electronics

MDPI

Article

An Email Cyber Threat Intelligence Method Using Domain Ontology and Machine Learning

Algimantas Venčkauskas [1], Jevgenijus Toldinas [1,*], Nerijus Morkevičius [1] and Filippo Sanfilippo [2]

[1] Department of Computer Science, Kaunas University of Technology, 44249 Kaunas, Lithuania; algimantas.venckauskas@ktu.lt (A.V.); nerijus.morkevicius@ktu.lt (N.M.)
[2] Department of Engineering Sciences, University of Agder (UiA), 4879 Grimstad, Norway; filippo.sanfilippo@uia.no
* Correspondence: eugenijus.toldinas@ktu.lt

Abstract: Email is an excellent technique for connecting users at low cost. Spam emails pose the risk of collecting a user's personal information by fooling them into clicking on a link or engaging in other fraudulent activities. Furthermore, when a spam message is delivered, the user may read the entire message before deciding it is spam and deleting it. Most approaches to email classification proposed by other authors use natural language processing (NLP) methods to analyze the content of email messages. One of the biggest shortcomings of NLP-based methods is their dependence on the language in which a message is written. To construct an effective email cyber threat intelligence (CTI) sharing framework, the privacy of a message's content must be preserved. This article proposes a novel domain-specific ontology and method for emails that require only the metadata of email messages to be shared to preserve their privacy, making them applicable to solutions for sharing email CTI. To preserve privacy, a new semantic parser was developed for the proposed email domain-specific ontology to populate email metadata and create a dataset. Machine learning algorithms were examined, and experiments were conducted to identify and classify spam messages using the newly created dataset. Feature-ranking algorithms, chi-squared, ANOVA (analysis of variance), and Kruskal–Wallis tests were used. In all experiments, the kernel naïve Bayes model demonstrated acceptable results. The highest accuracy of 92.28% and an F1 score of 95.92% for recognizing spam email messages were obtained using the proposed domain-specific ontology, the newly developed semantic parser, and the created metadata dataset.

Keywords: cyber threat intelligence; email; domain ontology; machine learning

Citation: Venčkauskas, A.; Toldinas, J.; Morkevičius, N.; Sanfilippo, F. An Email Cyber Threat Intelligence Method Using Domain Ontology and Machine Learning. *Electronics* **2024**, *13*, 2716. https://doi.org/10.3390/electronics13142716

Academic Editors: Sicong Shao and Jielun Zhang

Received: 3 June 2024
Revised: 24 June 2024
Accepted: 2 July 2024
Published: 11 July 2024

1. Introduction

When an organization shares and consumes external information while engaging in cyber threat intelligence (CTI) sharing, it may adopt precise, actionable, and effective procedures that greatly reduce security risks. Although it is seen as a competitive disadvantage, which can be changed with the right mix of incentives and public policies, everyone benefits from a collective commitment to open sharing [1]. Email is an effective type of technology that connects users at minimal cost; however, communications pose a threat to users of both the internet and email. Email spam is not only annoying, but it also poses a security risk since it contains phishing links, which have the potential to steal a user's personal information by deceiving them into clicking on links or taking part in other fraudulent activities. Furthermore, when a spam message is delivered, the user may read the full message before recognizing it as spam and deleting it. Problems and unresolved issues related to the classification of emails have been researched and discussed. For example, 11 future directions were highlighted in [2]: (1) the use of ontology and semantic webs, (2) real-time learning (stream learning) of the email classifier, (3) dynamic updating of the feature space, (4) deep learning, (5) classification of the emails using hierarchical

classification, (6) reducing processing and classification times using hardware accelerator technology, (7) dealing with the phenomenon of concept drift, (8) reducing the false positive rate, (9) image- and text-based classification, (10) language-based barriers, and (11) the barriers and biases of datasets.

Publicly accessible formats for unstructured CTI reports include news bulletins, blogs written by security professionals, and white papers. It is paradoxical that security analysts are not fully capable of quickly identifying and attributing cyber threats, even with the abundance of CTI reports available and the similarities between the attack techniques of cyber threat actors [3]. The reason for this is the absence of widely accepted and followed CTI standards. This leads to a situation in which attack patterns are reported in an unstructured textual format, which is very difficult to interpret by machines. On the other hand, there is a trend toward the greater exchange of data and expertise among enterprises to support the mitigation of incidents and vulnerability, facilitating threat management. The sharing of CTI addresses these issues of interoperability and communication in a CTI system and handles the exchange of CTI and the protection of the data's privacy [4].

Several standards, such as CybOX, STIX, TAXII, and MISP, have been created to facilitate the sharing of CTI [5]. Most of these standards are based on ontologies that use formal semantics to derive meaning from data in a way that computers can interpret by defining concepts according to how they relate to other concepts [6]. These ontologies clearly define classes and entities, as well as their properties and relationships, in a specific domain.

The behaviors of threat actors are described in the following two MITRE taxonomies: "common attack pattern enumeration and classification" (CAPEC) and "adversarial tactics, techniques, and common knowledge" (ATT&CK). CAPEC is "a publicly available catalog of common attack patterns that helps users understand how adversaries exploit weaknesses in applications and other cyber-enabled capabilities" [7]. Based on investigations of actual events, ATT&CK is a publicly accessible knowledge base of adversaries' tactics and procedures [8,9].

The sharing of CTI on several platforms requires the use of a common and well-defined language to characterize the information being communicated. The Structured Threat Information eXpression (STIX) language was created to help achieve this goal [10]. The Trusted Automated Exchange of Intelligence Information (TAXII) is a protocol for exchanging CTI represented in STIX over HTTPS [11].

The Unified Cybersecurity Ontology (UCO) [12] incorporates and integrates various schemas of data from different cybersecurity domains and standards for the sharing and exchange of CTI information. The authors claimed that UCO provides comprehensive coverage and is mapped to a number of other existing cybersecurity ontologies and concepts.

The authors of [13] proposed a novel ontology-based cybersecurity framework by extending the well-known cybersecurity ontologies to better correspond to the challenges in the area of applications, systems, and services based on the use of artificial intelligence.

The Cybersecurity Operations Center Ontology for Analysis (CoCoa) approach [14] is a processing ontology for CSOC analyses that aligns with the NIST cybersecurity framework. The authors of the ontology claimed that this processing ontology is fundamentally different from log collection approaches in the way that it helps in the CSOC analytical process.

The authors of [15] presented an ontology-based cybersecurity framework using knowledge reasoning for the IoT. The method used two approaches to help enhance cybersecurity in the stages of design and operation.

A framework for identifying and understanding adversaries' tactics and techniques was presented in [16]. The authors proposed an ontology and an automatic method for extracting information that enable the integration of information from CTI reports. The ontology is represented in the Web Ontology Language (OWL), which is accessible through the SPARQL query language. The proposed framework can effectively infer adversaries' information and help security analysts recognize their tactics and techniques.

The content of spam can differ significantly across the various domains that spammers target. For example, advertisements for fake therapies or medications are typically the focus of spam operations in the health domain, while advertisements for questionable financial services and goods can be found in the finance domain; therefore, when classifying spam, the domain-specific features of spam may be more effective than general-purpose ones [17].

Classification of spam emails is a dynamic and difficult problem, and various approaches can be used for classifying emails, such as unsupervised machine learning (ML), supervised ML, and ensemble ML [18]. Even though the most recent binary classifiers perform well, it is important to note that the emails used to train the machine learning models are from publicly accessible datasets that date back to the early 2000s [19], including the Ling-Spam (2000), PU3 (2003), and Enron-Spam (2006) datasets. However, it is crucial to emphasize that spam email is constantly evolving due to the subjects' evolution over time and the methods used by spammers to avoid detection by spam filters. As a result, datasets are bound to change, and the most recent studies on spam emails are training their models without considering the latest tricks used by spammers [20].

The main contributions of this study are as follows:

- A novel, domain-specific ontology for emails is presented that focuses only on email message metadata, including technical fields that indicate the email message's path from the initial sender to the final recipient.
- A new semantic parser for preserving the privacy of data in email messages was developed that uses a semantic representation of the email message's metadata to populate the proposed ontology and create a dataset.
- It is possible to use a semantic representation of an email message's metadata to classify encrypted email messages without knowledge of the decryption key. The email encryption standard S/MIME [21] cryptographically protects only the body of the messages, while the header fields remain in plaintext as the SMTP servers need to deliver messages correctly.
- Empirical quantification of the proposed method using machine learning for spam classification enabled us to improve the accuracy of classifying emails; in particular, an accuracy of 92.28% and an F1 score of 95.92% were obtained.

The remainder of this article is organized as follows: Section 2 discusses the related works. Section 3 presents and explains the methodology. Section 4 presents and discusses the experimental results. Finally, Section 5 presents the discussion, and Section 6 presents the conclusion.

2. Related Work

In this section, an overview of the related work is provided. The methodical, logical, and organized process known as the Intelligence Cycle turns data and information into intelligence, and its four steps include the following: direction, collection, processing, and dissemination [22]. Cyber threat intelligence (CTI) helps companies, governments, and individual users make faster, more educated, and data-driven security-related decisions and changes their behavior against threats from a reactive to a proactive approach [23].

Using fast text (FT) in combination with the attentional hybrid neural networks (HANs) model architecture for the task of detecting email spam was proposed in [24]. The suggested approach uses fast text as a word-embedding model in the first layer, resulting in more accurate embeddings that consider the word structure. The suggested strategy leverages convolutional neural networks (CNNs) in the second layer to extract meaningful, abstract, and generalizable features using attention approaches at the sentence and word levels. Five different datasets were utilized to assess the proposed model and to evaluate the outcomes.

A classification framework for spam that allows for the classification of encrypted emails was proposed in [25]. The proposed model was built around a neural network with a quadratic network component and a multilayered perceptron network. The quadratic

network's design works with an existing encryption technique with a quadratic function. To protect the privacy of email content, the following two solutions for categorizing spam were suggested: homomorphic encryption (HE) and functional encryption (FE). The classifiers for evaluation can predict the labels of encrypted emails.

In [26], an effective ontology-based spam filtering technique was proposed. The Spam Base dataset, Weka, and Jena were utilized in the creation of the ontology, which was specifically made to filter spam. Wang et al. [27] explored ways to encourage users to report phishing emails. To overcome the problem, a tripartite evolutionary game model was developed that involved email security providers, email users, and attackers. The process of evolution towards the desired stable approach was determined, guiding email security providers to provide appropriate incentive mechanisms. The results of the experiment indicated that the proposed model would be both effective and feasible.

Sathya et al. [28] aimed to optimize the detection of crime on social media platforms through a multiagent ontology-based strategy. The goal was to improve the efficiency and accuracy of detecting crimes through semantic analysis, natural language processing, and multiagent optimization. The authors presented an ontology-based technique that used semantic analysis and natural language processing to detect suspicious patterns and structures in social media data. The proposed approach improved the detection of illegal activity and suspicious conduct on social media.

The performances of two ensemble models based on the random forest and extreme gradient boost ensemble algorithms were evaluated and compared in [29], demonstrating the effectiveness of the proposed models for detecting and classifying spam emails. The grid-search-based cross-validation technique was then used to optimize the ensemble models to find the ideal values for the hyperparameters by searching the hyperparameter space.

A domain-specific ontology called DSpamOnto that tagged a particular domain to identify social spammers on microblogging sites was proposed in [30]. On the basis of domain-specific behaviors, such as uploading inconsistent or irrelevant content and presenting false information, DSpamOnto can detect social spammers. The usefulness of the suggested approach for identifying social spammers was confirmed and validated by benchmarking it against established machine learning models using a range of assessment measures.

From the summary presented in Table 1, we can make the following assumptions:

- The email classification approaches proposed by the other authors use natural language processing methods to analyze the content of the messages. On the other hand, if we want to construct an effective email CTI sharing framework, then the privacy of the messages' content must be preserved.
- One of the biggest shortcomings of natural-language-processing-based methods is their dependence on the written language of the message.
- The ontologies proposed by other authors [26,30] tried to semantically express the content of the messages to help in classification.
- As stated in [30], to differentiate spam from authentic information, ML techniques examine patterns and attributes in the data, and ontologies formally represent domain knowledge to generate rules for detecting social spammers.

These assumptions guided the construction of the proposed email CTI sharing framework using a domain-specific ontology of email metadata and a semantic parser preserving the privacy of the email message data.

Table 1. List of state-of-the-art research papers on CTI sharing.

Reference	Dataset	Advantages	Disadvantages
Zavrak et al. [24]	TREC 2007, GenSpam, SA, LS, Enron (EN)	A combination of CNN, gated recurrent units, and attention mechanisms Cross-dataset experiments	Does not preserve the privacy of email messages Only for messages in English language
Nguyen et al. [25]	TREC07p, CEAS08-1, ENRON	Preserves email messages' privacy using HE and FE Predicts the label of an encrypted email	Challenges with the distribution of keys and setting up the server
Kiamarzpour et al. [26]	SpamBase	Filters spam by using the words' ontology	Does not preserve the privacy of email messages A legacy dataset was used
Wang et al. [27]	–	The approach was based on the tripartite evolutionary game model	The custom dataset of email networks was collected by the North University of China and is not publicly available
Sathya et al. [28]	ImageNet	A framework that leverages ontology-based techniques, multiagent optimization algorithms, and semantic analysis	Used to classify images from social media platforms Utilized a pretrained CNN model and the ImageNet dataset
Omotehinwa et al. [29]	Enron	Proposed fine-tuned spam detection models based on the random forest (RF) and extreme gradient boost (XGBoost) algorithms	Does not preserve the privacy of email messages
Al-Hassan et al. [30]	MIB dataset	Proposed a domain-specific ontology for detecting social spammers on microblogging platforms that target a certain domain	Detects social spammers on microblogging platforms only

3. Materials and Methods

In this section, we introduce the newly developed email CTI sharing framework, which employs a domain-specific ontology, as presented in Figure 1.

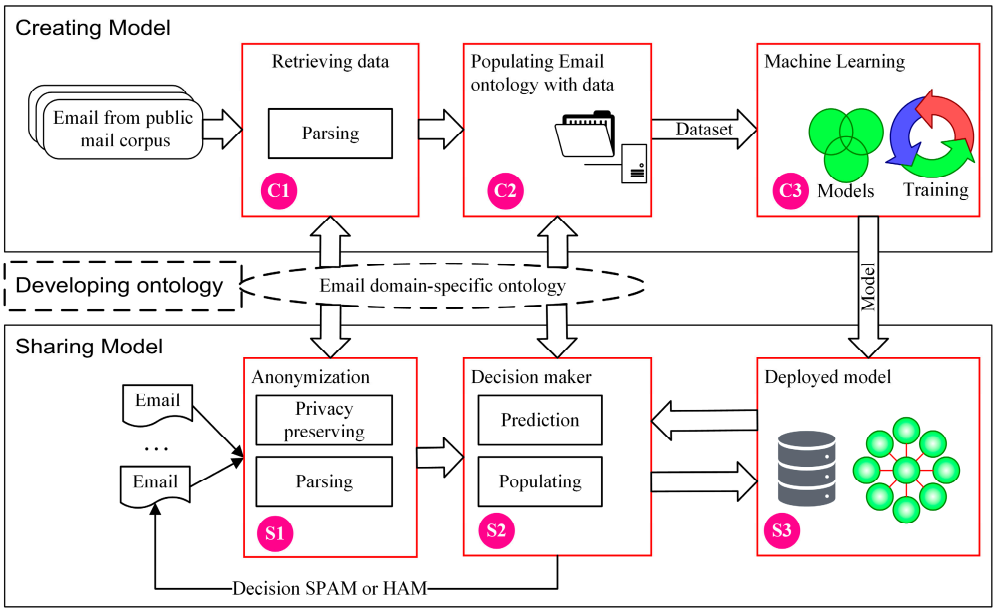


Figure 1. The proposed email CTI sharing framework using a domain-specific ontology.

The process of developing the proposed email CTI sharing framework with a domain-specific ontology consisted of the following three main stages: development of the email domain-specific ontology, creation of the ML model using the domain-specific ontology, and sharing the ML model for making predictions using the ontology. The stage of creating the ML model using the domain-specific ontology for emails depicted in Figure 1 consisted of the following three tasks:

- C1—developing a semantic parser module using emails from the public mail corpus that parses only the messages' metadata;
- C2—populating the domain-specific ontology developed from the email with the data obtained from the parsing module and preparing the labeled dataset for ML;
- C3—using various ML algorithms to train the models and performing evaluations to select the best model.

The model-sharing stage of the proposed development process used the ML model based on the newly developed domain-specific ontology using the messages' metadata. This stage, as depicted in Figure 1, consisted of the following three tasks:

- S1—to preserve privacy, the parser module parses only the metadata from the users' email;
- S2—populating the domain-specific ontology with data obtained from the parsing module and transmitting this to the trained ML model for making predictions;
- S3—the deployed ML model evaluates the message's metadata according to the domain-specific ontology and returns predictions developed for the next decision-making module.

The decision obtained via the shared model evaluates emails as spam or ham.

3.1. Domain-Specific Email Ontology

The first stage of the proposed process is based on the development of a domain-specific email ontology. The domain of the proposed ontology is the metadata of email messages according to the internet message format (IMF), a text message syntax used in email communications that is exchanged between computer users [31]. This ontology includes concepts describing the email addresses of the receivers, senders, and forwarders; information on all the SMTP servers involved—including the mail transfer agents (MTAs) acting as the originators, intermediaries (relays), and deliverers, as well as selected metadata of the content of the message, including URL links to external resources and the type of attached document. The email ontology was developed according to the following assumptions: it should not include potentially confidential data, it should be small and easy to populate using the automatic message parser, it should provide an easily understood basis for further analysis by security experts using standard inference and query tools, and it should include enough information to allow security experts to find potentially malicious SMTP servers, email provider domains, open-relay MTAs, and so on.

The full specifications of the proposed domain-specific email ontology are publicly available in RDF format [31]. The ontology consists of 22 classes, 59 object properties, and 25 data properties. It was developed using Protégé, an open-source solution that provides a user-friendly interface for creating, querying, and editing ontologies. Two parts of the ontology corresponding to the data of the email sender and receiver and the message transfer path are provided in Figures 2 and 3.

Some of the most important object relationships among the concepts of the email ontology are listed in Table 2.

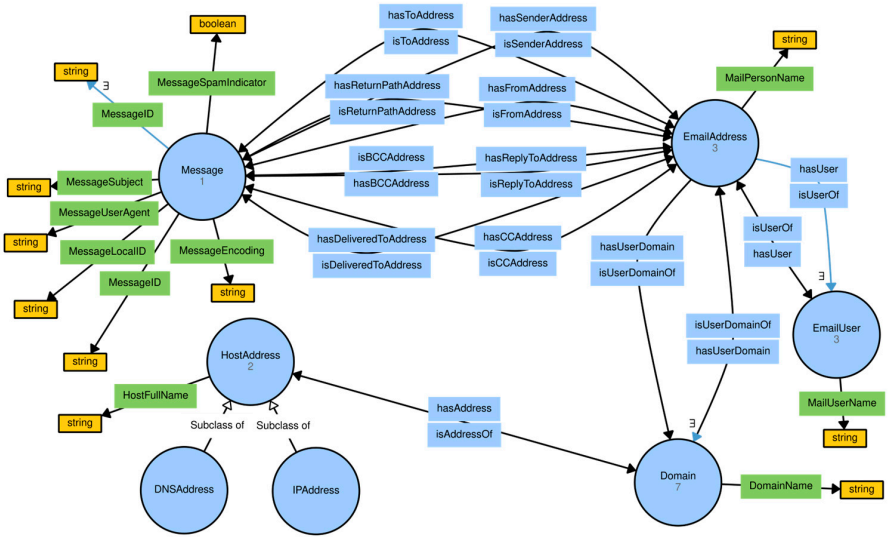


Figure 2. Fragment of the domain-specific ontology for emails, including information related to the sender and receiver.

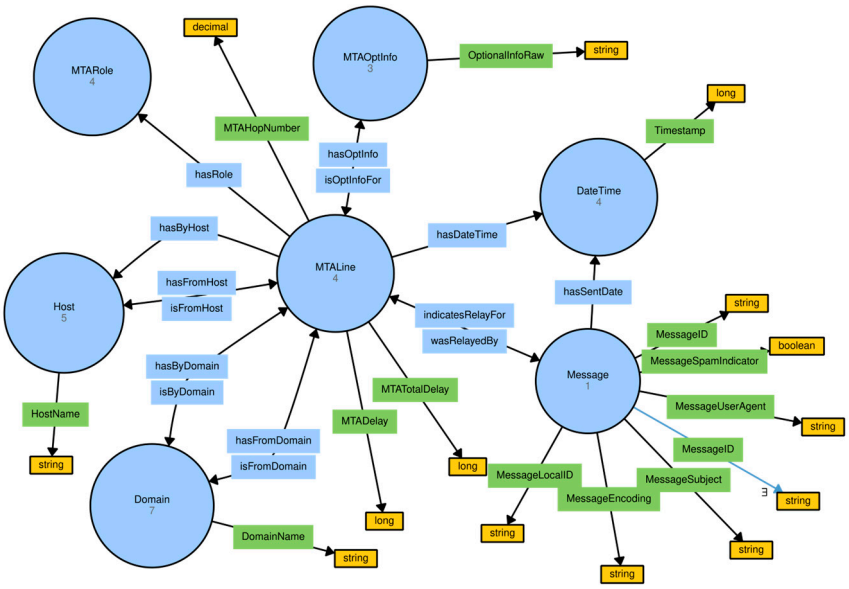


Figure 3. Fragment of the domain-specific ontology for emails, including information on the path traveled by the message.

Table 2. List of the main relationships of the objects in the email ontology.

Subject	Forward Relationship	Reverse Relationship	Object
Message	hasToAddress hasFromAddress hasSenderAddress hasReplyToAddress	isToAddress isFromAddress isSenderAddress isReplyToAddress	EmailAddress
Message	hasSentTime		DateTime
EmailAddress	hasUserDomain	isUserDomainOf	Domain
EmailAddress	hasUser	isUserOf	EmailUser
Message	wasRelayedBy	indicatesRelayFor	MTALine
MTALine	hasRole		Role
MTALine	hasDateTime		DateTime

Table 2. *Cont.*

Subject	Forward Relationship	Reverse Relationship	Object
MTALine	hasByHost hasFromHost	isFromHost isByHost	Host
MTALine	hasByDomain hasFromDomain	isByDomain isFromDomain	Domain
Message	hasURL	isURLIn	URL
URL	hasHost	isHostOf	HostAddress
URL	hasResource	isResourceOf	URLResource
Message	hasAttachment	isAttachmentIn	Attachment
Attachment	hasFileName	isFileNameOf	AttachedFileName
Attachment	hasFileType	isFileTypeOf	AttachedFileType

3.2. Applying the Domain-Specific Ontology to the Collected Email Messages

The easiest and most efficient way to semantically interpret an email message under consideration is to use a semantic parser. A semantic parser is software that tries to convert an email message into an ontological representation using semantic annotations. A domain-specific reference ontology provides the language that is used in an annotation to explicitly explain the meaning of all resources. All annotations, metadata, and other important data are represented using the RDF triplet notation in subject–predicate–object format (see Table 2, and Figures 2 and 3). Each message is represented by a collection of predicates that denote the connections between subjects and objects. The parsing tool produces the triplets according to the domain-specific ontology's schema. The simplified pseudo-code of the semantic parser is presented below (Algorithm 1).

The semantic representation of the metadata of the email message has the following advantages, compared with using and sharing the full source of the message:

1. None of the sensitive parts in terms of privacy (such as the body or the content of the attachments) of the message are included while asserting instances and relationships of the entities forming the semantic representation of the email message. As the results presented in this study showed, in most cases, it is sufficient to use only the semantically enriched metadata of the message to successfully filter unsolicited messages from the good ones.
2. The data represented in RDF format could be very easily processed using an OWL-based reasoner. On the basis of the relationships between the instances, the reasoner can infer the new properties, implicit relationships with other instances, and membership of subclasses. Furthermore, the reasoner's ability to correlate instances more precisely than stated facts is made possible by the taxonomy of the classes' hierarchical structure, ontological relationships, and constraints.
3. The asserted and inferred data are stored in the form of triplets in specialized structures that have the capacity for ad hoc data queries. Well-known querying languages, such as SPARQL (SPARQL Protocol and RDF Query Language), could be used to further enrich the semantic representation of the email message's metadata. The foundation of SPARQL queries is the "triplet pattern" matching mechanism, which follows the triplet configuration of the RDF statements and offers an efficient mechanism for matching triplets.
4. It is possible to use the semantic representation of email message metadata for the classification of encrypted email messages without knowledge of the decryption key. The S/MIME email encryption standard cryptographically protects only the body of the messages, whereas the header fields remain in plaintext because the SMTP servers need to deliver the message correctly. In such a case, only the metadata of the attached files and links to external resources cannot be extracted and populated.

That is, classification is possible on the SMTP servers of the service provider without compromising the confidentiality of the final user's data.

Algorithm 1. The pseudo-code of the semantic parser.

Input:
Email message *msg*, spam indicator *sp*, email ontology *O*.
Parsing:

1. Append *O* with a new individual *m* of class *Message* representing *msg*.
2. Analyze the header of *msg* and append *O* with the data properties of *MessageSpamIndicator*, *MessageUserAgent*, *MessageID*, *MessageEncoding*, *MessageSubject*, etc., connected to *m*.
3. Analyze the header of *msg* and append *O* with the individual *ci* of the class *ContentInfo* connected to *m*. Append *O* with content-info-related properties and connect these properties to *ci*.
4. Analyze the header of *msg* and append *O* with the individual *dt* of the class *DateTime* connected to *m* using the object relationship *hasSentDate*.
5. Analyze the header of *msg* and append *O* with the individuals *fr*, *to*, *cc*, *bcc*, *snd*, *rto*, and *dto* of the class *EmailAddress* connected to *m* using the corresponding relationships.
6. **With each** individual *ind* from the set { *fr*, *to*, *cc*, *bcc*, *snd*, *rto*, *dto*} do:
    a. Append *O* with the individual from the class *EmailPerson* connected to *ind*.
    b. **If** (domain from email address is already in *O*), select the corresponding individual *d*.
    c. **Else** append *O* with the individual *d* from the class *Domain*.
    d. Connect *d* to *ind*.
7. Analyze the header of *msg* and append *O* with the individuals *mta1, mta2, . . .* of the class *MTALine* connected to *m*.
8. **With each** individual *mta* from the set { *mta1, mta2, . . .*}, do:
    a. Analyze the corresponding line in *msg* and append *O* with individuals from the classes *MTARole* and *MTAInfo* connected to *mta*.
    b. Analyze the corresponding line in *msg* and determine the *fromDomain* and *byDomain* of the MTA line.
    c. **With each** *dom* from the set {*fromDomain*, *byDomain*}, do:
        i. **If** (*dom* is already in *O*), select the corresponding individual *d*.
        ii. **Else** append *O* with the individual *d* from the class *Domain*.
        iii. Connect *d* to *mta* using the corresponding relationship.
9. Analyze the body of *msg* and append *O* with the individuals *ln1, ln2, . . .* of the class *Link* connected to *m*.
10. **With each** individual *ln* from set { *ln1, ln2, . . .*}, do:
    a. Analyze *msg* and append *O* with the individuals *pr*, *fl*, and *add* from the corresponding classes *URLProtocol*, *URLFile*, and *Address* connected to *ln*.
    b. **If** (domain in *add* is already in *O*), select the corresponding individual *d*.
    c. **Else** append *O* with the individual *d* from the class *Domain*.
    d. Connect *d* to *add* using the corresponding relationship.
11. Analyze the body of *msg* and append *O* with the individuals *at1, at2, . . .* of the class *Attachment* connected to *m*.
12. With each individual *at* from the set { *at1, at2, . . .*}, do:
    a. Analyze *msg* and append *O* with the individual *af* from the class *AttachedFile* connected to *at*.
    b. **If** (attachment file type is already in *O*), select the corresponding individual *ft*.
    c. **Else** append *O* with the individual *ft* from the class *AttachedFileType*.
    d. Connect *ft* to *at* using the *hasFileType* relationship.

Output:
Email-specific ontology *O* appended with new individuals and relationships representing email message *msg*.
End the semantic parser algorithm.

3.3. Creation of the ML Model Using the Domain-Specific Ontology

The second stage of the proposed process is based on the creation of ML models using the domain-specific ontology developed for email messages using metadata. This stage is depicted in Figure 1 (see Tasks C1, C2, and C3).

A total of 52 features (listed in Table 3) were chosen to prepare the dataset for ML. These subjects were divided into two main classes. The first class contains the features that could be directly retrieved from the corresponding triplets of the email ontology (e.g., SENT_TS, FROM_U, TO_U). Some additional preparation of these features was performed since the ontological representation of a message has very wide capabilities, which are not feasible when expressing one message with one line in a CSV file. For example, only the first email address from all 'TO' addresses was used as a feature. We also limited the number of attachments to three, and intermediary SMTP servers were not included, leaving only information on the originator and delivery servers as the features used for ML. The second class included derivative features, which were calculated using the full message's semantic representation and used as additional features. These included DMTA_TT, DMTA_NR, ATT_COUNT, URL_CNT, and so on.

Table 3. List of the email messages' header fields [31–35] for populating the email ontology with data.

Feature	Description	Feature	Description
M_ID	Globally unique message identifier assigned by the originator MTA	OMTA_BY_H	Host name extracted from the *By-domain* part of the *Received* field at the originator SMTP server
SUBJECT	Human-visible subject of the message	OMTA_BY_D	Host domain extracted from the *By-domain* part of the *Received* field at the originator SMTP server
SENT_TS	Sent timestamp assigned by the originator server	OMTA_TS	Timestamp at the originator SMTP server
CONT_TYPE	*Content* part of the *Content-Type* header field	OMTA_DELAY	Delay of the message (in ms) at the originator SMTP server
CONT_SUBTYPE	*Subtype* part of the *Content-Type* header field	OMTA_TT	Total travel time of the message (in ms)
CONT_PARAM	*Parameter* part of the *Content-Type* header field	DMTA_NR	Delivery SMTP server's hop number
ENC	Encoding of the email body	DMTA_FROM_H	Host name extracted from the *From-domain* part of the *Received* field at the delivery SMTP's server
USER_AGENT	*User-Agent* header field provided by the sender	DMTA_FROM_D	Host domain extracted from the *From-domain* part of the *Received* field at the delivery SMTP server
FROM_P	*Display-name* part of the *From* email address	DMTA_BY_H	Host name extracted from the *By-domain* part of the *Received* field at the delivery SMTP server
FROM_U	*Local-part* of the *From* email address	DMTA_BY_D	Host domain extracted from the *By-domain* part of the *Received* field at the originator SMTP server
FROM_D	*Domain* part of the *From* email address	DMTA_TS	Timestamp at the delivery SMTP server
TO_P	*Display-name* part of the first *To* email address	DMTA_DELAY	Delay of the message (in ms) at the originator SMTP server
TO_U	*Local-part* of the first *To* email address	DMTA_TT	Total travel time of the message (in ms)
TO_D	*Domain* part of the first *To* email address	ATT_COUNT	Total count of attachments

Table 3. *Cont.*

Feature	Description	Feature	Description
CC_P	*Display-name* of the first *CC* email address	ATT_FIRSTNAME	Name of the first attached file
CC_U	*Local-part* of the first *CC* email address	ATT_FIRSTEXT	Extension of the first attached file
CC_D	*Domain* part of the first *CC* email address	URL_CNT	Total count of unique URLs in the message's body
REPLY_P	*Display-name* part of the *Reply-to* email address	URL1_PROT	Protocol of the first URL in the message's body
REPLY_U	*Local-part* of the *Reply-to* email address	URL1_HOST	Host of the first URL in the message's body
REPLY_D	*Domain* part of the *Reply-to* email address	URL1_FILE	Resource name of the first URL in the message's body
SENDER_P	*Display-name* part of the *Sender* email address	URL2_PROT	Protocol of the second URL in the message's body
SENDER_U	*Local-part* of the *Sender* email address	URL2_HOST	Host of the first URL in the message's body
SENDER_D	*Domain* part of the *Sender* email address	URL2_FILE	Resource name of the second URL in the message's body
OMTA_NR	Originator SMTP server's hop number (usually 1)	URL3_PROT	Protocol of the third URL in the message's body
OMTA_FROM_H	Host name extracted from the *From-domain* part of the *Received* [35] field at the originator SMTP server	URL3_HOST	Host of the third URL in the message's body
OMTA_FROM_D	Host domain extracted from the *From-domain* part of the *Received* field at the originator SMTP server	URL3_FILE	Resource name of the third URL in the message's body

Additionally, there are two features that are not mentioned in Table 3, namely, the feature with the row number in the dataset and the feature with the class label that are used in supervised ML.

4. Experimental Settings and Results

In this section, we provide the experimental results obtained when evaluating the proposed email CTI sharing framework using the domain-specific ontology.

4.1. Dataset

The SpamAssassin public mail corpus [36] was used to parse and populate the domain-specific email ontology with the messages' metadata. The groups of messages from the SpamAssassin public mail corpus included in the labeled dataset are provided in Table 4.

The dataset was created from the SpamAssassin public mail corpus with the newly developed semantic parser. The different numbers of email messages presented in the two last columns of Table 4 indicate that the SpamAssassin collection of email messages contains a significant amount of syntactically invalid messages. During the process of semantic parsing, some messages were labeled as invalid and were not used because of the errors in their metadata. Some of the syntactical errors (e.g., invalid date formats or incorrect timestamps provided by the SMTP servers) were automatically corrected or ignored, but other errors (including invalid local addresses, domains with illegal characters, invalid charset declarations, and so on) prevented them from being included in the domain-specific ontology. Interestingly, malicious messages have considerably more problems of this kind compared with normal messages. In total, 52 features of the email messages' metadata were extracted and appended with class labels, then used to populate the dataset for supervised ML.

Table 4. The groups of messages used in the labeled dataset created for ML.

Message Group Name	Definition	Class Label	Number of Messages in the Group	Number of Messages in the Dataset
easy-ham-1	Easily detected non-spam emails	easy-ham	2500	2500
easy-ham-2	Easily detected non-spam emails collected later	easy-ham-2	1400	1397
hard-ham-1	Non-spam emails that are hard to detect	hard-ham	250	248
spam-1	Spam emails	spam	500	485
spam-2	Spam emails collected later	spam-2	1396	1331
	Total number of messages		**6046**	**5961**

4.2. Experimental Results of Evaluating the Proposed Framework

An evaluation of the performance of the proposed framework was performed using the classification leaner application from MATLAB 2024. For the first experiment, all 52 features parsed from the SpamAssassin public mail corpus were used to evaluate and select the ML algorithm with highest performance. The dataset created using MATLAB was split for training and testing. The numbers of records used for learning and testing are provided in Table 5.

Table 5. The numbers of records used for learning and testing.

Class Label	Number of Messages for Learning (~90%)	Number of Messages for Testing (~10%)	Total Number of Messages of This Class in the Dataset
easy-ham	2250	250	2500
easy-ham-2	1257	140	1397
hard-ham	224	24	248
spam	436	49	485
spam-2	1198	133	1331
Total number of records	**5365**	**596**	**5961**

The main characteristic used for evaluating the machine learning methods was accuracy, calculated using the following equation:

$$Accuracy = \frac{CP}{PN}, \quad (1)$$

where CP is the total number of all correct predictions across all classes, and PN is the total number of all predictions. The initial results of the evaluation of the method's performance are provided in Table 6.

Table 6. The initial results of the evaluation of performance.

Model Type	Accuracy, % (Validation)	Accuracy, % (Test)
Ensemble	53.6	58.14
Tree	41.94	41.95
Efficient logistic regression	41.94	41.95
Efficient linear SVM	41.94	41.95
SVM	41.94	41.95
Kernel naïve Bayes	41.94	41.95

As all classification models demonstrated low predictive accuracy, we needed to select the relevant features by removing features with low predictive power to improve the accuracy. Various feature-ranking algorithms can be used to decide which significant predictors to include. In our experiments, we used the feature-ranking algorithms chi-squared, ANOVA (analysis of variance), and the Kruskal–Wallis test in MATLAB 2024 [37]. We continued to experiment to obtain a model that performed well even without some predictors. Chi-squared [38] was the first feature-ranking algorithm used, which selected the top 28 features with higher scores and greater indicators of the features' importance. Next, using ANOVA and the Kruskal–Wallis ranking algorithms, the top 28 features with higher scores were selected accordingly.

The hyperparameters and their definitions for use in the ML models in our experiments are provided in Table 7.

Table 7. Values of the hyperparameters and their definitions.

Model Type	Names and Values of Hyperparameters	Definition
KNB	Distribution of numeric predictors: Kernel	Specifies that at least one predictor has a kernel distribution.
	Distribution name of categorical predictors: MVMN	Some predictors are categorical and are specified to be multivariate, multinomial random variables (MVMN).
	Kernel type: Gaussian	The density of kernel smoothing calculated using the Gaussian equation [39].
	Support: unbounded	This means that the density of support has real values only.
	Standardized data: Yes	Each kernel-distributed predictor variable is centered and scaled by the software using the matching column's mean and standard deviation.
Ensemble	Ensemble method: Bag Learner: Decision tree	The class labels or response variables used to train the ensemble of bagged decision trees can be supplied as a category, character, or string array; a logical or numeric vector; or a cell array of character vectors.
	Maximum number of splits: 5364	The number of messages for learning was ~90% of the total number of messages of that class in the dataset (5961).
	Number of learners: 30	The default learner was used, with 30 learners.
	Number of predictors to sample: Select All	The default (randomly chosen) number of predictive variables for each decision split, given as all.
Tree	Maximum number of splits: 100	To decrease the computation time and model complexity, trees with a depth of 100 were chosen.
	Split criterion: Gini's diversity index	Gini's diversity index was the default.
	Surrogate decision splits: Off	The dataset has no data with missing values; thus, no surrogate decision splits were used.

The best performance was demonstrated by the Kernel naïve Bayes (KNB) algorithm with the Gaussian kernel. The results are provided in Table 8.

From Table 8, we can see that the kernel naïve Bayes algorithm had the best performance. In further experiments, we refined the features to find the optimal number of features with high importance scores that improved the model to make effective predictions using the kernel naïve Bayes algorithm. The best results for all experiments were obtained by the kernel naïve Bayes algorithm.

Table 8. Evaluation of the performance when 28 features were selected using feature-ranking algorithms.

Feature-Ranking Algorithm	Type of Model	Validation					Test	
		Accuracy, %	Total Misclassification Cost	Training Time (s)	Prediction Speed (obs/s)	Model Size, MB	Accuracy, %	Total Misclassification Cost
Chi-squared	KNB	90.90	488	15.62	~2400	~3	89.09	65
	Ensemble	79.94	1076	53.35	~12,000	~49	81.21	112
	Tree	71.48	1530	8.15	~39,000	~2	70.63	175
ANOVA	KNB	93.30	359	15.6	~2700	~4	91.44	51
	Ensemble	59.45	2175	66.15	~9900	~66	59.56	241
	Tree	58.45	2229	10.87	~2500	~2	59.56	241
Kruskal–Wallis	KNB	88.53	615	28.71	~1800	~4	91.27	52
	Ensemble	64.13	1924	79.92	~9200	~8	62.08	226
	Tree	63.46	1960	13.14	~3000	~2	61.57	229

Finding the ideal set of values for the hyperparameters to produce the highest accuracy is known as hyperparameter tuning. As we already knew the appropriate range of values of the hyperparameters because of the suggested domain-specific ontology, a grid search was the most efficient method for producing a model with satisfactory performance. The results obtained with various numbers of characteristics with the highest scores are provided in Table 9.

Table 9. Evaluation of the performance of the kernel naïve Bayes algorithm with various numbers of features.

Feature-Ranking Algorithm	Number of Selected Features	Type of Model	Accuracy, % (Validation)	Accuracy, % (Test)
Chi-squared	22	Kernel naïve Bayes	93.04	92.28
ANOVA	32	Kernel naïve Bayes	93.47	91.94
Kruskal–Wallis	24	Kernel naïve Bayes	88.61	91.94

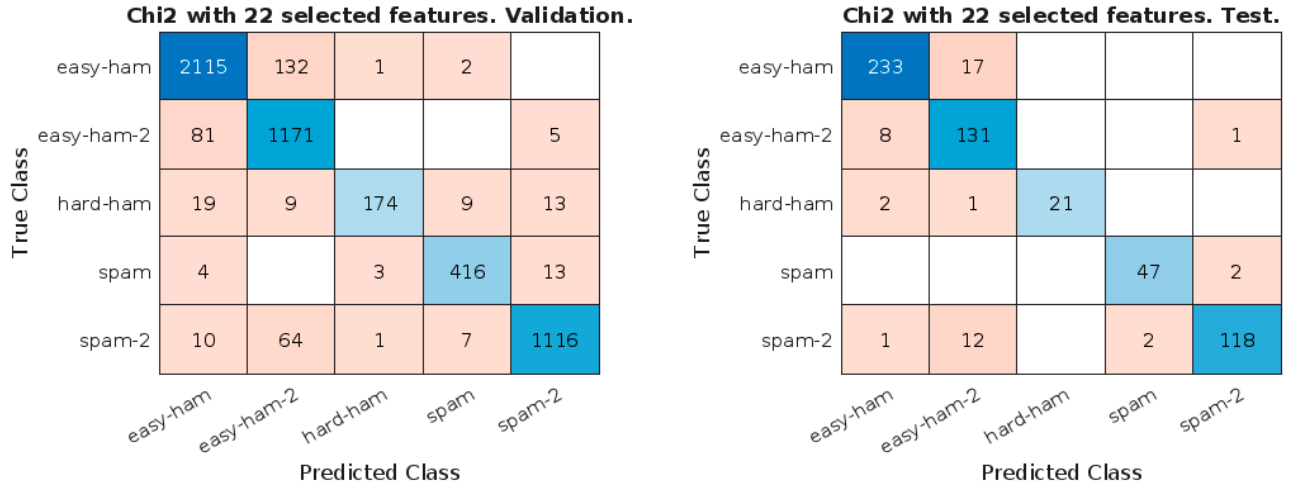
In Figures 4–6, confusion matrices with the number of observations demonstrate the models' prediction performance, as presented in Table 8.

The number of observations depicted in Figure 4 was used to calculate the metrics for evaluating the performance of the models (see Table 10), such as precision, recall, and F1 score. To calculate these additional metrics, the true positives (TP_{ClassX}), true negatives (TN_{ClassX}), false positives (FP_{ClassX}), and false negatives (FN_{ClassX}) were calculated for each class ($ClassX \in \{\text{easy} - \text{ham}, \text{easy} - \text{ham} - 2, \text{hard} - \text{ham}, \text{spam}, \text{spam} - 2\}$) independently. The precision, recall, and F1 scores were calculated for each class using the following equations:

$$Precision_{ClassX} = \frac{TP_{ClassX}}{TP_{ClassX} + FP_{ClassX}}, \quad (2)$$

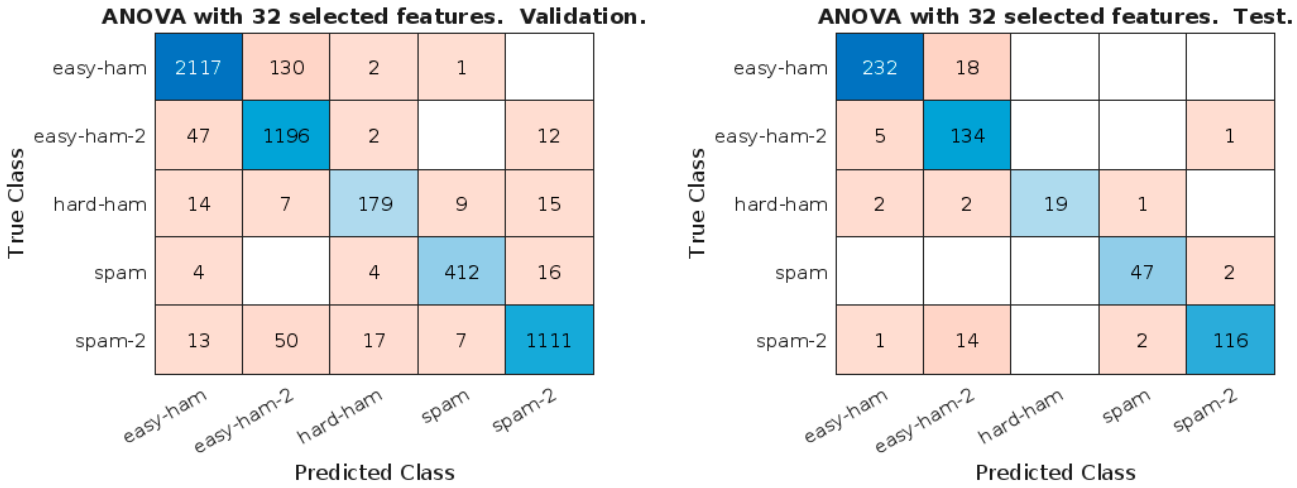
$$Recall_{ClassX} = \frac{TP_{ClassX}}{TP_{ClassX} + FN_{ClassX}}, \quad (3)$$

$$F1 - score_{ClassX} = \frac{2 \times Precision_{ClassX} \times Recall_{ClassX}}{Precision_{ClassX} + Recall_{ClassX}}. \quad (4)$$



(**a**) (**b**)

Figure 4. The confusion matrix for the kernel naïve Bayes model with 22 selected features using the chi-squared feature-ranking algorithm: (**a**) validation; (**b**) testing.

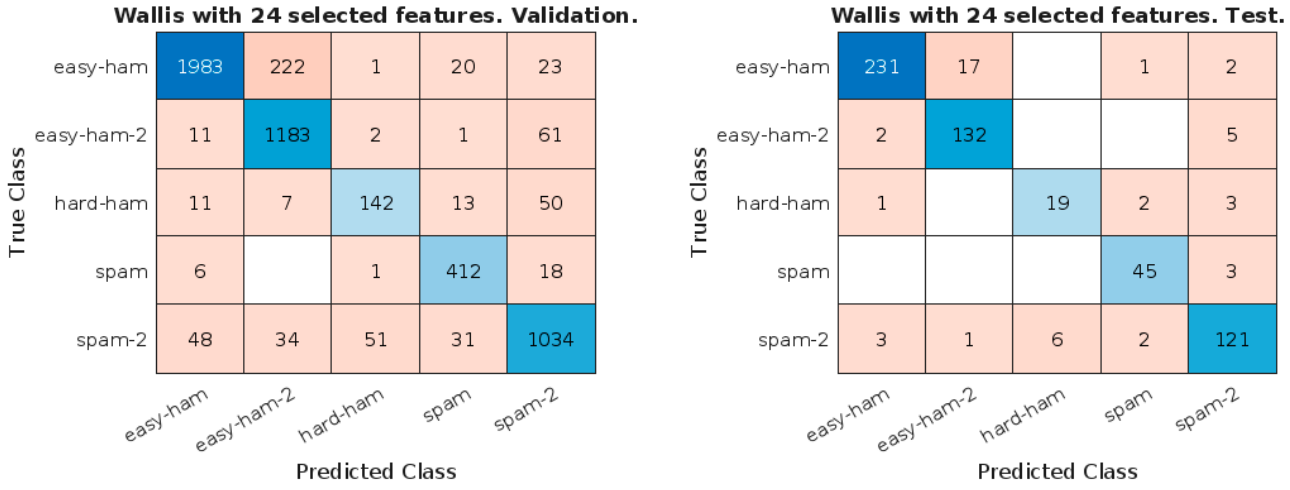


(**a**) (**b**)

Figure 5. The confusion matrix for the kernel naïve Bayes model with 32 features selected using the ANOVA feature-ranking algorithm: (**a**) validation; (**b**) testing.

Table 10. Metrics used to evaluate the performance of the models using the features selected by the chi-squared test.

Feature-Ranking Algorithm: Chi-Squared with 22 Selected Features; Kernel Naïve Bayes Model.												
Class Label	**Validation**						**Test**					
	TP	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**
easy-ham	2115	114	135	0.9489	0.9400	0.9444	233	11	17	0.9549	0.9320	0.9433
easy-ham-2	1171	205	86	0.8510	0.9316	0.8895	131	30	9	0.8137	0.9357	0.8704
hard-ham	174	5	50	0.9721	0.7768	0.8635	21	0	3	1.0000	0.8750	0.9333
spam	416	18	20	0.9585	0.9541	0.9563	47	2	2	0.9592	0.9592	0.9592
spam-2	1116	31	82	0.9730	0.9316	0.9518	118	3	15	0.9752	0.8872	0.9291



(**a**) (**b**)

Figure 6. The confusion matrix for the kernel naïve Bayes model with 24 features selected using the Kruskal–Wallis feature-ranking algorithm: (**a**) validation; (**b**) testing.

The observation results depicted in Figure 5 were used to calculate the metrics used to evaluate the performance of the models (see Table 11); namely, precision, recall, and F1 score.

Table 11. Metrics to evaluate the performance of the models using the features selected by ANOVA.

Feature-Ranking Algorithm: ANOVA with 32 Selected Features; Kernel Naïve Bayes Model.												
Class Label	**Validation**						**Test**					
	TP	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**
easy-ham	2117	78	133	0.9645	0.9409	0.9525	232	8	18	0.9667	0.9280	0.9469
easy-ham-2	1196	187	61	0.8648	0.9515	0.9061	134	34	6	0.7976	0.9571	0.8701
hard-ham	179	25	45	0.8775	0.7991	0.8364	19	0	5	1.0000	0.7917	0.8837
spam	412	17	24	0.9604	0.9450	0.9526	47	3	2	0.9400	0.9592	0.9495
spam-2	1111	43	87	0.9627	0.9274	0.9447	116	3	17	0.9748	0.8722	0.9206

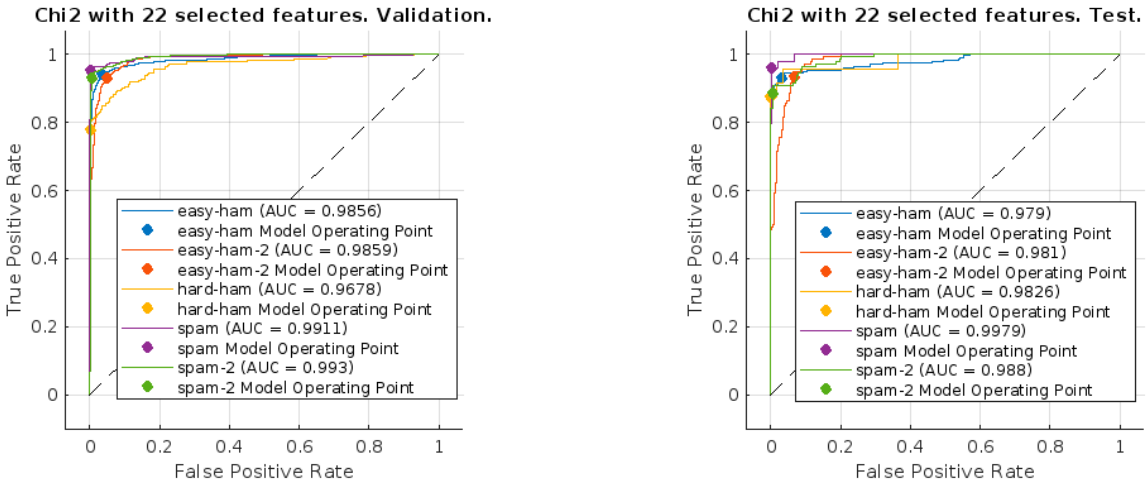
The observation results depicted in Figure 6 were used to calculate the precision, recall, and F1 score metrics to evaluate the performance of the models (see Table 12).

Table 12. Metrics used to evaluate the performance of the models using features selected by the Kruskal–Wallis algorithm.

Feature-Ranking Algorithm: Kruskal–Wallis with 24 Selected Features; Kernel Naïve Bayes Model.												
Class Label	**Validation**						**Test**					
	TP	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F1 Score**
easy-ham	1983	76	266	0.9631	0.8817	0.9206	231	6	20	0.9747	0.9203	0.9467
easy-ham-2	1183	263	75	0.8181	0.9404	0.8750	132	18	7	0.8800	0.9496	0.9135
hard-ham	142	55	81	0.7208	0.6368	0.6762	19	6	6	0.7600	0.7600	0.7600
spam	412	65	25	0.8637	0.9428	0.9015	45	5	3	0.9000	0.9375	0.9184
spam-2	1034	152	164	0.8718	0.8631	0.8674	121	13	12	0.9030	0.9098	0.9064

To rate the model's capacity to distinguish the various email classifications, ROC curves were used. A plot of the number of correct predictions in the positive class against the number of incorrectly predicted negative samples in the positive class is called a receiver operating characteristic (ROC) curve, which gauges how well the model can make predictions. The space beneath the ROC curve is measured by the area under the ROC curve (AUC). The AUC is between 0 and 1, and the model's ability to differentiate is better

the closer the AUC is to 1. Figure 7 depicts the ROC curve testing the kernel naïve Bayes model with 22 features selected using the chi-squared feature-ranking algorithm.

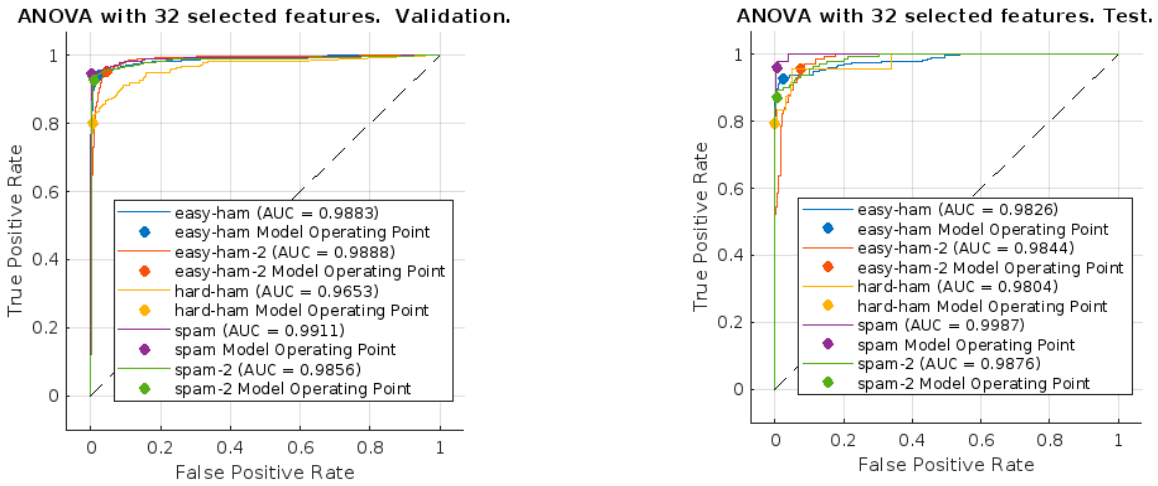


(**a**) (**b**)

Figure 7. ROC curve for the kernel naïve Bayes model with 22 selected features using the chi-squared feature-ranking algorithm: (**a**) validation; (**b**) test.

The AUCs were as follows: easy-ham, AUC = 0.979; easy-ham-2, AUC = 0.981; hard-ham, AUC = 0.9826; spam, AUC = 0.9979; spam-2, AUC = 0.988. The AUC for spam, which had the highest value of 0.9979, and the AUC for spam-2, with a value of 0.988, indicated that the tuned naïve Bayes model using 22 features selected by the chi-squared algorithm was better at distinguishing between spam and other emails.

Figure 8 depicts the ROC curve for the kernel naïve Bayes model with 32 features selected using the ANOVA feature-ranking algorithm.

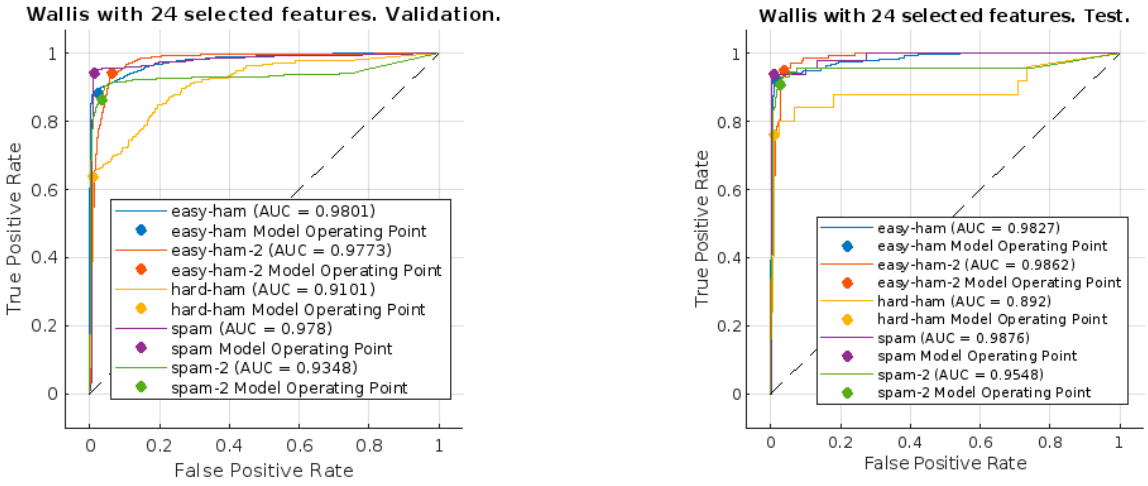


(**a**) (**b**)

Figure 8. ROC curve for the kernel naïve Bayes model with 32 features selected using the ANOVA feature-ranking algorithm: (**a**) validation; (**b**) test.

The AUCs were as follows (see Figure 8b): easy-ham, AUC = 0.9862; easy-ham-2, AUC = 0.9844; hard-ham, AUC = 0.9804; spam, AUC = 0.9987; spam-2, AUC = 0.9876. The AUC for spam, with the highest value of 0.9987, indicated that the tuned kernel naïve Bayes model using 32 features selected by ANOVA was able to distinguish between spam and other emails.

Figure 9 depicts the ROC curve for the kernel naïve Bayes model with 24 features selected using the Kruskal–Wallis feature-ranking algorithm.



(**a**) (**b**)

Figure 9. ROC curve for the kernel naïve Bayes model with 24 features selected using the Kruskal–Wallis feature-ranking algorithm: (**a**) validation; (**b**) test.

The AUCs were as follows (see Figure 9b): easy-ham, AUC = 0.9827; easy-ham-2, AUC = 0.9862; hard-ham, AUC = 0.892; spam, AUC = 0.9876; spam-2, AUC = 0.9548. The AUC for spam, with the highest value of 0.9876, indicated that the tuned kernel naïve Bayes model using the Kruskal–Wallis algorithm to select 24 features was able to distinguish between spam and other emails.

The ROC curves obtained for the models are compared in Figure 10.

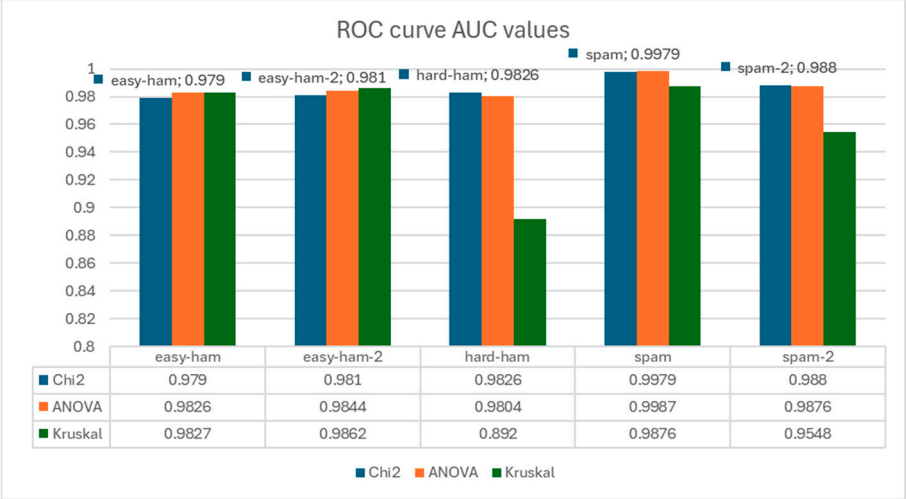


Figure 10. The AUC values of the ROC curve obtained by testing the models.

The best performance in predicting spam and ham emails was found for the kernel naïve Bayes algorithm that used the chi-squared feature-ranking algorithm with 22 features selected.

5. Discussion

CTI sharing leads to the exchange of CTI, allowing for the protection of privacy while addressing the interoperability and communication concerns of a CTI system. Most CTI sharing standards are based on ontologies that clearly define the classes, the entities, their

properties, and the relationships in a specific domain. The domain of the proposed ontology is the metadata of email messages, according to the IMF. The use of email message metadata preserves privacy since potentially confidential data are not included for further analysis.

There are many email spam datasets, as mentioned in Table 1, that researchers can use. For our experiments, the SpamAssassin public mail corpus was used with the five following groups of messages: easy-ham-1 (easily detected non-spam emails), easy-ham-2 (easily detected non-spam emails collected later), hard-ham-1 (non-spam emails that were harder to detect), spam-1 (spam emails), and spam-2 (spam emails collected later). It seems that there is an ability to convert messages into two groups (spam and ham) for binary classification. We decided to use the original message corpus with the five message groups, using a binary classification for further research.

To convert an email message into an ontological representation, a semantic parser was developed. During the process of semantic parsing, some messages from the SpamAssassin public mail corpus were labeled as invalid and were not used because of errors in their metadata. In total, 52 features of the metadata were extracted (see Table 3), appended with the class label, and then used to populate the dataset for supervised ML, as shown in Table 4.

The first experiment with various ML algorithms using the 52 features did not demonstrate an acceptable result (see Table 6). We continued the experiments to obtain a model that performed well even without some predictors by using feature-ranking algorithms (chi-squared, ANOVA, and the Kruskal–Wallis test). In all experiments, the kernel naïve Bayes model demonstrated not very high but acceptable results (see Table 8).

Optimistically, we continued the experiments by varying the number of predictors (features) to create a high-performance model (see Table 9). The highest validation accuracy (93.04%) was obtained using the ANOVA feature-ranking algorithm, with 32 predictors being selected. The highest test accuracy of 92.28% and an F1 score of 95.92% for recognizing spam were obtained using the chi-squared feature-ranking algorithm, with 22 predictors being selected.

Confusion matrices and ROC curves were created for all three experiments. Using the experimental results, the precision, recall, and F1-score were calculated (see Tables 10–12). It is known that for balanced datasets, accuracy is the best measure of performance; meanwhile, for imbalanced datasets, the best measure is the F1 score.

The experimental results were compared with state-of-the-art research (SOTA), as presented in Table 13.

Table 13. Experimental results compared with state-of-the-art research.

Research	Approach	Method	Features	Privacy	Classes	Accuracy	F1
Zavrak et al., 2023 [24]	Hierarchical attentional hybrid neural networks (HANs)	FastText + HAN model architecture	Three features: links, words, and emo-jis/emoticons	Not preserved	Spam, ham	72.3–91.6%	78.9–90.3%
Nguyen et al., 2022 [25]	Homomorphic encryption (HE) and functional encryption (FE)	A feature sparsity-based information masking method	Varying the encrypted features' vector length n between 2000 and 5000	Spam classification at a server without decrypting the email contents	Emails without target words.	75–80%	-
					Emails with target words	90%	-
Kiamarzpour et al., 2013 [26]	Used Weka software and converted the data to the ontological format	Combining the output of several decision trees and the concept of an ontology	Waikato Environment for Knowledge Analysis (Weka) explorer	Not preserved	Spam	-	84.6–94%
					Ham	-	85.6–94.2%

Table 13. *Cont.*

Research	Approach	Method	Features	Privacy	Classes	Accuracy	F1
Sathya et al., 2023 [28]	Deep learning algorithms, specifically CNN	The One-R method serves as a baseline for selecting the features	Pretrained CNN models, such as VGGNet or ResNet, were used to extract the high-level features from the visual data	Not preserved	Five categories for the detection of crime: very high, high, moderate, low, and very low	99.01%	98.76%
Omotehinwa et al., 2023 [29]	Hyperparameter optimization	Random forest (RF) and extreme gradient boosting (XGBoost) ensemble algorithms	The class label of each of the emails and the text of each email	Not preserved	Spam, Ham	97.48–98.09%	97.58–98.16%
Al-Hassan et al., 2023 [30]	DSpamOnto: an ontology-based spam detection model	Integrates a top-down methodology, with a mixed-based methodology	The MIB dataset is a collection of Twitter accounts	Not preserved	Social spambots, traditional spambots, fake followers	69.99–80.29%	59.25–75.79%
Proposed	A domain-specific ontology for emails	Semantic parser	Dataset of email metadata	Preserved	easy-ham easy-ham-2 hard-ham spam spam-2	92.28%	76.00–95.92%

As we can conclude from Table 13, various SOTA approaches can be used to classify emails; three of them use ontological representations, but only the proposed approach preserves the privacy of the email messages. Preserving privacy in ML models encourages the sharing of CTI. Additionally, the proposed approach classifies email using five classes, and the obtained performance was in line with that of other methods.

6. Conclusions

The vast majority of the approaches to the classification of emails proposed by other authors rely on natural language processing methods to analyze the content of email messages. On the other hand, if we want to construct an effective email CTI sharing framework, then the privacy of the messages' content must be preserved. The proposed method requires only the metadata of the messages to be shared, which preserves the privacy of email messages and is suitable for application in email CTI sharing solutions. The method could be applied to encrypted messages without needing to know the decryption key, as the metadata in the headers of the messages are not encrypted.

The proposed domain-specific ontology for emails helps to unambiguously interpret the meanings of the fields, an essential feature of the method, which is intended to be used in the many heterogeneous infrastructures of email service providers. The ontologies proposed by other authors aimed to semantically express the content of the messages in order to help the task of classification. The proposed ontology concentrates on the metadata of the messages, including technical fields indicating the path of the message from the initial sender to the final receiver. This is very useful for people trying to improve the efficiency of the email infrastructure. For example, this information could be used to enrich blacklists of compromised SMTP servers or to inform the administrators of such servers about potentially compromised infrastructure.

The main difference between metadata and the actual content of the messages is the ability to recognize and form strict rules (usually based on the use of specific words

and phrases) for recognizing malicious messages. The machine learning approach is one of the best solutions for this shortcoming. Modern ML methods not only provide an efficient means of recognizing essential features but also allow the optimization of the parameters of the selected model and perform actual classification with good accuracy. They are among the best alternatives to be used in the scenario under consideration. One of the biggest shortcomings of NLP-based methods is their dependence on the written language of the message. Some adversaries try to use pictures instead of text to avoid having malicious content detected. The metadata-based classification approach is immune to these weaknesses.

As mentioned above, our main contribution is the proposed email CTI sharing framework using a domain-specific ontology and a semantic parser for preserving privacy. Future work could include further application of the proposed domain-specific ontology for emails and semantic parser to different datasets, exploring the effectiveness of different proportions of the validation and test data, and investigating cross-dataset learning and testing approaches.

Author Contributions: Conceptualization, A.V., J.T. and N.M.; methodology, A.V., J.T., N.M. and F.S.; software, N.M.; validation, J.T. and N.M.; formal analysis and investigation, J.T., N.M. and F.S.; data curation, N.M.; writing—original draft preparation, A.V., J.T. and N.M.; writing—review and editing, A.V., J.T., F.S. and N.M.; visualization, J.T. and F.S.; supervision, A.V. and F.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was conducted as part of the execution of the project "Mission-driven Implementation of Science and Innovation Programs" (No. 02-002-P-0001), funded by the Economic Revitalization and Resilience Enhancement Plan "New Generation Lithuania".

Data Availability Statement: The newly created domain-specific email ontology and metadata dataset are publicly available at https://doi.org/10.17632/tgm39cfggr.1.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jesus, V.; Bains, B.; Chang, V. Sharing Is Caring: Hurdles and Prospects of Open, Crowd-Sourced Cyber Threat Intelligence. *IEEE Trans. Eng. Manag.* **2023**, *71*, 6854–6873. [CrossRef]
2. Mujtaba, G.; Shuib, L.; Raj, R.G.; Majeed, N.; Al-Garadi, M.A. Email Classification Research Trends: Review and Open Issues. *IEEE Access* **2017**, *5*, 9044–9064. [CrossRef]
3. Noor, U.; Anwar, Z.; Amjad, T.; Choo, K.R. A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise. *Future Gener. Comput. Syst.* **2019**, *96*, 227–242. [CrossRef]
4. Sakellariou, G.; Fouliras, P.; Mavridis, I.; Sarigiannidis, P.A. Reference Model for Cyber Threat Intelligence (CTI) Systems. *Electronics* **2022**, *11*, 1401. [CrossRef]
5. Ramsdale, A.; Shiaeles, S.; Kolokotronis, N. A Comparative Analysis of Cyber-Threat Intelligence Sources, Formats and Languages. *Electronics* **2020**, *9*, 824. [CrossRef]
6. Hitzler, P.; Krötzsch, M.; Rudolph, S. *Foundations of Semantic Web Technologies*; Chapman & Hall/CRC: Boca Raton, FL, USA, 2009.
7. The MITRE Corporation about CAPEC. Available online: https://capec.mitre.org/index.html (accessed on 22 June 2024).
8. Roy, S.; Panaousis, E.; Noakes, C.; Laszka, A.; Panda, S.; Loukas, G. SoK: The MITRE ATT&CK Framework in Research and Practice. 2023. Available online: https://arxiv.org/abs/2304.07411 (accessed on 22 June 2024).
9. Al-Sada, B.; Sadighian, A.; Oligeri, G. MITRE ATT&CK: State of the Art and Way Forward. 2023. Available online: https://arxiv.org/abs/2308.14016 (accessed on 22 June 2024).
10. OASIS Open. Introduction to STIX. 2019. Available online: https://oasis-open.github.io/cti-documentation/stix/intro.html (accessed on 22 June 2024).
11. Jordan, B.; Varner, D. TAXII Version 2.1. OASIS Standard. 10 June 2021. Available online: https://docs.oasis-open.org/cti/taxii/v2.1/os/taxii-v2.1-os.pdf (accessed on 31 May 2024).
12. Syed, Z.; Padia, A.; Finin, T.W.; Mathews, M.L.; Joshi, A. UCO: A Unified Cybersecurity Ontology. In Proceedings of the AAAI Workshop: Artificial Intelligence for Cyber Security, Phoenix, AZ, USA, 12 February 2016. [CrossRef]
13. Preuveneers, D.; Joosen, W. An Ontology-Based Cybersecurity Framework for AI-Enabled Systems and Applications. *Future Internet* **2024**, *16*, 69. [CrossRef]
14. Onwubiko, C. CoCoa: An Ontology for Cybersecurity Operations Centre Analysis Process. In Proceedings of the 2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA), Glasgow, UK, 11–12 June 2018; pp. 1–8.

15. Mozzaquatro, B.A.; Agostinho, C.; Goncalves, D.; Martins, J.; Jardim-Goncalves, R. An Ontology-Based Cybersecurity Framework for the Internet of Things. *Sensors* **2018**, *18*, 3053. [CrossRef] [PubMed]
16. Huang, C.-C.; Huang, P.-Y.; Kuo, Y.-R.; Wong, G.-W.; Huang, Y.-T.; Sun, Y.S.; Chang Chen, M. Building Cybersecurity Ontology for Understanding and Reasoning Adversary Tactics and Techniques. In Proceedings of the 2022 IEEE International Conference on Big Data (Big Data), Osaka, Japan, 17–20 December 2022; pp. 4266–4274.
17. Saidani, N.; Adi, K.; Allili, M.S. A semantic-based classification approach for an enhanced spam detection. *Comput. Secur.* **2020**, *94*, 101716. [CrossRef]
18. Jeeva, L.; Khan, I.S. A Review Article On Enhancing Email Spam Filter's Accuracy Using Machine Learning. *Int. J. Innov. Res. Comput. Sci. Technol.* **2023**, *11*, 5–11. [CrossRef]
19. Gibson, S.; Issac, B.; Zhang, L.; Jacob, S.M. Detecting Spam Email With Machine Learning Optimized With Bio-Inspired Metaheuristic Algorithms. *IEEE Access* **2020**, *8*, 187914–187932. [CrossRef]
20. Jáñez-Martino, F.; Alaiz-Rodríguez, R.; González-Castro, V.; Fidalgo, E.; Alegre, E. A review of spam email detection: Analysis of spammer strategies and the dataset shift problem. *Artif. Intell.* **2023**, *56*, 1145–1173. [CrossRef]
21. Schaad, J.; Ramsdell, B.; Turner, S. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 4.0 Message Specification. RFC 8551. 2019. Available online: https://www.rfc-editor.org/info/rfc8551 (accessed on 1 July 2024). [CrossRef]
22. Ainslie, S.; Thompson, D.; Maynard, S.; Ahmad, A. Cyber-threat intelligence for security decision-making: A review and research agenda for practice. *Comput. Secur.* **2023**, *132*, 103352. [CrossRef]
23. Sun, N.; Ding, M.; Jiang, J.; Xu, W.; Mo, X.; Tai, Y.; Zhang, J. Cyber Threat Intelligence Mining for Proactive Cybersecurity Defense: A Survey and New Perspectives. *IEEE Commun. Surv. Tutor.* **2023**, *20*, 1186–1199. [CrossRef]
24. Zavrak, S.; Yilmaz, S. Email spam detection using hierarchical attention hybrid deep learning method. *Expert Syst. Appl.* **2023**, *233*, 120977. [CrossRef]
25. Nguyen, T.; Karunanayake, N.; Wang, S.; Seneviratne, S.; Hu, P. Privacy-preserving spam filtering using homomorphic and functional encryption. *Comput. Commun.* **2023**, *197*, 230–241. [CrossRef]
26. Kiamarzpour, F.; Dianat, R.; Bahrani, M.; Sadeghzadeh, M. Improving the methods of email classification based on words ontology. *arXiv* **2013**, arXiv:1310.5963. Available online: https://arxiv.org/ftp/arxiv/papers/1310/1310.5963.pdf (accessed on 1 July 2024).
27. Wang, M.; Song, L. An Incentive Mechanism for Reporting Phishing E-Mails Based on the Tripartite Evolutionary Game. *Secur. Commun. Netw.* **2021**, *2021*, 3394325. [CrossRef]
28. Sathya, J.; Mary Harin Fernandez, F. An Optimizing Crime Detection in Social Media Platforms Using Multiagent Ontology-Based Approach. In Proceedings of the 4th International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 20–22 September 2023.
29. Omotehinwa, T.O.; Oyewola, D.O. Hyperparameter Optimization of Ensemble Models for Spam Email Detection. *Appl. Sci.* **2023**, *13*, 1971. [CrossRef]
30. Al-Hassan, M.; Abu-Salih, B.; Al Hwaitat, A. DSpamOnto: An Ontology Modelling for Domain-Specific Social Spammers in Microblogging. *Big Data Cogn. Comput.* **2023**, *7*, 109. [CrossRef]
31. Venčkauskas, A.; Toldinas, J.; Morkevičius, N.; Sanfilippo, F. Email Domain-specific Ontology and Metadata Dataset. *Mendeley Data* **2024**. [CrossRef]
32. Resnick, P. Internet Message Format; RFC Editor. 2008. p. RFC5322. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc5322.txt.pdf (accessed on 23 June 2024).
33. Sirbu, M.A. Content-Type Header Field for Internet Messages; RFC Editor. 1988. p. RFC1049. Available online: https://www.rfc-editor.org/rfc/pdfrfc/rfc1049.txt.pdf (accessed on 23 June 2024).
34. Freed, N.; Borenstein, N. Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies; RFC Editor. 1996. p. RFC2045. Available online: https://www.ietf.org/rfc/rfc2045.txt (accessed on 23 June 2024).
35. Klensin, J. Simple Mail Transfer Protocol. RFC Editor. 2008, p. RFC5321. Available online: https://datatracker.ietf.org/doc/html/rfc5321 (accessed on 23 June 2024).
36. SpamAssassin. Available online: https://github.com/stdlib-js/datasets-spam-assassin (accessed on 18 May 2024).
37. Feature Selection and Feature Transformation Using Classification Learner App. Available online: https://se.mathworks.com/help/stats/feature-selection-and-feature-transformation.html#buwh5ae-1 (accessed on 18 May 2024).
38. Liu, H.; Setiono, R. Chi2: Feature selection and discretization of numeric attributes. In Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, 5–8 November 1995; pp. 388–391. [CrossRef]
39. Train Multiclass Naive Bayes Model. Available online: https://se.mathworks.com/help/stats/fitcnb.html (accessed on 22 June 2024).