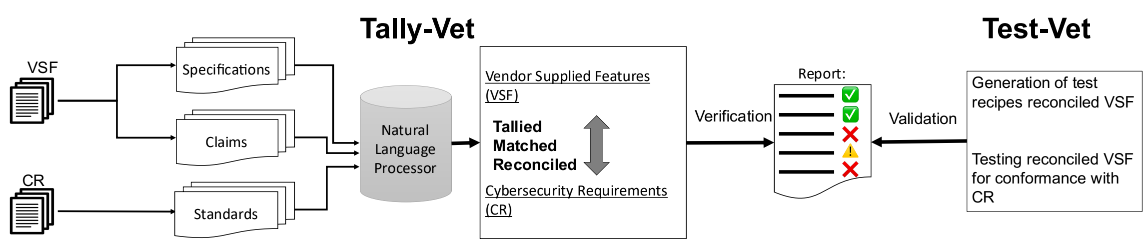


Fig. 1. Overall workflow of the CYVET cybersecurity vetting approach.

The end-to-end framework of the vetting engine, which is the core component of CYVET, is presented in this paper. The proposed framework starts with assembling and curating a large collection of ICS device documents, and performing Natural Language Processing (NLP) sentiment analysis in claims sequences to determine features support indicators for each product. This vetting system provides a vital capability to critical infrastructure operators to ensure that vendor-claimed device cybersecurity capabilities match industry requirements.

CYVET has broad applicability across all critical infrastructure sectors and beyond, since it is device- and architecture-independent. OT equipment, software, and the underlying control system architecture will be tested and validated using this cybersecurity verification and validation framework. In this paper we provide details on the framework's operations, present and discuss obtained results, and provide an outlook for our future efforts on CYVET.

CYVET and its Tally-Vet engine are a unique new approach. To the best of our knowledge there is no similar system available or published. We therefore lack the ability to compare our system with other published efforts. Instead, this paper aims to demonstrate the capabilities and results obtainable with our system using comprehensive results shown and detailed throughout this paper.

The remainder of this paper is thus organized as follows. In section 2, we briefly introduce background information related to CYVET vetting system and its components. In section 3, we briefly reviews the related work for different components of our framework.

We present our proposed framework, describe the operation of each functional element, and present results for these elements in sections 4 and 5, with each subsection also discussing relevant related results. In section 5, we also present and discuss our final Tally-Vet engine for feature claims detection. In section 6, we present a discussion and analysis our results and compare our methods with others and, in section 7, we present the conclusion and future work for this project.

## 2 BACKGROUND

### 2.1 A High-Level View of CYVET

An effective vetting engine is required to ensure device cybersecurity features meet industry-standard requirements. A key aspect of this vetting engine is to enhance the industry capabilities to verify and validate OT infrastructure cybersecurity claims, both pre- and post-deployment. The primary objectives for this vetting engine are listed below:

- **Verification:** Analysis and reconciliation of vendor features and standards.
- **Validation:** Process of generating, executing, and presenting testing scripts of the identified security features.

In pursuit of these goals, our team is researching a semi-automated cyber-physical security assurance system we call CYVET [62]. The CYVET vetting system has two primary components:

(1) **Tally-Vet:** aims to verify VSF claims against the relevant CRs. The verification process utilizes a variety of different NLP techniques to analyze both vendor cybersecurity device claims and CRs.
(2) **Test-Vet:** aims to validate the specific set of features identified by Tally-Vet, utilizing an automated approach involving actual hardware and software targets.

By integrating these two components, it is possible for CYVET to develop a sequence of cybersecurity tests to comprehensively assess and vet the target system, and to ensure that it meets the customer's requirements.

### 2.2 Tally-Vet Overview

Tally-Vet, as shown earlier, is one of the principal components of the CYVET vetting system, and is responsible for comparing and reconciling the VSFs against the corresponding CRs in order to detect matches, extended features, and possible requirement violations. To build this vetting system, a broad range of documents from ICS vendors is required.

These documents are curated and pre-processed prior to extracting textual information via NLP and subsequent parsing of human-readable text into machine-usable data.

Post-extraction from documents, all the content is stored as structured data in CYVET's database. This information is then processed by a machine learning classifier to determine and identify sequences related to the device feature claims. These claims are then reconciled and matched against the relevant CRs within Tally-Vet. The matching process enables us to identify and resolve conflicting feature claims and to collate a feature report (Figure. 1, the Tally-Vet component). ICS compliance reporting is thus simplified using Tally-Vet [62].

## 3 LITERATURE REVIEW

To the best of our knowledge, there is currently no framework available or published with the capabilities developed for our CYVET system. However, there are works published that share some similarities with individual components of our framework. In this section we therefore review these published works and compare them against our targeted functionality.

### 3.1 Document Library Curation

The number of scientific publications focused on scraping and classifying web content for domain-specific areas are limited. Anglin [3], Modi and Jagtap [59], and Luscombe *et al.*[55] were proposing NLP and machine learning techniques to scrape web content and a pipeline to classify their content and documents. The pipelines and methods these authors implemented is highly reliant on their specific domains, however. For example, Anglin [3] proposed a semi-automated framework to study local policy variation in school dress codes using web crawlers and NLP techniques. For web-scraping, the framework used a pre-determined list of schools to scrape their websites and collect all links leading to documents. The policy-relevant documents then were processed using a Convolutional Neural Network (CNN) and NLP techniques to identify policy nuances. Varela *et al.*[79] summarized different methodologies and terminology used for web scraping in the domain of political analysis. In this paper, the authors reviewed some methods and packages used for extracting political data from text documents available from the internet. Chandrika *et al.* [13] used Python to extract and parse unstructured information from the web. Then a relational database was used to store this extracted data.

All these papers are focused on scraping and crawling websites for domain-specific content, which does align with our goals. The methodologies and NLP techniques presented in this paper expand upon the approaches presented in the reviewed papers. For our framework we adopted a similar approach. We then provided it with novel capabilities to automate the process of identifying sources for domain-specific content and automated document library curation. Our framework is furthermore able to utilize multiple search engines to expand and refine its content source list of ICS vendor names and organizing downloaded documents into different categories.

### 3.2 Structured Content Extraction

There are many variations in document layout, their sections, elements or even encoding, making this a highly complex and challenging problem on how to best present essential information to the reader in a well-structured manner. The analysis of the layout of documents has been used by numerous researchers to develop techniques for detecting tables, layouts, and sections [5, 6, 18, 22, 23, 25, 27, 30, 34, 77, 89].

The two open-source Python packages, Camelot [57] and Tabula [6] are designed to find tabular content from PDF documents. However, these two packages can not successfully find all types of table formats, especially if the PDF page has a multi column format or tables span multiple pages.

For the past decade, table detection, extraction and annotation have been key research areas [17], leading to a variety of extraction approaches [11, 22, 26, 27, 50, 61, 70, 71, 86]. For example, for table detection in PDF documents, Hao *et al.* [30] and Khan *et al.* [41] and Gilani *et al.* used deep learning methods. These recent methods convert PDFs into images and then detect table boundaries and cells in the PDF page with the help of deep learning models. Hao *et al.* [30] used a set of pre-defined rules to compute region proposals. CNN is then used to identify whether these region proposals belong to tables or not. However, if the table spans across multiple columns, it is unable to recognize table regions and thus fails to properly localize the regions. Gilani *et al.* [25] presented a CNN model to detect table regions in document images. The output of this model includes the coordinates of bounding boxes of predicted tables. Khan *et al.* [41] used the CNN model introduced by Gilani *et al.* [25] to detect table boundaries. For each detected table, they then process the table image with bi-directional Gated Recurrent Unit (GRU), a type of Recurrent Neural Network, to find columns and rows. None of these papers discussed the text annotation from these detected tables.

From our review of published scientific efforts, we could not find any publications that address the specific needs and challenges of Tally-Vet's contextualized text extraction process, such as handling different list levels and tables. To address this shortcoming, our work contributes new algorithms to intelligently and automatically identify document elements such as lists and tables, as well as for contextually extracting and annotating sequences from DOCX and PDF documents.

### 3.3 Claim Detection

By adapting a pre-trained language model for a targeted downstream task, performance can be significantly improved [39, 42, 53, 75, 81]. These performance improvements can be divided into pre-training language models and fine-tuning language models process. For pre-training language models in specific domain, researchers focused on domain-specific corpus to build a specific language model such as BioBERT [48] and SciBERT [9], for biomedical language representation and scientific text, respectively. A number of studies have demonstrated significant enhancements in downstream tasks, including sentiment analysis [4] and classification [49] by fine-tuning an existing language model.

Some examples of using language models such as BERT specifically for applications in the cybersecurity domain are fine-tuning sentences based BERT sentiment analysis for vulnerability exploitability prediction (ExBERT [88]). Another example is fine-tuning BERT for Name Entity Recognition (NER) for the cybersecurity domain in English [14, 21, 90], Russian [76] and Chinese [85].

From our review, we could not find any language model tuned for cybersecurity text classification tasks. In this paper, we used our claim sequence database we developed specifically for the cybersecurity domain (details discussed in [1]. This database was used to generate CyBERT, a fine-tuned BERT classifier on cybersecurity sequences to identify feature claims. These claim sequences and the fine-tuning process we introduced in CyBERT [1] are used in this paper for our cybersecurity vetting engine to verify those features against the published industry requirements for cybersecurity.

### 3.4 Sentiment Analysis

Sentiment analysis is the field of study that analyzes opinions, emotions, sentiments, attitudes, and evaluations from written language [51]. Hutto and Gilbert [35] developed Valence Aware Dictionary for sEntiment Reasoning (VADER) algorithm, a parsimonious lexicon and rule-based model for general sentiment analysis. The prior polarity of lexical entries is usually indicated by a sentiment lexicon [84]. A sentiment lexicon refers to a list of lexical features (e.g., words) that are labeled positively or negatively based on their inherent semantic orientation [52]. A number of studies have been proposed to apply sentiment analysis for text classification [28, 43, 45, 58, 60, 64, 72, 83]. Classification methods for sentiment analysis can be divided into three types: lexicon-based approaches, machine learning approaches, and hybrid approaches [56]. Lexicon-based approaches use a set of words that have been precoded for polarity to identify sentiments [52]. To predict sentiment polarity, machine learning approaches use both supervised and unsupervised learning methods [86]. In the hybrid approach, sentiment polarity is detected through both machine learning and lexicon-based approaches [28]. For this project, we defined the feature attribution as a text classification problem based on lexicon-based approaches for sentiment analysis.

Several studies have used sentiment classification in cybersecurity textual data over the past few years [24, 28, 45, 72, 73]. Examples of these models using sentiment analysis techniques to interpret cybersecurity-related texts are shown in [28] for social media sentiment analysis, and in [24] for analyzing the sentiment expressed in newspapers. Gupta *et al.* [28] analyzed the relationship between cybersecurity attitudes of users on Twitter and online financial behaviors. They hypothesized that users with positive sentiment on cybersecurity will engage in more online financial transactions and might disclose sensitive information. The researchers used a hybrid NLP technique to integrate linguistic and statistical analysis techniques. Based on this hybrid model, the researchers were able to measure whether users are genuinely concerned about cybersecurity issues and if that concern impacts their online behavior.

Shu, Kai *et al.* [73] proposed an unsupervised sentiment predictor model to detect cyber attack behavior in Twitter users based on sentiment polarity score. In their proposed model, the sentiment analyzer was connected to a regression model to find relation and correlation between sentiments of trends in tweets, and the probability of attacks in the real world.

These studies show the benefits of using sentiment analysis in cybersecurity text classification. However, as we could observe from these related publications on sentiment analysis, their focus is on evaluating positive/negative emotional sentiment. In our work presented in this paper, we leverage the sentiment analysis paradigm to evaluate supportive/unsupportive sentiment, which only partially aligns with emotional sentiment and thus required research to be conducted that culminated in our specialized sentiment analysis approach. In other words, we use the sentiment polarity score not as an emotional indicator, but rather as an indicator to understand whether the feature is supported or not supported by the device.

## 4 PROPOSED TALLY-VET PREPROCESSING METHODOLOGY

The main focus of this paper is to present and discuss our framework for CYVET's Tally-Vet engine. Figure. 2 shows each component and their connections within the Tally-Vet framework. This framework shows that the process starts with gathering ICS device documents from online resources, and parsing of human-readable documents (PDFs and DOCXs) in order to extract information and represent them in a machine-usable format for CYVET's remaining processing workflow. A key aspect to the content extraction aspect is maintaining structural and contextual relationships between elements extracted from these document resources when they are stored in CYVET's database, for example contextually linking information from different rows and columns of tables, across images, etc. Additionally, sequences extracted from ICS documents were used to train a feature claim identifier (CyBERT) [1]. With CyBERT's classifier and NLP techniques we can identify sequences related to claims about device features. The NLP sentiment analysis technique is then performed on selecting cybersecurity device claim sequences in order to gauge whether that sequence expresses support or lack of support for that feature. By aggregating the reports for each feature across all documents related to a given device, our framework is able to identify conflicting feature claims. The final step in the Tally-Vet framework utilizes that information in order to compile a report of features that are being claimed for the device. Test-Vet, the other core component of CYVET, then can utilize that information in order to execute a sequence of tests designed to validate each claimed feature and to check for flaws within that implementation.

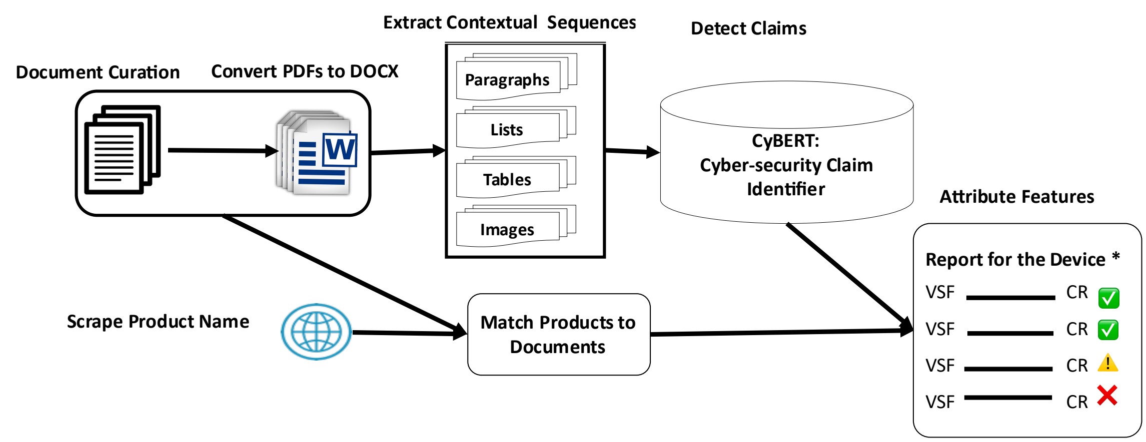


Fig. 2. Tally-Vet framework components.

As part of our research work, all steps of our framework were implemented in `Python 3.8.` in order to test and verify this semi-automated framework for the Tally-Vet engine of our CYVET system. Thus far our CYVET work focuses on English language-based vendor documents and specifications, but the same framework could be adapted to work across a range of other languages as well.

### 4.1 Document Library Curation

As shown in Figure. 1, CYVET is built around the availability of relevant documents for its vetting process. Tally-Vet includes functionality to identify vendors and their product names by scanning web sites, downloading product documentation accessible on those websites, as well as processing these documents for classification based on type, language, and device attribution. This curation of CYVET's document repository is a vital first step towards its overall functionality, and is the foundation upon which our NLP and cybersecurity vetting processes are being developed.

Identifying domain-specific information from online sources and document classification is itself a challenging task. It requires a generalized and robust approach to identify ICS vendors, analyze websites to determine ICS product offerings, and classify documents based on their content for relevant product claims. Furthermore, there was no publicly available dataset for training cybersecurity-related NLP models available for CYVET's development. Hence, CYVET's development also included the research into establishing a cybersecurity-specific NLP dataset from its document repository.

In our previous paper [2], we discussed our semi-supervised framework to establish this ICS device information repository. This framework consists of four main steps, including vendor identification, homepage classification, vendor classification, and document classification. The process starts with scanning ICS-Cert's website for ICS vendor name extraction, and using Google search to identify each vendor's homepage. Additionally, we incorporated multiple search engines to find ICS vendor-related links for vendor and homepage identification using pre-defined keywords. This process is followed by NLP techniques including Latent Dirichlet Allocation (LDA) topic modeling [10] and context-dependent scoring metrics applied to the text content of each identified web site. LDA is an unsupervised Bayesian probabilistic model capable of determining the semantic relationships among words in a corpus. The demonstrated context-dependent scoring metric algorithms are based on the appearance of defined keywords and phrases in the textual content of these web pages. These methods require some human supervision for labeling the topics of a vendor's homepage via LDA and scoring metrics. Through this process we were able to train a neural network classifier that can automatically label homepages to minimize such human interactions. This neural network model is used to identify new ICS vendor websites through an automated search process.

#### *4.1.1 Document Classification.*

Tally-Vet then downloads all PDFs from identified vendor websites, extract the textual content from these documents, and determines key phrases, key terms, and key segments. The `NLTK Toolkit` [54] Python package was used to pre-process and tokenize text content, keywords and phrases. The final step in Tally-Vet's document curation is to apply a matching algorithm to compare these key elements with a pre-defined set of ICS device information. The matching algorithm separates documents containing ICS device information from other PDF files, such as annual reports. Among all downloaded PDFs, five percent were found to be scanned documents, three percent were corrupted files, and the remaining 92 percent were regular documents for CYVET's processing pipeline. Table. 1 presents the ICS document repository statistics [2].

Table 1. ICS document repository entities statistics

| | **Number of Entities** | | **Description** |
|---|---|---|---|
| **Vendor Identification** | 1930 possible vendors | 340 vendor names | From ICS-Cert website |
| | | 1590 links | From web searches using pre-defined keywords |
| **Homepage Classification** | 1457 unique websites | 420 links | From vendor name queries |
| | | 1084 links | From keyword-driven queries |
| **Vendor Classification** | 286 ICS Vendor | 578 websites | Topic Modelling with LDA on website text content |
| | | 850 websites | Label websites with context-dependent scoring metric |
| **Document Classification** | 12581 ICS product-related | 2844 Manual | NLP techniques to determining key terms, key elements, key segments, and key phrases and match these against a pre-defined set of important phrases |
| | | 7832 Brochure | |
| | | 666 Catalog | |

In our proposed framework, we combined results from keyword searching in multiple search engines with results from scanning the ICS-Cert's website. This method ensures we cover a wide range of vendor names for our database. The other contribution of our framework in comparison with the other methods mentioned in the literature is the ability to identify ICS vendor websites with our NN model. The main advantage of our database curation framework is the ability to classify downloaded documents into four different groups of manual, brochure, catalog, and not relevant.

### 4.2 Product Name Scraper

Once our system obtained and curated a comprehensive and relevant document library of vendor documents related to products, TallyVet's analysis framework needs the ability to attribute these documents to individual product names associated with each vendor. Hence, we needed to curate a library of product names and attribute them to vendors first, before being in a position to process each document associate it to zero, one, or multiple product names. In order to curate that product name library we use a product name scraper.

Automated Web scraping uses computer code to download, extract and organize data from the web and then utilize it for further analysis [78]. This has become a valuable tool for data collection in research areas such as information technology [44], public health [67], the financial sector [46], and cybersecurity [82]. For Tally-Vet, in order to attribute a vendor's products to documents, Tally-Vet uses web scraping of ICS vendor websites in order to automate the process of identifying product names. The challenge herein is that there are wide range of vendors, each with unique website styles, formats, and methods of representing products and their names. To enable Tally-Vet to automatically detect product names we developed a customized semi-supervised approach to web scraping, which focuses on the HTML behind the web content of each ICS vendor.

This algorithm is driven by predefined rules specific to each website. The algorithm is composed of four major steps:

(1) **Link Fetching:** Compile a set of links to visit on the vendor website, which are extracted from the sitemap or a specific product section on the website.
(2) **Conditions-based HTML element identification:** For each page identified via the fetched link, find all tags that meet the requirements and conditions, utilizing nesting rules representing a hierarchy of tags and Boolean logic.
(3) **Rules-based text extraction:** For each tag identified through the condition matching above, extract text from element.
(4) **Pattern testing:** In all steps, the applied rules can leverage pattern definitions for processing extracted information.

The product name scraper algorithm starts with a supervised instruction set, which helps the script to find the pages to visit, what to look for on each page during these visits, and how to extract elements-of-interest (product names) from each page. using these instruction sets, the script will be able to identify conditions-based HTML elements. For each page, it will find all tags that meet the requirements and parse them to extract product names. An example of this rules-based approach for product name extraction is presented in Fig 3, using the SEL website for illustration.

These rules are able to, for example, represent the following process in the form of individual rules that can be applied to numerous web pages from a vendor's website:

- Analyze a paginated Product Page, including dynamically determining the maximum page index
  - Find a `<small>` tag within HTML
    - Needs to have a "translate" property with value "currentPageNofM"
    - Also needs to be nested INSIDE a `<div>` tag
      - This `<div>` tag needs to have classes "tab-pane" and "active"
  - Extracts the maximum page index from the element's property "translate-values" to get field "totalPages", and converts it to integer
  - Generates links to each paginated product page using a defined pattern, iterated over for each page number from 1 to the maximum page index

```
#---------------------------------------------------------------------------------
#   SEL inc
website=self.CreateWebsite('https://www.selinc.com/', 'SEL',2,60,30,1,5,True,sslverify=True)
#PAGNINATED PRODUCT PAGE LINK EXTRACTOR===============
c_proptranslate=self.CreateCondition_ElemProp('translate','currentPageNofM',None,False,False)
c_tabactiveclass=self.CreateCondition_ElemClass(['tab-pane','active'],True,False,False)
c_insidediv_with_tabactiveclass=self.CreateCondition_InsideTag('div',[c_tabactiveclass])
c_small_with_proptranslate=self.CreateCondition_ElemTag('small',[c_proptranslate,c_insidediv_with_tabactiveclass])
a_propextract=self.CreateAction_Extract_ElemPropValueJSON('translate-values',False,'object','totalPages',False,'int')
x_maxpages=self.CreateExtractor(c_small_with_proptranslate,a_propextract,"extract")
prodpagepattern=self.CreatePattern(['https://selinc.com/search/?product.cpg=','<PAGE>','#tab-products'])
prodpagefetcher=self.CreateProductPageLinkFetcher('https://selinc.com/search/#tab-
        products',prodpagepattern,'links',x_maxpages,self.CreateConstantExtractor(1,'int'),self.CreateConstantExtractor(1,'int’))
#PRODUCT NAME EXTRACTOR==============================
c_tabactiveclass=self.CreateCondition_ElemClass(['tab-pane','active'],True,False,False)
c_insidediv_with_tabactiveclass=self.CreateCondition_InsideTag('div',[c_tabactiveclass])
c_propngbind=self.CreateCondition_ElemProp('ng-bind-html','::$ctrl.searchResult.title',None,False,False)
c_span_with_propngbind=self.CreateCondition_ElemTag('span',[c_propngbind,c_insidediv_with_tabactiveclass])
a_prodnameextract=self.CreateAction_PatternMatchExtract_ElemText(['SEL-','<*>'],[' ',','],'any',False,None,'string')
x_prodname1=self.CreateExtractor(c_span_with_propngbind,a_prodnameextract,'extract','any','always’)
#FINALIZING WEBSITE SETUP============================
self.AddLinkFetcherToWebsite(website,prodpagefetcher)
self.AddProductExtractorToWebsite(website,x_prodname1)
self.AddWebsiteToLibrary(website)
```

Fig. 3. Product name scraper rule-base script example for one website.

- Visits each page link generated from the pagination
- On each page it locates <span> tags
  - Each span needs to have style classes "`tab-pane`" and "`active`"
  - It also needs to have an element property with a given value
- From the matched element it extracts the text and matches it against a specific list of product name patterns
- Only pattern-matching product names are accepted and added to the list of extracted product names

These rules shown above are an example taken from our rule set, and are specific to a given vendor website. Using rules such as these, Tally-Vet's product name extractor was able to identify a total of 264 product names from that vendor's website. By customizing these rules we can adapt the extraction process to any vendor website.

### 4.3 Sequence Extraction

An important aspect of NLP applies to document content is the ability to extract textual content in the form of sequences. Additionally, although understanding the structure of a document is important for many NLP tasks and overall document content analytics, there are also use cases where the structure does not play an important role, and only the sequences themselves - the raw text - is needed. The Document Classification, as shown in the earlier subsection 4.1.1, is one of the examples where only the textual information without associated context is relevant and extracted by the system. For document classification, the focus is on finding key elements and key phrases in the document. Thus, in these cases we only focus on raw text directly extracted from PDF documents, which is a faster process than conducting the full contextual text extraction pathway detailed in the next subsection (subsection 4.4), while providing all relevant information for a full-text keyword search.

As mentioned above, Tally-Vet provides two pathways for accessing content from documents:

- A direct access to the PDF for simple raw text extraction, and contextual sequence extraction.

- By first converting the PDF to DOCX in order to make the structure and context accessible, and then extracting structural sequences from the DOCX representation.

In the following section, we detail our proposed algorithm for contextual sequence extraction from PDF documents, i.e., the algorithms we use to extract sequences in such a way that it maintains the context in which they were present in the document.

### 4.4 Contextual Sequence Extraction from PDF

Due to the PDF document's challenges in maintaining the context of the presented information when shown within tables, lists, etc., the procedure that Tally-Vet employs for content extraction with context representation is complex.

To handle working directly with PDFs, Tally-Vet uses the `PyMuPDF` [38] Python package to extract metadata for each PDF document. This metadata is then used in our *StructuredSequence* algorithm to extract contextualized sequences from PDF elements, such as paragraphs and table content. In this section we describe the details of each stage of Tally-Vet's *StructuredSequence* algorithm. Figure. 4 illustrates a hierarchical structure in a PDF document. This hierarchical structure is the basis for *StructuredSequence* elements. Algorithm 2 presents a high level pseudo-code for Tally-Vet's *StructuredSequence* algorithm.

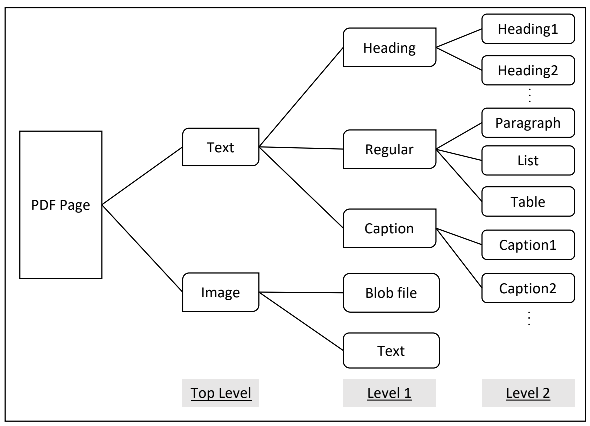


Fig. 4. Hierarchical structure in PDF documents.

For each page in the PDF, the algorithm groups and orders each text block based on page orientation, using the $PageOrientation(Page)$ function. This function will use the *BBOX* tag values extracted using `PyMuPDF` to decide if the paper is in portrait or landscape orientation. Page orientation is important factor which help us order each element of the page, and also to to find the spatial location which will be used for lists, captions, and headings identification. For image blocks, it reads the text embedded within the image using Optical Character Recognition (OCR), provided by the open-source `Python-Tesseract (PyTesseract)` library [32]. Then, for each text block the structural type is determined, considering the font size, style and spatial location. Next, for each text block content is grouped into headings, regular and captions. Each paragraph and caption group is then sequentially ordered based on the calculated bounding box of the group and other information obtained from `PyMuPDF`'s dictionary.

For regular type elements, the *DissectTextRegular* function is called to further differentiate the type of content, detecting for example tables and lists within paragraphs. The contextual sequence extraction algorithm allows us to use `PyMuPDF` to obtain the text spans and identify paragraphs, tables and list. For each text block in each document page, based on spans for each line a list of bullet point identifiers is determined using regular expressions (*ExtractLayouts* function). This list information, combined with the start position for each line, is then used to identify the list level. The indent and line start positions are also used as an identifier for tables and boxes. Figure. 5 shows an example of a PDF used in our contextual sequence extraction algorithm. This algorithm allows us to group the text content of each PDF into different levels of headings, paragraphs and captions, and sequential reading order of these groups. Also it can differentiate between lists and tables and regular text paragraphs.

The *DissectTextHeading* algorithm uses the *ExtractLayouts* function outputs to sort each element of the page based on their font size. Captions are the lines of page which their font size is smaller than the document paragraph font size, whereas headings have font size equal to or larger than the document paragraph font size. If for any line, the font size is equal to the paragraph font size, the indent and line start positions can separate paragraphs form headings.

| ID | Type | SubType | TextContent | BytesContent | Metadata | PageNum | BBox | ParentElementID | PreviousElementID | Level | SequenceID | Document_ID |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1435 | Text | Heading 3 | BLOB | NULL | null | 18 | [72.0, 170.5733642578... | 549 | -1 | 2 | 0 | 1 |
| 1436 | Text | Paragraph | BLOB | NULL | null | 18 | [90.0, 205.3093261718... | 550 | -1 | 2 | 0 | 1 |
| 1437 | Text | Table | BLOB | NULL | null | 18 | [72.0, 345.2292480468... | 550 | 1436 | 2 | 1 | 1 |
| 1438 | Text | Paragraph | BLOB | NULL | null | 18 | [90.0, 387.7092285156... | 550 | 1437 | 2 | 2 | 1 |
| 1439 | Text | Table | BLOB | NULL | null | 18 | [108.0, 448.429199218... | 550 | 1438 | 2 | 3 | 1 |
| 1440 | Text | Table | BLOB | NULL | null | 18 | [72.0, 667.6691894531... | 550 | 1439 | 2 | 4 | 1 |
| 1441 | Text | Caption 1 | BLOB | NULL | null | 18 | [72.0, 740.5924072265... | 551 | -1 | 2 | 0 | 1 |
| 1442 | Text | Paragraph | BLOB | NULL | null | 19 | [90.0, 87.70935821533... | 552 | -1 | 2 | 0 | 1 |
| 1443 | Text | Heading 3 | BLOB | NULL | null | 19 | [72.00000762939453,... | 553 | -1 | 2 | 0 | 1 |
| 1444 | Text | Heading 5 | BLOB | NULL | null | 19 | [72.00000762939453,... | 554 | -1 | 2 | 0 | 1 |
| 1445 | Text | Paragraph | BLOB | NULL | null | 19 | [90.0, 522.5891113281... | 555 | -1 | 2 | 0 | 1 |
| 1446 | Text | Caption 1 | BLOB | NULL | null | 19 | [72.0, 740.5924072265... | 556 | -1 | 2 | 0 | 1 |
| 1447 | Text | Paragraph | BLOB | NULL | null | 20 | [90.0, 87.70935821533... | 557 | -1 | 2 | 0 | 1 |
| 1448 | Text | Heading 7 | BLOB | NULL | null | 20 | [72.0, 154.7150115966... | 558 | -1 | 2 | 0 | 1 |
| 1449 | Text | Table | BLOB | NULL | null | 20 | [90.0, 180.4692993164... | 559 | -1 | 2 | 0 | 1 |
| 1450 | Text | Paragraph | BLOB | NULL | null | 20 | [90.00001525878906,... | 559 | 1449 | 2 | 1 | 1 |
| 1451 | Text | Table | BLOB | NULL | null | 20 | [90.00001525878906,... | 559 | 1450 | 2 | 2 | 1 |
| 1452 | Text | Heading 5 | BLOB | NULL | null | 20 | [72.00001525878906,... | 560 | -1 | 2 | 0 | 1 |
| 1453 | Text | Paragraph | BLOB | NULL | null | 20 | [90.00001525878906,... | 561 | -1 | 2 | 0 | 1 |
| 1454 | Text | Heading 7 | BLOB | NULL | null | 20 | [90.00001525878906,... | 562 | -1 | 2 | 0 | 1 |
| 1455 | Text | Table | BLOB | NULL | null | 20 | [108.00001525878906,... | 563 | -1 | 2 | 0 | 1 |

DOCUMENT_ELEMENTS 17

Fig. 5. Content extraction from PDFs.

### 4.5 Document-to-Product Attribution

In the Document Library Curation part (subsection 4.1), we showed how we are able to classify downloaded documents into manuals, brochures and catalogs. We have also shown in subsection 4.2 how our system is able to automate the collection of product names associated with the various vendors. The combination of having curated a document library and a product name list allows us then to process each document and attribute it to zero, one, or multiple product names, i.e. describing which products this document refers to and for which it presents relevant information. In this section we thus describe our algorithm that conducts attribution of these documents to specific products offered by a vendor. A manual, for example, may be focused on a specific product, whereas a catalog will be attributed to multiple products offered by the vendor. Product attribution sets the stage for processing only the relevant documents when Tally-Vet extracts feature claims for a specific product. Product names are collected as described in the Product Name Scraper section (subsection 4.2). The matching algorithm described herein will result in a product name that we identified from the content or metadata of a given document. Similarly, we can later produce a list of documents relevant to a given product.

To match a given product name to a specific document in Tally-Vet's document library, the algorithm first converts the product name to all lower-case, and then extracts all individual components of the product name. For example, "MyWISE-PaaS" will be represented by the two components 'mywise' and 'paas'. Then any variation of these components, including subsets of components, will be used to curate all combinations of a product name's sub-words.

```
Searching in Advantech advisory
====================================================
Processing PDF : advantech/02131141.pdf
Extracting 139 Sequences
Searching ATCA-9223 product in advantech/02131141.pdf
Searching AMC-4201 product in advantech/02131141.pdf
Searching MIC-5603 product in advantech/02131141.pdf
Searching ATCA-9112 product in advantech/02131141.pdf
Searching FMM-5001F product in advantech/02131141.pdf
Searching FMM-5001Q product in advantech/02131141.pdf
Searching FMM-5006 product in advantech/02131141.pdf
Searching DLT-M8110 product in advantech/02131141.pdf
Searching TREK-570 product in advantech/02131141.pdf
...
```

Fig. 6. An example of the Product Name Search algorithm in a PDF document.

This new list of all patterns is subsequently sorted in descending order based on length. For example, for the above name, the final pattern represented as a regular expression will be ['mywise[â-z0-9]+paas','paas[â-z0-9]+mywise', 'mywise', 'paas']. The first item in the list will match any combination of two parts (mywise and paas) connected with any character such as underline and hyphen. For example, mywise_paas, mywise-paas will be matched with the 'mywise[â-z0-9]+paas' pattern. Algorithm 1 shows the required steps to find all patterns for a given product name. The objective of this algorithm is to eliminate any possible discrepancies between the product name as found on a vendor's website and the product name's usage in the vendor's documents. This will increase the chance of correctly identifying and attributing all documents to all applicable product names.

Once this process completes and Tally-Vet obtained the permutations of all product names, Tally-Vet then conducts a search across all sequences extracted from a given document in order to determine if any product name permutation appears in the document. Product name permutations are considered in order as they appear in the list. The algorithm will therefore first search for all exact matches and compile a list from the matches in the list and the document. Only if no exact match of a given product name is found will the algorithm consider the remaining permutation patterns. Thus, the algorithm stops when a product name's permutation is detected and will no longer consider other permutations of the same product name. Thus, by sorting patterns from longer to shorter length, we ensure that the algorithm is able to consider the best possible matches first before relaxing the match requirements. This ensures accuracy and efficiency. An example of the product search algorithm is shown in Figure. 6.

**Algorithm 1** Find all patterns for a given product name

**Input** : Product Name $N$

**Output**: List of Patterns $AllPatterns$

keywords = Split product name ($N$) into its components (letters, underscore, digits)

$Variants$ = Find combinations of each keyword length:

itertools.combinations (keywords, L)

$Patterns$ = All permutations of a given $Variant$'s elements (itertools.permutations)

▷ Sort patterns based on $Variant$'s length:

$AllPatterns$ = sorted($Patterns$, key=len, reverse=True) **return** $AllPatterns$

The output of this algorithm is then stored by Tally-Vet in its database for later retrieval. This effectively establishes the attribution from product names to documents. It can then be queried in either direction: for finding all product names attributed to a given document, or for finding all documents that are related to a given product name. An example output of such a data retrieval is presented in Figure. 7.

```
All product names attributed to a given document

{'Vendor': 'advantech', 'Products': {'SRP-FPV240', 'WISE-PaaS'}, 'PDF': '02021103.pdf'},
{'Vendor': 'advantech', 'Products': {'AIM-68', 'DLT-V72', 'DS-570', 'DS-570 ', 'UTC-520', 'WISE-PaaS'},
'PDF': 'MyWISE-PaaS-iCS-2020.pdf'},
{'Vendor': 'advantech', 'Products': {'EIS-D210', 'WISE-PaaS'}, 'PDF': 'MyWISE-PaaS-Embedded-2020.pdf'},
{'Vendor': 'advantech', 'Products': {'POC-W102', 'POC-W152', 'POC-W213', 'POC-W243'}, 'PDF':
'01151748.pdf'}

All documents mentioning a given product

{'Vendor': 'advantech', 'Product': 'AIMB-586', 'PDFs': {'01171001.pdf'}},
{'Vendor': 'advantech', 'Product': 'ROM-7510', 'PDFs': {'01171001.pdf', '03131018.pdf'}},
{'Vendor': 'advantech', 'Product': 'IDK-2108 ', 'PDFs': {'01171001.pdf', '2014_ATCA_Brochure_0818.pdf',
'03131018.pdf', 'MyWISE-PaaS-iCS- 2020.pdf', '02091518.pdf', '01151748.pdf', '01091800.pdf'}}
{'Vendor': 'advantech', 'Product': 'IDK-1107', 'PDFs': {'01171001.pdf', '02091518.pdf', '03131018.pdf'}}
```

Fig. 7. An example for the product name attribution.

## 5 PROPOSED TALLY-VET NATURAL LANGUAGE PROCESSING METHODOLOGY

### 5.1 Rationale

TallyVet is a highly complex system, comprised of numerous processing steps. Whereas the algorithms and tasks presented in the previous section form a vital preprocessing aspect for TallyVet, the algorithms described within this section, with their emphasis on NLP, form the core of the tallying process employed by CYVET.

### 5.2 Structured Content Extraction from DOCX

The PDF document format has become the industry standard for web-accessible documents due to its independence from software, hardware, or operating systems. Extracting and analyzing information from PDF documents, however, is a complex procedure. PDFs represent rendered content. Thus, all structural context, such as information being represented in a tabular or list format, are lost through this process. PDFs do not exhibit the concept of tables, footnotes, or lists. Rather, information is represented as character sequences positioned on a page, and drawing objects similarly applied to a page. Thus, it is exceedingly complicated to detect and process structured information directly from PDFs [7]. To increase the reliability of structured text extraction, we decided to convert PDF documents to DOCX format.

#### *5.2.1 PDF-to-DOCX Conversion.*

All documents downloaded from ICS vendor websites are in PDF format. Thus, intelligently converting documents from PDF to DOCX, the Microsoft Word document format, enables far easier extraction of structured data. Adobe® `Acrobat Export PDF` [8] is an Acrobat online service to convert PDF files into editable Word, Excel, or RTF (Rich Text Format) documents. The Adobe PDF Services Application Programming Interface (API) is a cloud-based tool for PDF manipulation. This API service implementation directly uses a Representational State Transfer (REST) interface [20] for both the access control and storage server APIs. The REST API framework, designed to be stateless and based on

HTTP protocols, is context independent [40]. The REST interface accepts and responds using JSON messages over the HTTPS protocol. The API starts with authentication provided by Adobe. For authentication, `JSON Web Tokens (JWT)` [36] were adopted to exchange authentication information into an access token. Using a `JWT` token for authentication in a stateless REST API architecture is a well-established technique that is used in many scientific research efforts [12, 16, 20, 29, 40, 69]. Tally-Vet successfully employs this approach for utilizing Adobe's API system. More specifically, it uses the `PyJWT` python package [37] for `JWT` support, in order to retrieve a request token. This token is then attached to subsequent requests using a custom HTTP header field. Adobe Acrobat API's `JWT` tokens are only valid for a fixed amount of time. For this reason, Tally-Vet will automatically check the validity of this token and refresh it after expiration.

The Tally-Vet document conversion system will retrieve a PDF from its library and automatically submit it to Adobe I/O's Export API, await its conversion, retrieve the result, and store the resulting DOCX file back into Tally-Vet's document library database. This system handles all authentication, conversion, and document management aspects. It automates the process to convert all PDFs (readable and scanned) in our database into DOCX files and automatically rejects any corrupted or otherwise unreadable documents obtained from vendor websites.

#### *5.2.2 DOCX Structure.*

Tally-Vet employs the `python-docx` library [31] to enable it to work with DOCX files. This library provides a convenient object representation of the document including document paragraphs, table objects, headings and captions objects. We define the algorithm 3 to parse each DOCX document, creating a tree structure of its elements and parsing them logically.

```
Text = ' A. Introduction'
Text._element.pPr.numPr.numId.val
Out: 1
Text.text
Out: 'Introduction'
```

Fig. 8. An example list output from `python-docx` library.

In the *ContextualizeSequence* algorithm (Algorithm 3), a high level pseudo-code for sequence extraction from documents is presented. This algorithm parses DOCX documents to extract content from paragraphs, lists, and tables. There are two main challenges for these DOCX document elements: first is finding the style of lists (decimal, letter, roman numeral, and type of bullet points), and second is finding the table structure. Table templates define a guideline to extract the sequences and content of tables logically. We predefined a group of table templates that provide structural maps on how to parse tables in a document. These structural maps then will be used in our *ContextualizeSequence* algorithm to concatenate sequences from different cells and columns of the table and make a logical sequence. The `python-docx` library is able to identify each element in the paragraph with an specific tag. For example, for a list entry in the document, the numerical value for the list identifier can be retrieved from the `Paragraph._element.pPr.numPr.numId.val` tag. However, this value is not always the exact printable character as it appears in the document. An example of the text content (`text`) and `._element.pPr.numPr.numId.val` tag value from `python-docx` library is shown in Figure. 8.

To resolve this problem for lists, our *ContextualizeSequence* algorithm first iterates over the list elements of the document. The *ListExtract* function will find the list's format, the level for each entry in a multi-level list and their hierarchical structure within the document. List elements can appear in any paragraph, whether they are regular paragraphs or paragraphs as part of table cells and other structural elements. Through its hierarchical parsing, the

**Algorithm 2** Structured Sequences from PDF document

```
Input: PDF Page
    ▷ This function will find regular paragraphs, captions and headings, lists and tables in each PDF page.
▷ For each caption and heading, it will use PyMuPDF metadata for the font size, indent and line start positions.
▷ For each regular paragraph, it will search for text spans from PyMuPDF to identify tables and lists.
▷ For images it will use PyTesseract to extract text from each image.
Function Contextual Sequence Extraction(PDFDocument):
  if Page is readable then
        ▷ For each page, we need to first detect if this is a readable PDF, otherwise return
      Orientation = PageOrientation(Page)
      ▷ For each page, find page orientation using BBOX tag in PyMuPDF. This is used to order different blocks
      (images, texts)
      Layouts = ExtractLayouts(Page)
      ▷ Find layout information for the PDF page including font size, text span, and flag identifier with PyMuPDF.
      Tags = ExtractHeadingSectionTags(Page)
      ▷ Use Layouts from ExtractLayouts to identify heading, caption and regular text elements.
      Positions = ExtractLinePositions(Page)
      ▷ Find positions for Paragraphs, Lists, an Tables with PyMuPDF identifiers and regular expression.
      for Element in Page.Elements do
        if Element.Type == Image then
          yield PyTesseract(Element)      ▷ This function extracts raw text from images with PyTesseract
          return
        if Element.Type == Text then
          for SubElem in Element do
            if SubElem.Type == Text then
              if Level== 0 then
                yield DissectTopLevel(Page, Element, Tags, Layouts, Positions)
                ▷ Takes top level text elements (text blocks), and breaks elements down into headings
                (independent of level) and regular text.
              for Level >=1 do
                if SubElem.Type == Heading Or Caption then
                  yield DissectTextHeading(Page, Element, SubElem, Tags, Layouts, Positions)
                  ▷ This will go through SubElems of each Heading Element type.
                  ▷ This sorts headings and captions based on their font size.
                if SubElem.Type == Regular then
                  yield DissectTextRegular(Page, Element, SubElem, Tags, Layouts, Position)
                  ▷ This will go through SubElems of each TextElement type.
                  ▷ This groups text into regular paragraphs, tables, and lists.
  else
    return
```

*ContextualizeSequence* algorithm can detect all of them. An example of a page including lists with different formats and the output of our algorithm is available in Figure. 9. The tree structure clearly shows the identifier tags our algorithm added to each line of the list.

Detecting tables is a crucial step in document analysis, since tables are often used to present essential information in a structured way to the reader. The main objective in parsing tables is to render a set of flattened sequences that are

**An example document with different list levels**

```
A. Introduction
  1.  Title:     Cyber Security — Physical Security of BES Cyber
                 Systems
  2.  Purpose:   To manage physical access to Bulk Electric System
                 (BES) Cyber Systems.
3.     Applicability:
  3.1.  Functional Entities: For the purpose of the requirements
        contained list of functional entities.
     3.1.1 Balancing Authority
     3.1.2 Distribution Provider that owns one or more of the
           following Facilities, systems, and equipment for
           restoration of the BES:
        3.1.2.1 Each underfrequency Load shedding (UFLS):
           3.1.2.1.1 is part of a Load shedding program that is
                     subject
  4.   Background:
       Standard CIP-006 exists as part of a suite of CIP Standards
       related to cyber security.
```

**Tree structure and sequences output from the algorithm**

```
=TREE VIZ===================================
  LPAR: [num]    A.   Introduction
  LPAR: [num]    1.   Title:    Cyber Security — Physical Security of BE...
  LPAR: [num]    2.   Purpose:          To manage physical access to Bulk Elec...
  LPAR: [num]    3.   Applicability:
  LPAR: [num]    3.1.   Functional Entities: For the purpose of the req...
  LPAR: [num]    3.1.1   Balancing Authority
  LPAR: [num]    3.1.2   Distribution Provider that owns one or more of ...
  LPAR: [num]    3.1.2.1    Each underfrequency Load shedding (UFLS):
  LPAR: [num]    3.1.2.1.1   is part of a Load shedding program that is subj...
  LPAR: [num]    4.   Background:
  PARA: Standard CIP-006 exists as part of a suite of C...
=TREE SEQUENCE LIST===================================
A.          Introduction
1.          Title:        Cyber Security — Physical Security of BES Cyber Systems
2.          Purpose:      To manage physical access to Bulk Electric System (BES) Cyber Systems.
3.          Applicability:
3.1.        Functional Entities: For the purpose of the requirements contained list of functional entities .
3.1.1       Balancing Authority
3.1.2       Distribution Provider that owns one or more of the following Facilities, systems, and equipment for
restoration of the BES:
3.1.2.1      Each underfrequency Load shedding (UFLS):
3.1.2.1.1   is part of a Load shedding program that is subject
4.          Background:
Standard CIP-006 exists as part of a suite of CIP Standards related to cyber security.
```

Fig. 9. An example output from a list structure extracted from DOCX documents.

comprised of the content spanning one or more table cells, by logically concatenating table cells and column information together. Figure. 10 shows an example specification table from a document. This table contains important information which needed to be parsed logically to maintain the meaning of sentences.

The python-docx library allows us to access tables and their cells and columns directly. Our algorithm for table cell content concatenation begins with defining a list of templates for different types of tables contained in a document. This takes advantage of the fact that vendors typically represent tabulated data in a similar format across their various documents. These templates define a way to describe how to parse the content of the table and extract sequences from them. Templates are defined in two main steps; the first step is filtering/matching, and the other is the data extraction/labeling step. The algorithm identifies a match between the template and the table only if the labelling passes the filter stage ($MatchTemplate$ function). Figure. 11 shows an example script we defined for a table template. This script starts with defining labels for cells based on their rows and columns positions, number of spans, and background color. The filter section then will be used to set up a filter for each label. Filters are used to decide if a template applies to a table or not. The last section in table template script is responsible for setting up a format section. Formats are used to produce a sequence as output by pulling content from multiple cells together, based on their labels; the first Format that works (we find all required sections by labels and search limiters) wins and the process stops for this cell.

| CIP-006-6 Table R1 Physical Security Plan | | | |
|---|---|---|---|
| Part | Applicable Systems | Requirements | Measures |
| 1.4 | High Impact BES Cyber Systems and their associated:<br>1. EACMS; and<br>2. PCA<br><br>Medium Impact BES Cyber Systems with External Routable Connectivity and their associated:<br>1. EACMS; and<br>2. PCA | Monitor for unauthorized access through a physical access point into a Physical Security Perimeter. | An example of evidence may include, but is not limited to, documentation of controls that monitor for unauthorized access through a physical access point into a Physical Security Perimeter. |

Fig. 10. An example of a specification table.

**Algorithm 3** Contextualized Sequences from DOCX document.

```
Input: DOCX Element
    ▷ This function will find regular and heading paragraphs, list elements, and table elements in each document.
    ▷ For each table element, it will search for cell templates based on their properties (font size, color, cell spans). Table elements might include list elements and regular paragraphs.
    ▷ For each list element, it will search for list style and levels.
Function Flattened Sequences(DOCXDocument):
    if Element has a child then
        for Child in Element.children do
            if Element.Type == Table then
                Template=MatchTemplate(Element) ▷ This function will search a predefined set of table templates for the best match.
                ▷ If it finds any matches for the current element, it will return a label that specifies the structure behind the element (tables can match different templates for each cell, and this function can return all the matching labels), otherwise it will return None
                if Template is None then
                    yield Element.TEXT
                else
                    for SubElem in TableExtract(Element, Template) do
                        yield SubElem    ▷ Each element might be a list or a regular paragraph.
                        ▷ Here based on the element type and the returned label for the table template, it will determine the structure for the contextualized sequence.
                        return
            else
                if Element.Type == List then
                    for SubElem in ListExtract(Element) do
                        yield SubElem ▷ For each document, the ListExtract function will search in all list styles that it can find.
                        ▷ List style definition includes number formatting (decimal, letter, roman numeral, and type of bullet points)
                        ▷ List level is defined based on tag identifiers in python-docx library.
                        return
                else
                    yield (Element.TEXT)
    else
        yield (Element.TEXT)
        return
```

For each table in the document, the *ContextualizeSequence* algorithm conducts a search for a suitable table matching template with *MatchTemplate* function. Each template indicates the table heading location, sub-heading (if it has any), etc. (for an example see the labeling section in Figure. 11) and enables the algorithm to find the cells and column headings that should be concatenated to generate a sequence from that content and to identify the table logic direction (for an example see the filtering section in Figure. 11). Table direction here refers to the position of each heading or sub-heading, whether the headings are located in the first cell of each row (row-wise) or the top columns (column-wise). The *TableExtract* function (see the algorithm 3) then uses this element and the template to go over the table, label

```
set=self.CreateAndAddTemplateSet("temp_example")
template=self.CreateAndAddTemplate(set,"type1_planning")
#the labels for the template
#row and column indexes are 0-based
lbl_title=self.CreateLabel(template,"title")
self.CreateLabelPositionRule(lbl_title,0,0,0,0,3,100,1,1)
self.CreateLabelBkgColorIncludeRule(lbl_title,['1F4B81'])
lbl_colheading1=self.CreateLabel(template,"colheading1")
self.CreateLabelPositionRule(lbl_colheading1,0,0,1,1,1,2,1,1
self.CreateLabelBkgColorIncludeRule(lbl_colheading1,['5D85A9'])
lbl_colheading=self.CreateLabel(template,"colheading")
self.CreateLabelPositionRule(lbl_colheading,1,100,1,1,1,2,1,1)
self.CreateLabelBkgColorIncludeRule(lbl_colheading,['5D85A9'])
lbl_rowheading=self.CreateLabel(template,"rowheading")
self.CreateLabelPositionRule(lbl_rowheading,0,0,2,100,1,1,1,3)
self.CreateLabelBkgColorIncludeRule(lbl_rowheading,['FFFFFF','auto',None])
lbl_content=self.CreateLabel(template,"content")
self.CreateLabelPositionRule(lbl_content,1,100,2,100,1,100,1,100)
self.CreateLabelBkgColorIncludeRule(lbl_content,['FFFFFF','auto',None])
#the filter for the set
filter=self.CreateFilter(template)
self.CreateFilterConditionMayNotHaveEmpty(filter)
self.CreateFilterConditionMayHave(filter,lbl_title['name'],1,1)
self.CreateFilterConditionMayHave(filter,lbl_colheading1['name'],1,1)
self.CreateFilterConditionMayHave(filter,lbl_colheading['name'],1,100)
self.CreateFilterConditionMayHave(filter,lbl_rowheading['name'],1,100)
self.CreateFilterConditionMayHave(filter,lbl_content['name'],1,100)
#the formatters for the set
formatter=self.CreateFormat(template,"content1",lbl_content['name'])
self.CreateFormatFinderAny(formatter,lbl_colheading1['name'],"%s ")
self.CreateFormatFinderHorizontal(formatter,lbl_rowheading['name'],-1,-100,0,"%s; ")
self.CreateFormatFinderVertical(formatter,lbl_colheading['name'],-1,-100,0,"%s: ")
self.CreateFormatAnchorElem(formatter,"%s",False)
```

Fig. 11. A table template example script.

elements, extract cell content, and then concatenate elements together as described by the template (for an example see the formatter section in Figure. 11). The resulting content is a flattened sequence that, for example, takes the form $HEADING[>> SUBHEADING] : CONTENT$ for iterable content elements. Figure. 12 shows an example table, the table structure from the table template (TREE VIZ section), and the parsed flattened sequences produced by our algorithm.

The *ContextualizeSequence* algorithm we proposed in this paper is based on `python-docx` Python package, and is able to identify levels in lists (Figure. 9) and tables (Figure. 12), and can extract and annotate contextualized sequences from DOCX documents. Also, our *StructuredSequence* algorithm is based on the `PyMuPDF` Python package and can identify hierarchical levels of different elements in PDF documents. An output example is shown in Figure. 5.

### 5.3 Claim Detection

One of the primary purposes for Tally-Vet is to identify VSF claims from all ICS device documentation of a given vendor or for a given product. In our previous paper [1], we introduced CyBERT, a classification model for labeling claims by

**A table example from a sample**

| CIP-006-6 Table R1 Physical Security Plan | | | |
|---|---|---|---|
| Part | Applicable Systems | Requirements | Measures |
| 1.4 | High Impact BES Cyber Systems and their associated:<br>1. EACMS; and<br>2. PCA<br><br>Medium Impact BES Cyber Systems with External Routable Connectivity and their associated:<br>1. EACMS; and<br>2. PCA | Monitor for unauthorized access through a physical access point into a Physical Security Perimeter. | An example of evidence may include, but is not limited to, documentation of controls that monitor for unauthorized access through a physical access point into a Physical Security Perimeter. |

**Tree structure and flattened sequences output from the algorithm**

```
=TREE VIZ==================================
  TABL:
    CELL: R= 0 C= 0 RS= 1 CS= 4 COLOR= 1F4B81
      PARA: CIP-006-6 Table R1 Physical Security Plan
    CELL: R= 1 C= 0 RS= 1 CS= 1 COLOR= 5D85A9
      PARA: Part
    CELL: R= 1 C= 1 RS= 1 CS= 1 COLOR= 5D85A9
      PARA: Applicable Systems
    CELL: R= 1 C= 2 RS= 1 CS= 1 COLOR= 5D85A9
      PARA: Requirements
    CELL: R= 1 C= 3 RS= 1 CS= 1 COLOR= 5D85A9
      PARA: Measures
    CELL: R= 2 C= 0 RS= 1 CS= 1 COLOR= FFFFFF
      PARA: 1.4
    CELL: R= 2 C= 1 RS= 1 CS= 1 COLOR= FFFFFF
      PARA: High Impact BES Cyber Systems and their associa...
      LPAR: [num]    1.   EACMS; and
      LPAR: [num]    2.   PCA
      PARA: Medium Impact BES Cyber Systems with External R...
      LPAR: [num]    1.   EACMS; and
      LPAR: [num]    2.   PCA
    CELL: R= 2 C= 2 RS= 1 CS= 1 COLOR= FFFFFF
      PARA: Monitor for unauthorized access through a physi...
    CELL: R= 2 C= 3 RS= 1 CS= 1 COLOR= FFFFFF
      PARA: An example of evidence may include, but is not ...
=Flattened SEQUENCE LIST==================================
Part 1.4; Applicable Systems: High Impact BES Cyber Systems and their associated:
Part 1.4; Applicable Systems: 1.       EACMS; and
Part 1.4; Applicable Systems: 2.       PCA
Part 1.4; Applicable Systems:
Part 1.4; Applicable Systems: Medium Impact BES Cyber Systems with External Routable
Connectivity and their associated:
Part 1.4; Applicable Systems: 1.       EACMS; and
Part 1.4; Applicable Systems: 2.       PCA
Part 1.4; Requirements: Monitor for unauthorized access through a physical access point into a
Physical Security Perimeter.
Part 1.4; Measures: An example of evidence may include, but is not limited to, documentation
of controls that monitor for unauthorized access through a physical access point into a Physical
Security Perimeter.
```

Fig. 12. An example output of table structure from DOCX documents.

fine-tuning the BERT-base (Bidirectional Encoder Representations from Transformers [19]) language model on our cybersecurity domain dataset. From the ICS device document dataset, as we explained in the subsection on Document Library Curation (subsection 4.1), all sequences were extracted with the methods and libraries defined in both the contextual sequence extraction and structured content extraction sections. These sequences are obtained from a wide range of vendors and device documents, and we manually labeled a subset of these as “Claim” and “NotClaim”. Any sequence that represents cybersecurity-related claims about features of the product being evaluated was referred to as "Claim". The resulting dataset was used to train a classifier that detects any type of claims about device features from these documents. Figure. 13 illustrated the labeled dataset we used for Tally-Vet. Tally-Vet uses two classifier, the first classifier we call CyBERT, which focuses only on identifying "Claim" sequences from others (Figure. 13 top row). The claim sequences itself can further be divided into three types; cybersecurity claims, device claims, and generic claims. Hence, our second classifier is a claims type classifier (Figure. 13 bottom row), and it allows us to differentiate and detect the cybersecurity claims from all other claims. That classifier is applied once a sequence is determined to be a claim.

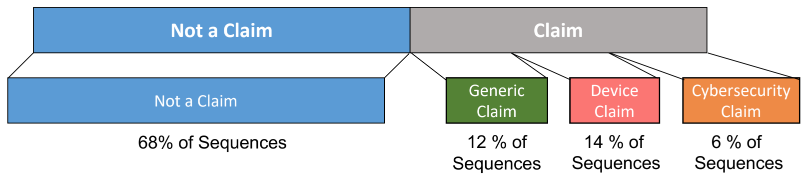


Fig. 13. An illustration of Tally-Vet labeled dataset and classes used to train CyBERT and Claim classifier.

For the fine-tuning process of our claim classifier we conducted extensive experimentation to optimize hyperparameters, including the learning rate, the number of dense layers and their corresponding configuration such as drop-out rate and number of neurons. Fine-tuning all hyperparameters of the resulting BERT classifier model led to building our CyBERT classifier, which can detect sequences related to feature claims with a 94.4% accuracy. CyBERT is intended to be used as a cybersecurity-specific classification model for detecting claim sequences from a large pool of sequences extracted from ICS device documents.

The trained model has 12 encoders (from BERT-Base model) and 3 dense layers stacked on top of the encoders. The best learning rate for training our claim classifier model based on this architecture and dataset was determined to be 4e-06. The Claim classifier's overall accuracy is 94% with 93% F1 weighted core. This model is able to detect claim sequences with 92% accuracy (Figure. 14). Table. 2 shows the detailed classification metric reports for each sequence label. In Figure. 14, we plot the normalized confusion matrix for our claim classifier. A confusion matrix shows the actual and predicted results (or results for correct and incorrect predictions) of a classifier [80].

Identifying "Claim" sequences subsequently enables us to not only extract a list of claimed features for a device, but also to compare feature claims to cybersecurity requirements, which is the key to our Tally-Vet operation for OT infrastructure vetting. CYVET's semi-automated vetting system for ICS cybersecurity auditing relies on this Claim classifier.

Table 2. Classification Report for the Claim Classifier with Fine-Tuning BERT

| Class Label | Precision | Recall | F1-score | Accuracy |
|---|---|---|---|---|
| Claim | 0.92 | 0.89 | 0.91 | |
| Not a Claim | 0.95 | 0.75 | 0.96 | |
| Weighted Average | 0.94 | 0.94 | 0.94 | **0.94** |

When researching our claims classifier for TallyVet we conducted extensive experiments in order to maximize the accuracy of our claims classifier, including the architecture selection, hyperparameter selection, and studying the effects of randomness. We are presenting the details of our findings in the Analysis and Discussion section (section 6) further below.

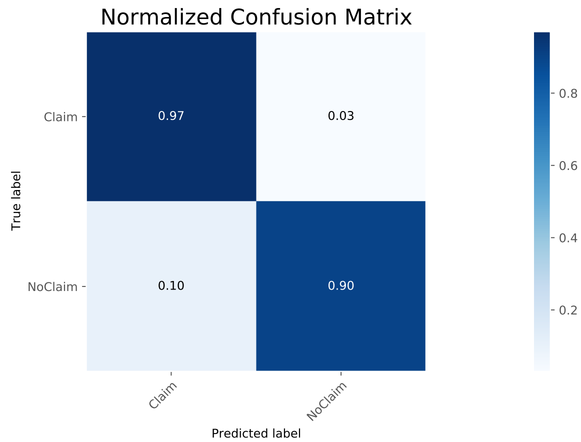


Fig. 14. Normalized confusion matrix for the Claim classifier.

### 5.4 Feature Attribution using Sentiment Analysis

In this section, we explain the process Tally-Vet uses for attributing features to products. This process broadly involves three steps: We first detect cybersecurity feature claim sequences from each document attributed to a given product. In the previous section we described how claim sequences are detected using our CyBERT claim classifier. We next perform an NLP sentiment analysis for these claim sequences in order to determine Feature Support Indication: an expression of whether the claim indicates that the device does or does not support this feature. Finally we tabulate and reconcile feature claims across all documents related to a given product, including conflict resolution, which involves features that in some document are claimed as supported and in others claimed as not supported.

In the Definition below we present and define our concept for Feature Support Indication:

**Definition 5.1** (Feature Support Indication)**.** A value assigned to an extracted feature in a specific document sequence that expresses whether the feature is supported or not supported by the device.

Feature Support Indication can be any value between 0 and 1. Values greater than 0.5 indicate that the sentence evaluation for the specified feature is indicating support, with values approaching 1 indicating higher confidence in this sentiment, whereas values below 0.5 indicate lack of support for this feature, with values approaching 0 indicating higher confidence in this sentiment. This value was defined as the sentiment of the feature claim sequence. The `NLTK Toolkit` python library does support the VADER sentiment analysis algorithm. The VADER sentiment analysis allows us to update its lexicon dictionary weights based on our project's definition. The impression behind each sequence will be expressed as the polarity score of the sequence. For this purpose, we update the polarity scores for some specified words. With this approach, VADER can contextualize the use of those terms within the scope and definition of our project [35].

Tally-Vet manages and defines features as a tree structure, with each branch indicating a more specific interpretation or incarnation of a given feature. This tree structure is a way to group features logically. For example, a "server-based authentication" could have "RADIUS" and "LDAP" as sub-branches. Based on our definition of this feature tree, each sub-feature can also have properties. For example, a sub-feature "General SSH" can have a property "Generic SSH Support" and a property "SSH v2 Support". Here, the "generic" could be used to detect any support of SSH, irrespective of its version, whereas the "v2" property could be used to specifically indicating support for SSH v2. Each feature, sub-feature and property is associated with keywords with NLP techniques. Tally-Vet attributes features using these keywords combined with NLP techniques to detect them. Additionally, a particular feature uses a name space nomenclature to represent itself. Tally-Vet compiles them into a feature expression: $< feature >::< sub - feature >::< property >::$ (e.g. "serverauth::LDAP::gen::").

NLP feature search within extracted sequences targets the identified set of keywords. This NLP keyword search can be case-insensitive or case-sensitive. The algorithm starts with splitting designated keywords and search for any exact matches between each partition and sentences. The final report will be the sequences that match these keyword sets (matches in features, sub-features, and properties).

For each sequence containing a defined feature, the algorithm then conducts a sentiment analysis. This assigns this instance of the feature occurrence a Feature Support Indication score. Next, the algorithm compiles a list of all features that were detected in this document and performs a deconflicting step: If the same feature appears multiple times, with conflicting support indications, Tally-Vet decides that a negative feature support indication wins over a positive indication. Hence, conflicting feature claims resolve as unsupported feature claims. Figure. 15 shows an output example of feature attribution algorithm for a sample PDF. The results show that the sample PDF does support password authentication feature (`pwdauth`), with generic sub-feature (`gen`) and the `strong` property, in combination indicating a feature claim of "strong password authentication". The other feature support extracted from this PDF is server authentication, with the "LDAP" sub-feature and the generic property. The last supported feature for the device represented by this PDF is SSH, with generic sub-feature and property. This sample PDF does not have any claim/sequence with nonsupporting label.

```
Feature Support Verdict Processing on Feature Sequences
       Processing PDF :   PF00253.pdf
          Sequence-Level Verdict Processing:
             Completed verdict processing
          Document-Level Verdict Processing:
             these features are supported:
               pwdauth::gen::strong::
               serverauth::LDAP::gen::
               ssh::gen::gen::
             these features are claimed to lack support:
               <none>
```

Fig. 15. An output example of feature attribution algorithm.

After processing each document for a given product with our feature attribution algorithm, Tally-Vet compiles a list of feature claims across these documents, once again applying a deconflicting step that resolves Feature Support Indication conflicts across the documents.

In a final processing step to feature attribution, Tally-Vet will use the resulting list of feature claims and compare them against a list of Customer Requirements (CR). These are requirements indicated by industry specifications that are imposed on devices and customers in order to obtain necessary cybersecurity compliance certification. This is especially important for compliance reporting by critical infrastructure sectors such as the energy sector, and the key application of CYVET.

### 5.5 CYVET's use of Feature Claims

The Tally-Vet component of the CYVET system identifies, verifies, and tabulates vendor-claimed features against requirements provided by relevant industry standards in eight steps, and then provides the compiled information to the Test-Vet component. Test-Vet focuses on the validation of specific features identified by the Tally-Vet. Validation encompasses two aspects: a) validation of feature availability and b) validation of correct implementation without the presence of known flaws. To accomplish either of these aspects, Test-Vet needs to execute actual hardware test scripts. Hence, Tally-Vet is a vital component in CYVET in general, and in particular in preparation of performing the Test-Vet functionality. Tally-Vet's overall processing pipeline steps can be summarized as shown below:

(1) **Product Document Retrieval and Sequence Extraction:** In this step, all documents relevant to the vetting of a given product from a given vendor will be retrieved from the data repository. All sequences for each document then need to be extracted in order to prepare the data for the next step.
(2) **NLP Feature Search within Extracted Sequences:** Tally-Vet then uses NLP techniques to identify sequences that contain feature expressions to find matches between features and sequences. The output from this step will be any sequences from the document that match feature expressions.
(3) **NLP Claims Detection on Feature Sequences:** Next, we classify any sequences containing feature expressions to detect if they appear to be claims about this feature. This utilizes our CyBERT claim classifier. The result from this step is a set of sequences that indicate cybersecurity feature claims.
(4) **NLP Sentiment Analysis on Feature Claims:** The sequences resulting from the previous step then undergo a sentiment analysis, in order to obtain the Feature Support Indication, a value that indicates if the feature is claimed to be supported, or claimed to lack support by the device. This polarity score assigned to each sequence ranges from 0 (indicating a high confidence in a lack of the feature) to 1 (indicating a high confidence that the feature is claimed to be present).
(5) **Feature Support Verdict Processing on Feature Sequences:** This step is comprised of two aspects: It first compiles a list of claimed features identified within each document, including the resolution of conflicting claims related to the same feature. It then further compiles a list across all relevant documents that were processed for the given device, again resolving any conflicts arising from this step. The end result is a list of claimed features and their support indication for the given device, across all relevant documents available for the device.
(6) **Feature Reporting:** This step combines the list from the previous step with the customer requirement level (CR-Level) provided by the end user for each feature. It reports the features that were detected and their required level. These CR-Levels are either "optional" or "required".
(7) **Feature Support Check:** Tally-Vet then compiles a list based on feature support indication and customer requirements. The resulting report is the primary output of Tally-Vet and a key component in CYVET's vetting process. The report output is classifying features into three lists:
    (a) All required and supported features,
    (b) All required but not supported features,
    (c) All required but not mentioned features.

    Devices will pass the feature support check step only if all customer-required features are in the "required and supported" category.
(8) **Compiling required Test-Vet List:** For all features that are claimed as supported, we then compile a list of tests to be executed by Test-Vet. The compiled list of tests includes validation tests designed to verify that the

claimed feature is indeed implemented and also tests that check for potential flaws in their implementation. Similar to Tally-Vet's vendor identification process, a similar process is executed periodically by CYVET to identify known vulnerabilities published to websites such as ICS-Cert. Our system then uses NLP techniques to identify the defined features within Tally-Vet that these published vulnerabilities are related to. We thus compile a list of known flaws for each feature. Test-Vet curates a test script library that is not only used to verify that a claimed feature is available on a given device, but also includes tests written to evaluate whether a feature's implementation on a given device exhibits one of these known flaws. The implementation of this feature is part of Test-Vet and our future work.

In Figure. 16 we demonstrate an example of the Tally-Vet processing pipeline. Here, an example of a report shows the results of each step in the Tally-Vet pipeline for the product RTAC3530 from vendor SEL. There were five documents related to this product in our data repository. Final results show that the device passes the required features check, with all required features claimed as supported by the device, and it also provides a list of tests for each feature to verify feature presence and detect potential flaws. This list would subsequently be passed to Test-Vet for further vetting of the device itself.

## 6 ANALYSIS AND DISCUSSION

As indicated earlier, to the best of our knowledge there is currently no framework available or published with capabilities similar to those developed for our CYVET system, its unique target dataset or algorithm collection.

Consequently, we could not find any comparison basis with other approaches and we thus focused our analysis efforts on demonstrating and presenting the particular data obtained and processed by our approach, and the novel insights gained from our approach.

Therefore, within the sections detailing each of our contributed methods and algorithms we are presenting the relevant results from our testing to demonstrate their merit and effectiveness.

In the following subsections, we compare the performance of our novel algorithm for sequence extraction from documents and our sentiment analysis for feature attribution to other related methods. We also discuss the performance improvements of our approaches compared to similar methods discussed in related works.

### 6.1 Structured Content Extraction

Data extraction and processing from PDF files has always been challenging. One of the main challenges here is detecting and extracting tables. Among the required associated techniques we focus on table border detection and cell structure recognition [17]. The term 'table border detection' refers to identifying the position of a table via recognition of its borders. The 'cell structure recognition' refers to determining logical relationships between cells and their contents inside a table. Several characteristics of tables make these processes challenging, including the potential absence of border lines for table and cell borders, merged rows and columns, and the spreading of tables across multiple pages.

There are three approaches in the literature to handle table detection in documents: conventional rule-based [30, 87], metadata extraction [6, 31, 57] and machine learning and deep learning approaches [5, 25, 41, 47, 89]. Machine learning approaches and conventional rule-based models are mainly focuses on table and cell detection. Whereas, the main focus of metadata extraction Python libraries [6, 31, 57] are on extracting plain text from different elements in a PDF, such as tables.

```
================================================================================
   TALLYVET PROCESSING PIPELINE
================================================================================



==============================================================================
    STEP 1) Product Document Retrieval and Sequence Extraction
       For product RTAC3530 from vendor SEL we will process these
       documents:
           3530-4_PF00253.pdf
           RTAC Product Family.pdf
           3530_RTAC_PF00174.pdf
           3530-4_DS_20200224-2.pdf
           3530_DS_20200224.pdf
       Processing PDF :   3530-4_PF00253.pdf
          We extracted 162 sequences
       Processing PDF :   RTAC Product Family.pdf
          We extracted 89 sequences
       Processing PDF :   3530_RTAC_PF00174.pdf
          We extracted 119 sequences
       Processing PDF :   3530-4_DS_20200224-2.pdf
          We extracted 136 sequences
       Processing PDF :   3530_DS_20200224.pdf
          We extracted 132 sequences

==============================================================================
    STEP 2) NLP Feature Search within Extracted Sequences
       Processing PDF :   3530-4_PF00253.pdf
          We found 8 sequences indicating relevant features
       Processing PDF :   RTAC Product Family.pdf
          We did not find any sequences indicating relevant features
       Processing PDF :   3530_RTAC_PF00174.pdf
          We found 4 sequences indicating relevant features
       Processing PDF :   3530-4_DS_20200224-2.pdf
          We found 7 sequences indicating relevant features
       Processing PDF :   3530_DS_20200224.pdf
          We found 7 sequences indicating relevant features

==============================================================================
    STEP 3) NLP Claims Detection on Feature Sequences
       Processing PDF :   3530-4_PF00253.pdf
          We found 5 Claims
          We found 5 Cybersecurity and Device Claims
       Processing PDF :   RTAC Product Family.pdf
          There are no feature sequences to process for this document
       Processing PDF :   3530_RTAC_PF00174.pdf
          We found 2 Claims
          We found 2 Cybersecurity and Device Claims
       Processing PDF :   3530-4_DS_20200224-2.pdf
          We found 6 Claims
          We found 6 Cybersecurity and Device Claims
       Processing PDF :   3530_DS_20200224.pdf
          We found 6 Claims
          We found 6 Cybersecurity and Device Claims

==============================================================================
    STEP 4) NLP Sentiment Analysis on Feature Sequences
       Processing PDF :   3530-4_PF00253.pdf
          Completed sentiment analysis
       Processing PDF :   RTAC Product Family.pdf
          There are no feature sequences to process for this document
       Processing PDF :   3530_RTAC_PF00174.pdf
          Completed sentiment analysis
       Processing PDF :   3530-4_DS_20200224-2.pdf
          Completed sentiment analysis
       Processing PDF :   3530_DS_20200224.pdf
          Completed sentiment analysis

==============================================================================
    STEP 5) Feature Support Verdict Processing on Feature Sequences
       Processing PDF :   3530-4_PF00253.pdf
          Sequence-Level Verdict Processing:
             Completed verdict processing
          Document-Level Verdict Processing:
             these features are supported:
               pwdauth::gen::gen::
               pwdauth::gen::strong::
               serverauth::LDAP::gen::
               ssh::gen::gen::
             these features are claimed to lack support:
               <none>
       Processing PDF :   RTAC Product Family.pdf
          There are no feature sequences to process for this document

    Processing PDF :   3530_RTAC_PF00174.pdf
       Sequence-Level Verdict Processing:
          Completed verdict processing
       Document-Level Verdict Processing:
          these features are supported:
            pwdauth::gen::gen::
            pwdauth::gen::strong::
            serverauth::LDAP::gen::
            ssh::gen::gen::
          these features are claimed to lack support:
            <none>
    Processing PDF :   3530-4_DS_20200224-2.pdf
       Sequence-Level Verdict Processing:
          Completed verdict processing
       Document-Level Verdict Processing:
          these features are supported:
            pwdauth::gen::gen::
            pwdauth::gen::strong::
            serverauth::LDAP::gen::
            serverauth::RADIUS::gen::
            ssh::gen::gen::
          these features are claimed to lack support:
            <none>
    Processing PDF :   3530_DS_20200224.pdf
       Sequence-Level Verdict Processing:
          Completed verdict processing
       Document-Level Verdict Processing:
          these features are supported:
            pwdauth::gen::gen::
            pwdauth::gen::strong::
            serverauth::LDAP::gen::
            serverauth::RADIUS::gen::
            ssh::gen::gen::
          these features are claimed to lack support:
            <none>
   Merging Feature Support at Product Level...
       Completed Feature Support Processing
================================================================================
  STEP 6) Feature Reporting
   SUPPORTED:
     pwdauth::gen::gen:: (from 3530-4_PF00253.pdf) CR-Level: optional
     pwdauth::gen::strong:: (from 3530-4_PF00253.pdf) CR-Level: required
     serverauth::LDAP::gen:: (from 3530-4_PF00253.pdf) CR-Level: optional
     ssh::gen::gen:: (from 3530-4_PF00253.pdf) CR-Level: required
     serverauth::RADIUS::gen:: (from 3530-4_DS_20200224-2.pdf) CR-Level: required
   UNSUPPORTED:
     <none>
================================================================================
  STEP 7) Feature Support Check
   REQUIRED AND SUPPORTED:
     pwdauth::gen::strong::  (from 3530-4_PF00253.pdf )
     ssh::gen::gen::  (from 3530-4_PF00253.pdf )
     serverauth::RADIUS::gen::  (from 3530-4_DS_20200224-2.pdf )
   REQUIRED BUT NOT SUPPORTED:
     <none>
   REQUIRED BUT NOT MENTIONED IN DOCUMENTS:
     <none>
   >>> DEVICE PASSES INITIAL TEST:    all required features claimed
=================================================================================
  STEP 8) Compiling Required TestVet List
   List of Tests:
     test_feature_pwdauth (for feature pwdauth::gen::gen::)
     test_potentialflaw_ICSMA-21-215-01 (for feature pwdauth::gen::gen::)
     test_potentialflaw_ICSA-21-208-03 (for feature pwdauth::gen::gen::)
     test_potentialflaw_ICSMA-21-012-01 (for feature pwdauth::gen::gen::)
     test_potentialflaw_ICSA-18-240-04 (for feature pwdauth::gen::gen::)
     test_feature_serverauth (for feature serverauth::LDAP::gen::)
     test_feature_ldap (for feature serverauth::LDAP::gen::)
     test_potentialflaw_ldap_ICSA-17-353-01 (for feature serverauth::LDAP::gen::)
     test_potentialflaw_ldap_ICSA-21-061-03 (for feature serverauth::LDAP::gen::)
     test_potentialflaw_ldap_ICSMA-21-175-01 (for feature serverauth::LDAP::gen::)
     test_feature_ssh (for feature ssh::gen::gen::)
     test_potentialflaw_ICSA-21-068-02 (for feature ssh::gen::gen::)
     test_potentialflaw_ICSA-21-082-02 (for feature ssh::gen::gen::)
     test_potentialflaw_ICSA-13-091-01 (for feature ssh::gen::gen::)
```

Fig. 16. Tally-Vet Processing Pipeline Example

In 2016, Hao *et al.* [30] combined CNN with a set of pre-defined rules to compute region proposals. This model fails to detect table regions for merged cells and columns [5, 25]. Gilani *et al.* [25] proposed a model based on Faster R-CNN to improve Hao *et al.* [30] model. Despite outperforming previous table detection techniques, this technique ignores visible features of the table and fails to detect cell structures and spanned cells [5]. Arif *et al.* [5] improved the Faster R-CNN model introduced by Gilani *et al.* [25] by considering foreground and background features of PDFs. Using a color-coded and transformed document image, Arif *et al.* [5] used Region Proposal Networks (RPNs) followed by fully connected neural networks to detect tabular regions. Khan *et al.* [41] used the CNN model introduced by Gilani *et al.* [25] to detect table boundaries. Zheng *et al.* [89] introduced a Global table extractor (GTE) model based on object detector techniques, which themselves are based on neural networks that analyze document images to find tables and their structure. Lee *et al.* [47] formulate tables as planar graphs based on cell regions. By solving a constrained optimization problem they calculate vertex confidence maps and line fields based on the heatmap regression networks.

Although the mentioned methods made progress towards understanding complex structured tables, several assumptions were made, such as that accurate word bounding boxes were available and that accurate document text could be used as additional inputs [66].

Table 3. Comparison of our method with other available approaches

| | **Model** | **Input Type** | **Output** | **Weakness** |
|---|---|---|---|---|
| Hao *et al.* (2016) [30] | Rule Sets +CNN | PDF | Table Boundary Detection | Fails to detect cell structures<br>Relies on Templates<br>Fails to return text |
| Gilani *et al.* (2017) [25] | Faster R-CNN | PDF | Table Boundary Detection | Fails to detect cell structures<br>Fails to return text |
| Arif *et al.* (2018) [5] | Faster R-CNN +<br>Region Proposal<br>Network (RPN) | PDF | Table Boundary Detection<br>Columns and Rows Detection | Fails to return text<br>Fails to detect cell structures |
| Khan *et al.* (2019) [41] | Faster R-CNN +<br>Gated Recurrent<br>Unit (GRU) | PDF | Table Boundary Detection<br>Columns and Rows Detection | Relies on heuristics<br>Fails to return text<br>Fails to detect cell structures |
| Zheng *et al.* (2021) [89] | Object Detector<br>+NN | PDF | Table Boundary Detection | Fails to detect cell structures<br>Fails to return text |
| Lee *et al.* (2021) [47] | R-CNN +<br>Graph-Based Network | PDF | Table Boundary Detection | Relies on heuristics<br>Fails to return text |
| Tabula (2013) [6] | Python + Java | PDF | CSV/JSON File | Only readable-PDFs<br>Fails to detect cell structures |
| python-docx (2013) [31] | Python | Word | Plain Sentence | Only works on DocX<br>Fails to detect cell structures |
| PyMuPDF (2015) [38] | Python | PDF | Plain Sentence | Fails to detect table boundaries<br>Fails to detect cell structures |
| Camelot (2018) [57] | Python | PDF | Plain Sentence/csv | Only works for<br>Readable PDFs<br>Fails to detect cell structures |
| Our Algorithm | Rule Sets<br>+ Python | Word/PDF | Text Annotation | Relies on Templates |

The Python package `Tabula` [6] does not identify cell structure correctly when there are no lines separating cells in the table. It also fails in reading any Scanned PDF. The result of the Tabula is a data form that can be converted to a CSV or JSON file. It also fails in reading scanned PDFs. The Python package `PyMuPDF` [38] also fails to read scanned PDFs. The `PyMuPDF` package will read the whole page as an image and utilizes OCR (Optical Character Recognition). This is a very strong package with access to meta information and links for each page into the PDF, which is very helpful when locating the formatting and other details for each element within the PDF, such as text and images. The meta information, such as full position and font information for each text character, is very helpful in terms of finding lists and tables. However, the `PyMuPDF` is not able to detect these elements by itself. `Camelot` [57] is an open-source Python library focused on extracting tables from PDF files [57]. This tool converts tabular data into a pandas DataFrame, and can export in multiple formats, such as JSON, Excel, and HTML. Only tables with distinct cell borders can be parsed by `Camelot`, however. This tool fails when extracting content from scanned documents [57]. The `python-docx` library [31] creates comprehensive document object representations of elements such as paragraphs, tables, headings, and caption objects. This python library only works on DOCX documents, however, and therefore cannot be used directly with PDF content. In our Tally-Vet engine we therefore need to consider readable PDFs and scanned PDFs, as well as DOCX documents. The `python-docx` library also fails to understand the structure of cells and columns in the table. Hence, it is not able to extract flattened sequences from tables that represent logical content from the spanned cells of the table.

For comparison of our proposed algorithm with other methods, we conducted a comprehensive search for the availability and a way to evaluate the functionality of those methods. To compare the performance and functionality of these methods against our algorithm and our system's specific requirements, we therefore studied the documents and papers associated with each method. In Table 3 and Table 4 we listed the characteristics and features of our proposed method with the reviewed characteristics, respectively.

We unfortunately could not directly evaluate the approaches presented in [5, 25, 30, 41, 47, 89] using our cybersecurity corpus documents because their respective implementations were not available online. Therefore, we focus on the available libraries in python that are specialized on extracting tables from PDFs and DOCX documents, in order to compare their results with our proposed algorithm. Figure 17 shows the results from our proposed novel algorithms and the Python libraries available for the corresponding task. The green box indicates an example of a flattened sequence from our algorithm, which connects different parts of the table (orange boxes) to make an informative sentence as an output. As illustrated in Figure. 17 `Camelot`, `PyMuPDF`, and `python-docx` python libraries produce similar results. All these libraries return the text inside each cell of the table as a new line. However, `Tabula` library failed to detect the cell structure of the example table. The cell structure, the logical relationships between cells and their contents was not taken into consideration in any of these libraries.

Research models trained on reference datasets such as [5, 25, 30, 41, 47, 89] often have difficulties coping with the complexity of real world document layouts [15]. Hence, we focused on defining rule sets to contribute novel algorithms for detecting tables and identifying cell structures in the cybersecurity corpus. We did this by examining cybersecurity standards and ICS vendor documents. The defined rule sets facilitate the development of templates that extract flattened sequences from ICS vendor documents.

### 6.2 Claim Detection

Claim detection is a crucial step for our Tally-Vet engine. This step led to a set of sequences that indicate cybersecurity feature claims. These claim sequences are very important for the rest of our vetting engine. If the classifier fails to detect a claim, it will negatively impact all other steps in our framework. Therefore, we focus on building a classifier with highest accuracy.

Adapting a pretrained language model can significantly improves downstream task performance. A pre-trained language model refers to NLP model that was trained unsupervised using a large corpus of text representing a general domain of language. There are several well-established pre-trained language models, including Embeddings from

Table 4. Comparison of our method's features against other available approaches

| | Table Detection | Cell Structure Recognition | Merged Cells/Columns | Scanned PDFs | Identify Lists | Identify Captions | Reading Paragraphs | Plain Sentence | Flattened Sequences |
|---|---|---|---|---|---|---|---|---|---|
| Hao *et al.* (2016) [30] | ✓ | × | ○ | ✓ | × | × | × | × | × |
| Gilani *et al.* (2017) [25] | ✓ | ✓ | ○ | ✓ | × | × | × | × | × |
| Arif *et al.* (2018) [5] | ✓ | ✓ | ○ | ✓ | × | × | × | × | × |
| Khan *et al.* (2019) [41] | ✓ | ○ | ○ | ✓ | × | × | × | × | × |
| Zheng *et al.* (2021) [89] | ✓ | ✓ | ○ | ✓ | × | × | × | × | × |
| Lee *et al.* (2021) [47] | ✓ | ✓ | ○ | ✓ | × | × | × | × | × |
| Tabula (2013) [6] | ✓ | ○ | × | × | ○ | × | × | ✓ | × |
| Python-docx (2013) [31] | ✓ | ✓ | ○ | × | ✓ | ✓ | ✓ | ✓ | × |
| PyMuPDF (2015) [38] | × | × | × | × | × | × | ✓ | ✓ | × |
| Camelot (2018) [57] | ○ | ○ | ○ | × | × | × | × | ✓ | × |
| Our Algorithm | ✓ | ✓ | ○ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

✓ means this feature is supported.
○ means this feature is partially supported.
× means this feature is not supported.

Language Models (ELMo)[63], Universal Language Model with Fine-Tuning (ULMFiT) [33], Bidirectional Encoder Representations from Transformers (BERT) [19], and the Generative Pre-Training (GPT) model [65].

The Table 5 compares the accuracy for all language models obtained using the test set from our cybersecurity NLP dataset. For each model, the highest accuracy is reported based on extensive experimentation we have conducted to determine optimal hyperparameters and the overall architecture [1]. As we show in Table 5, our model CyBERT has highest accuracy, F1 score and Under the ROC curve (AUC) value, which means CyBERT has the best performance when it comes to identifying cybersecurity claim sequences as compared with all other language models we evaluated.

### 6.3 Feature Attribution

The main focus for feature attribution is to perform an NLP sentiment analysis to determine whether the claim sequence does or does not express support of the feature indicated for the device - i.e. shows a positive or negative sentiment. For the purpose of this project we defined a specialized sentiment analysis based on VADER from the `NLTK Toolkit` python library. We optimized the VADER lexicon dictionary weights based on our project's definition. Using our claim sequence database, we evaluated our new model by comparing sentiment and polarity scores obtained from our specialized VADER implementation to those obtained from standard VADER and a transformer-based sentiment analysis model

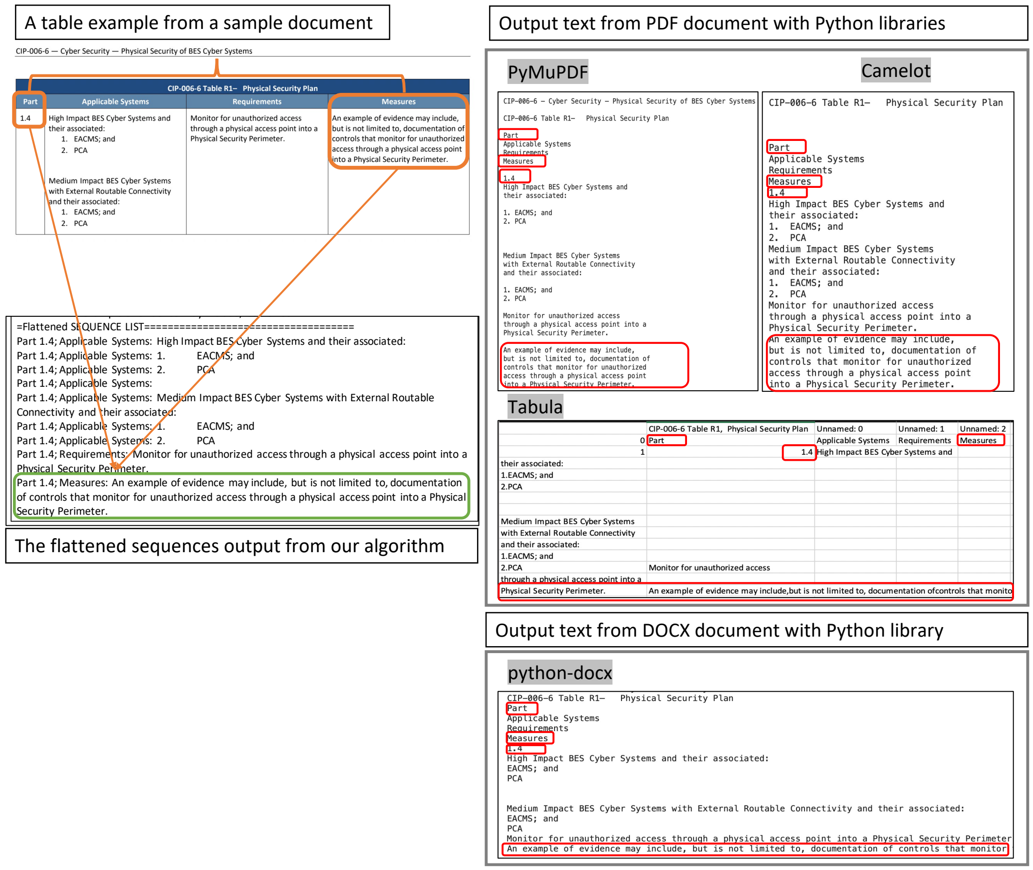


Fig. 17. An output from available libraries in python specialized on extracting tables from PDFs and DOCX document and our proposed novel algorithm

derived from DistilBERT [68]. This variant of DistilBERT intended for sentiment analysis was specifically fine-tuned on the Stanford Sentiment Treebank v2 (SST2) [74].

Sentiment analysis models assign a label to the input sentence based on the detected sentiment, and the polarity scores here show the confidence score for the detected sentiment. Comparing our CYVET sentiment analysis model with standard VADER (Table 6), we observe an improvement on the sentiment assignment for sentences and sequences extracted from our dataset of ICS vendor documents. For a simple sentence such as ”Assign individual user and role-based account authentication and strong passwords”, the DistilBERT Sentiment Analysis and our optimized VADER approach detect the correct intention of the sentence. However, our model’s confidence score is higher than that of the DistilBERT model. For the complex sequence example of ”User Security: Assign individual user and role-based account authentication and strong passwords; use Lightweight Directory Access Protocol (LDAP) for central user authentication.”, DistilBERT fails to assign a correct sentiment to the sentence. The Standard VADER confidence score for both sequence complexity types are in the middle of the range of [0,1], which indicates that it does not have enough

Table 5. Comparison across all tested language models.

| Model | Architecture | Accuracy | Macro Weighted F1 | AUC | Trainable Parameters |
|---|---|---|---|---|---|
| **Our approach (CyBERT)**[1] | 12 Encoder<br>3 Dense | 0.954 | 0.93 | 0.948 | 108,647,026 |
| BERT Classifier | 12 Encoder<br>1 Dense | 0.76 | 0.72 | 0.773 | 109,482,240 |
| GPT2-Small | 12 Decoder<br>1 Dense | 0.9 | 0.87 | 0.908 | 125,444,134 |
| ELMo+NN | 2 Dense | 0.91 | 0.9 | 0.910 | 295,554 |
| ELMo+CNN | 1 Convolution<br>2 Dense | 0.92 | 0.9 | 0.912 | 16,778,242 |
| ELMo+LSTM | 1 LSTM<br>2 Dense | 0.90 | 0.88 | 0.916 | 19,785,410 |
| ELMo+BiLSTM | 1 BiLSTM<br>2 Dense | 0.91 | 0.89 | 0.897 | 15,854,274 |
| ULMFiT | 3 AWD-LSTM | 0.91 | 0.91 | 0.902 | 62,652 |

confidence in the assigned sentiment label. Furthermore, the standard VADER assigned sentiment for both examples are incorrect.

## 7 DATASET AVAILABILITY

At the conclusion of our research project the authors are planning to make a curated dataset publicly available from our repository containing over 12000 product documents from ICS websites, including 2844 manuals, 7832 brochures and 666 catalogs for ICS products. Curated from these documents, our ICS sequence database currently contains over two million sequences.

## 8 CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel framework for a semi-automated vetting system (CYVET) for ICS devices. CYVET's goal is to provide its users with detailed insights into the cybersecurity features claimed by device vendors, and the impact those features may have on their cybersecurity posture. For that purpose, CYVET audits the devices of interest using two primary components: Tally-Vet and Test-Vet. In this paper we are detailing the framework and toolset underpinning Tally-Vet, which is CYVET's component that processes vendor documents about the devices of interest to extract feature claims and vet those against customer requirements.

Tally-Vet is comprised of several complex processing steps, all shown in this paper including sample results. It first extracts sequences from ICS documents located in CYVET's document repository. Then, NLP techniques are used to search for features of interest within all extracted sequences. Those sequences were then processed by our claim classifier to detect sequences that represent claims. Those claims are analyzed with our sentiment analysis algorithm

Table 6. Comparison of sentiment analysis using example sequences

**Example Sentence (1):**
”Assign individual user and role-based account authentication and strong passwords”

| Model | Sentiment | Confidence Score |
|---|---|---|
| Our Specialized VADER | **Positive** | **0.8834** |
| Standard VADER | Neutral | 0.5106 |
| DistilBERT Sentiment Analysis | Positive | 0.802 |

**Example Sentence (2):**
”Assign individual user and role-based account authentication and strong passwords”

| Model | Sentiment | Confidence Score |
|---|---|---|
| Our Specialized VADER | **Positive** | **0.8834** |
| Standard VADER | Neutral | 0.5106 |
| DistilBERT Sentiment Analysis | Positive | 0.802 |

**Example Sentence (3):**
”The RTAC also supports central authentication through your existing LDAP server.”

| Model | Sentiment | Confidence Score |
|---|---|---|
| Our Specialized VADER | **Positive** | **0.586** |
| Standard VADER | Neutral | 0.737 |
| DistilBERT Sentiment Analysis | Negative | 0.939 |

**Example Sentence (4):**
”User Security: Assign individual user and role-based account authentication and strong passwords; use Lightweight Directory Access Protocol (LDAP) for central user authentication.”

| Model | Sentiment | Confidence Score |
|---|---|---|
| Our Specialized VADER | **Positive** | **0.979** |
| Standard VADER | Neutral | 0.690 |
| DistilBERT Sentiment Analysis | Negative | 0.99 |

to identify supported or unsupported features within documents. The feature support processing finds any possible conflicts for feature claims in all documents for each device. The final check in the Tally-Vet pipeline is the customer requirements level check for each feature. Devices only pass the feature support check if all of their required features are claimed as supported by the vendor. The Tally-Vet pipeline is able to verify compatibility between device feature claims and customer requirements.

In this paper, we demonstrated that this framework is novel and able to gather and parse all components required for the Tally-Vet pipeline in the CYVET system. Our proposed algorithms can successfully extract structured sequences from ICS device documents, identify claim sequences and their features, evaluate the intention behind our claim sequence detection approach to identify sequences indicating support of a feature or lack thereof, as well as resolve any conflicting feature claims among the set of documents for each device.

Tally-Vet is able to provide a list of hardware tests based on known reported flaws for each feature. This list will be passed to Test-Vet, the other key component of CYVET, to verify these feature claims through conducting actual hardware tests. Our future work focuses on the Test-Vet functionality of CYVET, specifically:

- Discovering known vulnerability reports and automatic attribution to features.
- Automating Test-Vet script generation and selection for hardware tests.
- End-to-End Integration of CYVET's automated vetting system and its report generation functionality.

## 9 ACKNOWLEDGMENTS

This research was funded by the US. Dept of Energy through a subcontract from Oak Ridge National Laboratory, project No. 4000175929 (project CYVET).

## REFERENCES

[1] Kimia Ameri, Michael Hempel, Hamid Sharif, Juan Lopez Jr, and Kalyan Perumalla. 2021. CyBERT: Cybersecurity Claim Classification by Fine-Tuning the BERT Language Model. *Journal of Cybersecurity and Privacy* 1, 4 (2021), 615–637.

[2] Kimia Ameri, Michael Hempel, Hamid Sharif, Juan Lopez Jr, and Kalyan Perumalla. 2021. Smart Semi-Supervised Accumulation of Large Repositories for Industrial Control Systems Device Information. Academic Conferences Limited, ICCWS 2021 16th International Conference on Cyber Warfare and Security, 1.

[3] Kylie L Anglin. 2019. Gather-narrow-extract: A framework for studying local policy variation using web-scraping and natural language processing. *Journal of Research on Educational Effectiveness* 12, 4 (2019), 685–706.

[4] Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063* (2019).

[5] Saman Arif and Faisal Shafait. 2018. Table detection in document images using foreground and background features. In *2018 Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 1–8.

[6] Manuel Aristarán and Mike Tigas. 2013. Introducing Tabula-Features-Source: An OpenNews project. *Retrieved December* 2 (2013), 2014.

[7] Abderrahim Ait Azzi, Houda Bouamor, and Sira Ferradans. 2019. The finsbd-2019 shared task: Sentence boundary detection in pdf noisy text in the financial domain. In *Proceedings of the First Workshop on Financial Technology and Natural Language Processing*. 74–80.

[8] Donna L Baker and Tom Carson. 2008. *Adobe Acrobat 6: The Professional User's Guide.* Apress.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).

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

[11] Ángela Casado-García, César Domínguez, Jónathan Heras, Eloy Mata, and Vico Pascual. 2020. The benefits of close-domain fine-tuning for table detection in document images. In *International Workshop on Document Analysis Systems*. Springer, 199–215.

[12] Jack Chan, Ray Chung, and Jack Huang. 2019. *Python API Development Fundamentals: Develop a full-stack web application with Python and Flask.* Packt Publishing Ltd.

[13] G Naga Chandrika, Somula Ramasubbareddy, K Govinda, and E Swetha. 2020. Web scraping for unstructured data over web. In *Embedded Systems and Artificial Intelligence*. Springer, 853–859.

[14] YuXuan Chen, Jianwei Ding, Dashuang Li, and Zhouguo Chen. 2021. Joint BERT Model based Cybersecurity Named Entity Recognition. In *2021 The 4th International Conference on Software Engineering and Information Management*. 236–242.

[15] Igor Cherepanov, Andrey Mikhailov, Alexey Shigarov, and Viacheslav Paramonov. 2020. On automated workflow for fine-tuning deepneural network models for table detection in document images. In *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*. IEEE, 1130–1133.

[16] Sean B Cleveland, Anagha Jamthe, Smruti Padhy, Joe Stubbs, Michale Packard, Julia Looney, Steve Terry, Richard Cardone, Maytal Dahan, and Gwen A Jacobs. 2020. Tapis API development with Python: best practices in scientific REST API implementation: experience implementing a distributed stream API. In *Practice and Experience in Advanced Research Computing*. 181–187.

[17] Andreiwid Sheffer Corrêa and Pär-Ola Zander. 2017. Unleashing tabular content to open data: A survey on pdf table extraction methods and tools. In *Proceedings of the 18th Annual International Conference on Digital Government Research*. 54–63.

[18] Hervé Déjean and Jean-Luc Meunier. 2006. A system for converting PDF documents into structured XML format. In *International Workshop on Document Analysis Systems*. Springer, 129–140.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[20] Roy Thomas Fielding. 2000. *Architectural styles and the design of network-based software architectures.* University of California, Irvine.

[21] Chen Gao, Xuan Zhang, and Hui Liu. 2021. Data and knowledge-driven named entity recognition for cyber security. *Cybersecurity* 4, 1 (2021), 1–13.

[22] Liangcai Gao, Yilun Huang, Hervé Déjean, Jean-Luc Meunier, Qinqin Yan, Yu Fang, Florian Kleber, and Eva Lang. 2019. Icdar 2019 competition on table detection and recognition (ctdar). In *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 1510–1515.

[23] Liangcai Gao, Xiaohan Yi, Yuan Liao, Zhuoren Jiang, Zuoyu Yan, and Zhi Tang. 2017. A deep learning-based formula detection method for PDF documents. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Vol. 1. IEEE, 553–558.

[24] Piyush Ghasiya and Koji Okamura. 2020. Comparative analysis of Japan and the US cybersecurity related newspaper articles: A content and sentiment analysis approach. In *International Conference on Advanced Information Networking and Applications.* Springer, 431–443.

[25] Azka Gilani, Shah Rukh Qasim, Imran Malik, and Faisal Shafait. 2017. Table detection using deep learning. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, Vol. 1. IEEE, 771–776.

[26] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2012. A methodology for evaluating algorithms for table understanding in PDF documents. In *Proceedings of the 2012 ACM symposium on Document engineering.* 45–48.

[27] Max Göbel, Tamir Hassan, Ermelinda Oro, and Giorgio Orsi. 2013. ICDAR 2013 table competition. In *2013 12th International Conference on Document Analysis and Recognition.* IEEE, 1449–1453.

[28] Babita Gupta, Shwadhin Sharma, and Anitha Chennamaneni. 2016. Twitter sentiment analysis: An examination of cybersecurity attitudes and behavior. *Proceedings of the 2016 Pre-ICIS SIGDSA/IFIP WG8* 3 (2016).

[29] Muhamad Haekal et al. 2016. Token-based authentication using JSON web token on SIKASIR RESTful web service. In *2016 International Conference on Informatics and Computing (ICIC).* IEEE, 175–179.

[30] Leipeng Hao, Liangcai Gao, Xiaohan Yi, and Zhi Tang. 2016. A table detection method for pdf documents based on convolutional neural networks. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS).* IEEE, 287–292.

[31] S Hoffstaetter, M Lee, , J Bochi, and L Kistner. [n. d.]. Python-docx. A Python library for creating and updating Microsoft Word (.docx) files—python-docx 0.8.5 documentation. ([n. d.]). https://python-docx.readthedocs.io/en/latest/.Accessed:July2021

[32] S Hoffstaetter, M Lee, , J Bochi, and L Kistner. 2014. pytesseract. https://pypi.org/project/pytesseract/. (2014). https://pypi.org/project/pytesseract/.Accessed:July2020

[33] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[34] Yilun Huang, Qinqin Yan, Yibo Li, Yifan Chen, Xiong Wang, Liangcai Gao, and Zhi Tang. 2019. A YOLO-based table detection method. In *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 813–818.

[35] Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 8.

[36] M Jones, J Bradles, and N Sakimura. May 2015. RFC 7519: JSON web token (JWT). Tech. rep., Internet Engineering Task Force. (May 2015). http://www.ietf.org/rfc/rfc7515.txt

[37] M Jones, J Padilla, and J Lindsay. [n. d.]. pyJWT: A Python implementation of RFC 7519. https://github.com/jpadilla/ pyjwt. ([n. d.]). https://pypi.python.org/pypi/pyjwt.Accessed:July2021

[38] M Jones, J Padilla, and J Lindsay. 2016. PyMuPDF. https://github.com/pymupdf/PyMuPDF. (2016). https://github.com/pymupdf/PyMuPDF.Accessed:July2021

[39] Heejung Jwa, Dongsuk Oh, Kinam Park, Jang Mook Kang, and Heuiseok Lim. 2019. exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert). *Applied Sciences* 9, 19 (2019), 4062.

[40] Klara Kaleb, Alex Warwick Vesztrocy, Adrian Altenhoff, and Christophe Dessimoz. 2019. Expanding the Orthologous Matrix (OMA) programmatic interfaces: REST API and the OmaDB packages for R and Python. *F1000Research* 8 (2019).

[41] Saqib Ali Khan, Syed Muhammad Daniyal Khalid, Muhammad Ali Shahzad, and Faisal Shafait. 2019. Table structure extraction with bi-directional gated recurrent unit networks. In *2019 International Conference on Document Analysis and Recognition (ICDAR).* IEEE, 1366–1371.

[42] Vivek Khetan, Roshni Ramnani, Mayuresh Anand, Shubhashis Sengupta, and Andrew E Fano. 2020. Causal BERT: Language models for causality detection between events expressed in text. *arXiv preprint arXiv:2012.05453* (2020).

[43] Hannah Kim and Young-Seob Jeong. 2019. Sentiment classification using convolutional neural networks. *Applied Sciences* 9, 11 (2019), 2347.

[44] Jan Kinne and Janna Axenbeck. 2018. Web mining of firm websites: A framework for web scraping and a pilot study for Germany. *ZEW-Centre for European Economic Research Discussion Paper* 18-033 (2018).

[45] Judith L Klavans. 2015. *Cybersecurity-What's Language got to do with it?* Technical Report.

[46] Vlad Krotov and Matthew Tennyson. 2018. Research note: scraping financial data from the web using the R language. *Journal of Emerging Technologies in Accounting* 15, 1 (2018), 169–181.

[47] Eunji Lee, Jaewoo Park, Hyung Il Koo, and Nam Ik Cho. 2021. Deep-learning and graph-based approach to table structure recognition. *Multimedia Tools and Applications* (2021), 1–22.

[48] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2020. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* 36, 4 (2020), 1234–1240.

[49] Jieh-Sheng Lee and Jieh Hsiang. 2019. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124* (2019).

[50] Xiao-Hui Li, Fei Yin, and Cheng-Lin Liu. 2020. Page Segmentation Using Convolutional Neural Network and Graphical Model. In *International Workshop on Document Analysis Systems*. Springer, 231–245.

[51] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

[52] Bing Liu et al. 2010. Sentiment analysis and subjectivity. *Handbook of natural language processing* 2, 2010 (2010), 627–666.

[53] Chao Liu, Xinghua Wu, Min Yu, Gang Li, Jianguo Jiang, Weiqing Huang, and Xiang Lu. 2019. A two-stage model based on BERT for short fake news detection. In *International Conference on Knowledge Science, Engineering and Management*. Springer, 172–183.

[54] Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028* (2002).

[55] Alex Luscombe, Kevin Dick, and Kevin Walby. 2021. Algorithmic thinking in the public interest: navigating technical, legal, and ethical hurdles to web scraping in the social sciences. *Quality & Quantity* (2021), 1–22.

[56] Diana Maynard and Adam Funk. 2011. Automatic detection of political opinions in tweets. In *Extended Semantic Web Conference*. Springer, 88–99.

[57] Vinayak Mehta. 2018. Camelot. https://pypi.org/project/camelot-py/. (2018). https://pypi.org/project/camelot-py/.Accessed:February2022

[58] Prem Melville, Wojciech Gryc, and Richard D Lawrence. 2009. Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1275–1284.

[59] Sangita S Modi and Sudhir B Jagtap. 2018. Multimodal Web Content Mining to Filter Non-learning Sites Using NLP. In *International conference on Computer Networks, Big data and IoT*. Springer, 23–30.

[60] K Mouthami, K Nirmala Devi, and V Murali Bhaskaran. 2013. Sentiment analysis and classification based on textual reviews. In *2013 international conference on Information communication and embedded systems (ICICES)*. IEEE, 271–276.

[61] Anssi Nurminen. 2013. *Algorithmic extraction of data in tables in PDF documents*. Master's thesis.

[62] Kalyan Perumalla, Juan Lopez, Maksudul Alam, Olivera Kotevska, Michael Hempel, and Hamid Sharif. 2020. A Novel Vetting Approach to Cybersecurity Verification in Energy Grid Systems. In *2020 IEEE Kansas Power and Energy Conference (KPEC)*. IEEE, 1–6.

[63] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365* (2018).

[64] Qiao Qian, Minlie Huang, Jinhao Lei, and Xiaoyan Zhu. 2016. Linguistically regularized lstms for sentiment classification. *arXiv preprint arXiv:1611.03949* (2016).

[65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[66] Sachin Raja, Ajoy Mondal, and CV Jawahar. 2020. Table structure recognition using top-down and bottom-up cues. In *European Conference on Computer Vision*. Springer, 70–86.

[67] Stuart Rennie, Mara Buchbinder, Eric Juengst, Lauren Brinkley-Rubinstein, Colleen Blue, and David L Rosen. 2020. Scraping the Web for Public Health Gains: Ethical Considerations from a 'Big Data'Research Project on HIV and Incarceration. *Public Health Ethics* 13, 1 (2020), 111–121.

[68] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[69] Bagus Satria, Ari Kusyanti, and Widhi Yahya. 2018. Implementasi Algoritme Blake2s pada JSON Web Token (JWT) sebagai Algoritme Hashing untuk Mekanisme Autentikasi Layanan REST-API. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN* 2548 (2018), 964X.

[70] Sebastian Schreiber, Stefan Agne, Ivo Wolf, Andreas Dengel, and Sheraz Ahmed. 2017. Deepdesrt: Deep learning for detection and structure recognition of tables in document images. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, Vol. 1. IEEE, 1162–1167.

[71] Alexey Shigarov, Andrey Mikhailov, and Andrey Altaev. 2016. Configurable table structure recognition in untagged pdf documents. In *Proceedings of the 2016 ACM Symposium on Document Engineering*. 119–122.

[72] Han-Sub Shin, Hyuk-Yoon Kwon, and Seung-Jin Ryu. 2020. A new text classification model based on contrastive word embedding for detecting cybersecurity intelligence in twitter. *Electronics* 9, 9 (2020), 1527.

[73] Kai Shu, Amy Sliva, Justin Sampson, and Huan Liu. 2018. Understanding cyber attack behaviors with sentiment information on social media. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer, 377–388.

[74] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.

[75] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification?. In *China National Conference on Chinese Computational Linguistics*. Springer, 194–206.

[76] Mikhail Tikhomirov, N Loukachevitch, Anastasiia Sirotina, and Boris Dobrov. 2020. Using bert and augmentation in named entity recognition for cybersecurity domain. In *International Conference on Applications of Natural Language to Information Systems*. Springer, 16–24.

[77] Tuan Anh Tran, Hong Tai Tran, In Seop Na, Guee Sang Lee, Hyung Jeong Yang, and Soo Hyung Kim. 2016. A mixture model using Random Rotation Bounding Box to detect table region in document image. *Journal of Visual Communication and Image Representation* 39 (2016), 196–208.

[78] Seppe vanden Broucke and Bart Baesens. 2018. *Practical Web Scraping for Data Science: Best Practices and Examples with Python*. Apress.

[79] Noel Varela, Omar Bonerge Pineda Lezama, and Milvio Charris. 2021. Web Scraping and Naïve Bayes Classification for Political Analysis. In *Proceedings of International Conference on Intelligent Computing, Information and Control Systems*. Springer, 1–8.

[80] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. 2011. Confusion matrix-based feature selection. *MAICS* 710 (2011), 120–127.

[81] Inna Vogel and Meghana Meghana. 2020. Detecting Fake News Spreaders on Twitter from a Multilingual Perspective. In *2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 599–606.

[82] Shengye Wan, Yue Li, and Kun Sun. 2019. PathMarker: protecting web contents against inside crawlers. *Cybersecurity* 2, 1 (2019), 1–17.

[83] Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, and Long Wang. 2018. An LSTM approach to short text sentiment classification with word embeddings. In *Proceedings of the 30th conference on computational linguistics and speech processing (ROCLING 2018)*. 214–223.

[84] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*. 347–354.

[85] Bo Xie, Guowei Shen, Chun Guo, and Yunhe Cui. 2021. The Named Entity Recognition of Chinese Cybersecurity Using an Active Learning Strategy. *Wireless Communications and Mobile Computing* 2021 (2021).

[86] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1192–1200.

[87] Burcu Yildiz, Katharina Kaiser, and Silvia Miksch. 2005. pdf2table: A method to extract table information from pdf files. In *IICAI*. Citeseer, 1773–1785.

[88] Jiao Yin, MingJian Tang, Jinli Cao, and Hua Wang. 2020. Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. *Knowledge-Based Systems* 210 (2020), 106529.

[89] Xinyi Zheng, Douglas Burdick, Lucian Popa, Xu Zhong, and Nancy Xin Ru Wang. 2021. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 697–706.

[90] Shieheng Zhou, Jingju Liu, Xiaofeng Zhong, and Wendian Zhao. 2021. Named Entity Recognition Using BERT with Whole World Masking in Cybersecurity Domain. In *2021 IEEE 6th International Conference on Big Data Analytics (ICBDA)*. IEEE, 316–320.