algorithms

MDPI

Article

Framework Based on Simulation of Real-World Message Streams to Evaluate Classification Solutions

Wenny Hojas-Mazo [1], Francisco Maciá-Pérez [2], José Vicente Berná Martínez [2], Mailyn Moreno-Espino [3], Iren Lorenzo Fonseca [2] and Juan Pavón [4,*]

[1] Departamento de Inteligencia Artificial e Infraestructura de Sistemas Informáticos, Facultad de Ingeniería Informática, Universidad Tecnológica de La Habana, José Antonio Echeverría, Calle 114 #11901, entre 119 y 127, CUJAE, Marianao, La Habana 19390, Cuba; whojas@ceis.cujae.edu.cu
[2] Department of Computer Science and Technology, University of Alicante, 03690 Alicante, Spain; pmacia@ua.es (F.M.-P.); jvberna@ua.es (J.V.B.M.); iren.fonseca@ua.es (I.L.F.)
[3] Centro de Investigación en Computación, Instituto Politécnico Nacional, Ciudad de México 07738, Mexico; mmorenoe2022@cic.ipn.mx
[4] Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid, 28040 Madrid, Spain
* Correspondence: jpavon@fdi.ucm.es

Abstract: Analysing message streams in a dynamic environment is challenging. Various methods and metrics are used to evaluate message classification solutions, but often fail to realistically simulate the actual environment. As a result, the evaluation can produce overly optimistic results, rendering current solution evaluations inadequate for real-world environments. This paper proposes a framework based on the simulation of real-world message streams to evaluate classification solutions. The framework consists of four modules: message stream simulation, processing, classification and evaluation. The simulation module uses techniques and queueing theory to replicate a real-world message stream. The processing module refines the input messages for optimal classification. The classification module categorises the generated message stream using existing solutions. The evaluation module evaluates the performance of the classification solutions by measuring accuracy, precision and recall. The framework can model different behaviours from different sources, such as different spammers with different attack strategies, press media or social network sources. Each profile generates a message stream that is combined into the main stream for greater realism. A spam detection case study is developed that demonstrates the implementation of the proposed framework and identifies latency and message body obfuscation as critical classification quality parameters.

Keywords: classification; evaluation; non-stationary message streams; simulation

Citation: Hojas-Mazo, W.; Maciá-Pérez, F.; Berná Martínez, J.V.; Moreno-Espino, M.; Lorenzo Fonseca, I.; Pavón, J. Framework Based on Simulation of Real-World Message Streams to Evaluate Classification Solutions. *Algorithms* **2024**, *17*, 47. https://doi.org/10.3390/a17010047

Academic Editor: Nuno Fachada

Received: 30 December 2023
Revised: 18 January 2024
Accepted: 19 January 2024
Published: 21 January 2024

1. Introduction

Information exchange through multiple communication channels (mail, SMS, internal applications, RSS, etc.) from different senders and to one or more recipients plays a key role in institutions [1] and companies [2] and generates a large, dynamic message stream. An example context of relevance to institutions and companies is to better understand how rational and emotional postings on social media influence customer behaviour [3,4].

The underlying characteristics of these dynamic message streams pose some serious challenges to effective classification, such as concept drift, concept evolution, latency and adversarial attacks [5,6]. First, the concepts embedded in a stream change over time. This is known as concept drift and requires a classifier to adapt to the current concepts. For example, a reader's topic of interest may change over time after reading a large number of messages with different topics. Second, a message stream usually consists of a large number of objects (instances), and these objects are characterised by a high-dimensional feature space (e.g., the message topics referred to in a message stream are described by a large vocabulary). Third, latency, verification latency or delay, is the time between the availability

of an unlabelled instance and its actual labelling [7]. This period can be measured in terms of time or number of instances. Since most benchmark datasets do not have time stamps, the number of instances is usually used in the literature as a measure of latency. The occurrence of such latencies has a direct impact on the model update strategy during drift events, which can lead to a decrease in classifier accuracy [8]. The latency can be divided into null latency, extreme latency and intermediate latency [8]. At null latency, the real labels of the instances are always available immediately after classification. At extreme latency the real labels are never available to the classifier, requiring an unsupervised approach or an incremental update of the model over time. At intermediate latency, an intermediate delay time L, where $0 < L < \infty$, is considered until the real labels are available. This time can be constant (the same for all stream instances) or variable. If the delay is variable, the time of availability of the real labels may differ from the arrival order of the examples, and therefore, the classifier may receive the label of the instance $\overrightarrow{x}_{t+5}$ before receiving the label of $\overrightarrow{x}_{t+2}$. Fourth, the adversarial attacks in the context of the message stream are carried out by the adversarial character known as the spammer. Spammers try to evade the classifier while maintaining the readability of the message content, for example, by including certain misspellings or authentic words in the message [9]. As a result, spam messages may contain malicious information strategically inserted by spammers to corrupt the data used to train classifiers. In [6], a detailed analysis is made of the tricks used by spammers to evade spam filters, such as text poisoning, obfuscated words or hidden text salting.

In order to evaluate the performance of classifiers on dynamic message streams, different corpus [6,10], different measures [6,10] and evaluation methods [6,11] have been published. In general, most models show high accuracy when evaluated on known and relevant public datasets [6,11]. This situation contributes to the generation of overly optimistic results and makes the actual deployment of message classification solutions uncertain, since the problems of adversarial data manipulation by spammers and concept drift are often ignored [6,11]. However, some proposals have presented forms of evaluation that take these aspects into account [6,11–15].

Proposals for evaluating adversarial data manipulation by spammers in the message stream context have not been found. However, proposals have been identified in other contexts that attempt to measure classifier stability in the face of adversarial attacks [12,13]. In [12], the stability of the classifier under adversarial contamination of the training data is quantified by introducing a metric to classify its robustness. In [13], frameworks for security analysis and evaluation of classification algorithms are proposed by simulating attack scenarios.

Regarding the evaluation of concept drift in the message stream context, two were found, specifically in spam detection [6,11]. In [11], the SDAI methodology is proposed, based on the measurement of classifier accuracy in four different but complementary scenarios: static, dynamic operations, adaptive capabilities and internationalisation. The static scenario evaluates the overall performance of the classifier in a controlled environment, while the dynamic operation scenario measures its behaviour under automatic updating schemes. The adaptive capabilities scenario simulates the operation of the classifier in a server context, categorising messages from multiple senders and covering different subject areas, and the internationalisation scenario evaluates the ability of the classifier to classify incoming messages in different languages, whether in standalone or server mode. In [6], a strategy very similar to the adaptive scenario of [11] is proposed, where the classifier is trained on a corpus (SpamAssassin corpus) and tested on another corpus (Ling-Spam corpus), which covers different topics than the training corpus. The main differences of the variant in [6] are the algorithms used to process and classify the messages and the corpora used for training and testing. In [6], four spam filters trained on five email datasets collected from different sources at different times are evaluated to see if the spam filters maintain their generalisation performance. Both proposals essentially try to deal with concept drift and spammer influence, ref. [11] in the dynamic scenario and both in the adaptive scenario. However, in the dynamic scenario, the following perspective is oriented

towards the action of the classifier rather than the generation of the stream. The stream is generated using cross-validation by segmenting the base corpus into 10 partitions and selecting 9 for training and 1 for testing. This procedure does not allow adequately modelling environments with intermediate and extreme latency; the behaviour of the different spammers and other message sources that may exist is limited to what is reflected in the messages of the corpus used and does not allow adapting the behaviour by adjusting parameters such as spammer strategies, message sending ratio, message batch size and other parameters that make the stream more realistic; and the configuration to evaluate different types of message processing services, classifiers or a combination of both can become complex.

In addition, several frameworks have been proposed to assist users in carrying out specific aspects of the stream classification process [16]. These frameworks allow the integration of different stream classification tasks, improve interoperability and include all necessary components for algorithm development [17]. A stream classification framework may include data generators or real-world datasets for benchmarking, data processing, classifier algorithms, and testing of classification results [16]. Existing frameworks that have been used to classify text messages include Jubatus [14] and Massive Online Analysis (MOA) [15]. Jubatus, a framework that emerged from a Japanese research project [14], emphasises distributed processing and features a model-sharing architecture to support practical training and collaboration of classification models. This aims to reduce the network costs and latency associated with distributed environments. The framework incorporates test datasets from various sources and includes features for feature space simplification and textual data preprocessing. It also integrates basic stream classification algorithms, provides minimal evaluation and monitoring functionality, and is compatible with the Spark analytics engine or the Python sci-kit-learn library [18]. MOA [15] is derived from WEKA, the Waikato Environment for Knowledge Analysis framework, written in Java and accessible via a graphical user interface (GUI) or command line. Originally developed to assess stream classification performance and manage algorithm speed, memory usage and accuracy MOA includes numerous datasets, preprocessing approaches, stream-classification-related algorithms and evaluation methods. Advanced applications include extensions for tweet collection, sentiment analysis and data reduction techniques in evolving streams. Both frameworks do not allow adequate modelling of environments where multiple message stream generators are simultaneously configured to emulate the behaviour of spammers and other message sources that may exist and contribute to the main message stream (limited to what is reflected in the messages of the corpus used).

This research hypothesises that the modelling of the sources of message emission to the main stream as profiles with configurable behaviours, and the modelling of the stream according to simulation techniques and queueing theory, will provide a message stream to evaluate message processing and/or classification solutions closer to the real one, thus allowing to detect in advance possible problems of processors and/or classifiers and to avoid degradation of their performance during their operation.

This paper presents a framework that simulates real-world message streams to evaluate classification solutions. The framework consists of four modules: message stream simulation, processing, classification and evaluation. The first module uses simulation techniques and queueing theory to replicate real-world message streams. The next module enhances the input messages for effective classification. The third module applies existing classification solutions to categorise the generated stream. Finally, the fourth module evaluates the performance of the classification solutions by measuring various metrics.

The rest of the paper is structured as follows: Section 2 describes the details of the proposed framework. Section 3 develops a case study in the spam detection environment to evaluate the proposed framework. Finally, Section 5 summarises the main conclusions.

2. Proposed Framework

This paper proposes a framework based on the simulation of real message streams for the evaluation of classification solutions. The proposed framework aims to provide researchers (especially data scientists) in the area of text stream classification with an environment that facilitates the evaluation of text message processing and/or classification solutions. The proposal is designed to facilitate the integration of existing processing and/or classification solutions, mainly in the form of web services, for evaluation with the generated message streams or as a basis for comparison with other solutions. Figure 1 shows a view of the framework architecture, which aims to make this framework as versatile as possible, with the ability to adapt and readjust the technological resources to the changing situation at any time, and which follows a similar scheme to the architecture proposed in [19]. An architectural style based on n-layer architectures has been used, structuring the elements into levels.

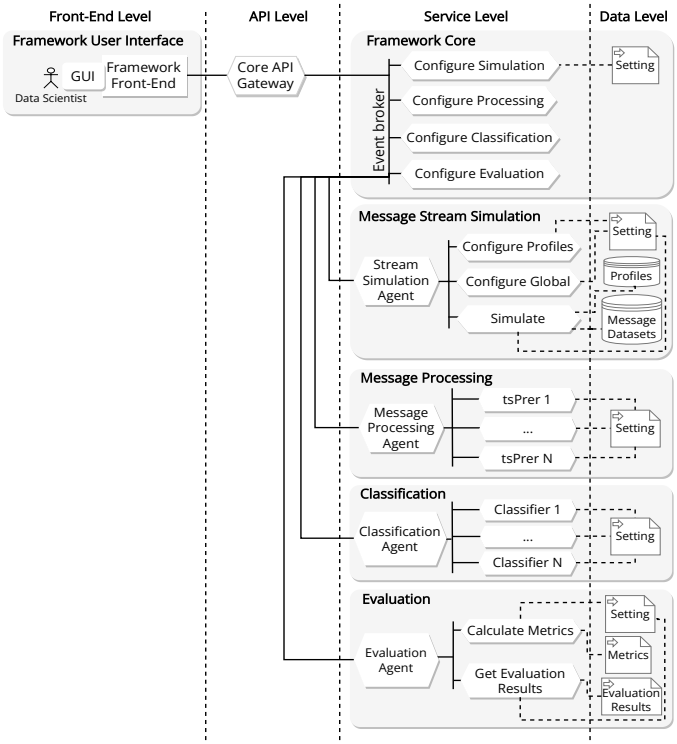

Figure 1. Framework architecture view.

The front-end level defines the elements needed to interact with end users, grouped under the concept of the framework user interface and organised according to the model, view, controller (MVC) architectural pattern. The framework front-end is intended for the data scientists, and is based on a graphical user interface (GUI). The API level contains the service that acts as an application programming interface (API) for the other of the services. This allows for the centralised exposure of all the endpoints defined in the backend. This level is the core of the solution and consists of five main modules: Framework Core, Message Stream Simulation, Message Processing, Classification, and Evaluation. Each of the modules in the service level accesses the data level through the persistence services in the service layer (Figure 1 shows the relationship between the services and their main data sources, with dashed lines to indicate that this relationship is indirect). The following is a description of each of the core-level modules that make up the proposed framework.

2.1. Framework Core

The Framework Core consists of a set of services that provide the basic functionality of the Framework Service: Configure Simulation, Configure Processing, Configure Classifi-

cation and Configure Evaluation. All these services are connected to the Gateway and to each other via an Event Broker that manages the messages.

2.2. Message Stream Simulation

Message Stream Simulation, using the simulation technique [20] and the queueing theory [21], attempts to mimic the generation of a message stream, similar to a real-world environment. The goal of this module is to generate a message stream S that is as close as possible to a real scenario. To achieve this goal, profiles are modelled. A profile p is considered in this paper as an abstract entity that can have n instances, each of which generates a message stream S_{p_n}. Profiles can be used to model thematic sources and/or user profiles. A thematic source generates a message stream about one or more topics such as Computers, Science, Society and others. A user profile generates a message stream with similar characteristics to user/entity types of the web such as Spammer, Social Network, Personal, Marketing, Information and others. A profile instance is defined as $p_n = (name, svb, mc, S_{p_n})$, where *name* is the identifier of each instance profile, *svb* is a set of variable behaviours that can be different for each profile instance, *mc* is a message corpus and S_{p_n} is the generated message stream.

The set of variable behaviours (*svb*) are parameters that can change the way a profile instance generates a message stream. When simulating non-stationary message streams to closely emulate message sending sources, the variables message send rate, message batch size, message size and message obfuscation play an essential role. The message send rate represents the frequency with which messages are sent by the source, and in a non-stationary environment, the message send rate can fluctuate over time. By adjusting this variable in the simulation, one can capture the varying intensity of message traffic. Similarly, the message batch size, which refers to the number of messages sent together as a batch, affects the burstiness and temporal patterns of the message streams. In non-stationary scenarios, batch size can change, and adjusting this variable in the simulation allows for the replication of such dynamics. Message size refers to the size in digital units of the content of each message. In non-stationary message streams, the size distribution of messages may change, and incorporating this variability into the simulation will affect the processing of message content, as the content and size of the context to be analysed by the word processor will vary. In addition, message obfuscation is one of the tricks used by spammers to evade spam filters and is relevant to the evaluation of solutions to be developed in the context of spam detection. Future versions of the framework could include the modelling of other tricks used by spammers. Based on that, the *svb* Message Send Rate (*msr*), Message Batch Size (*mbs*), Message Size (*ms*) and Message Obfuscation (*mo*) are defined. *msr* is equal to mean arrival rate λ, which is the time between sending messages. λ can be a constant or a random value that can change in a constant time t_λ. In *mbs*, the size s of the message batch *mb* can be a constant, or it can change randomly in a constant time t_{mb}. To change s, an integer uniform random number is generated with an interval $[0, z]$, where z is the maximum value of the size for *mb*. In *mbs*, before forming each *mb*, the input source can be filtered by a minimum message size *ms* that changes in constant time t_{ms}. This reduces the input source to form the *mb* to the message with a size equal to or greater than *ms*. To change *ms*, a uniform integer random number is generated with an interval $[1, k]$, where k is the maximum value of KB for *ms*. The *mo* consists of replacing the letters i and l in the body of the message with the characters ¡ and 1, respectively. Before sending each *mb*, q message from the batch of messages can be selected and obfuscated. To select the q messages, an integer uniform random number with an interval $[0, v]$ is generated, first to select the set of messages to be obfuscated, and second to select the q messages to be obfuscated (the last time with an interval $[1, v]$ without repetitions).

The *mc* form a message corpus about the profile class, which can receive messages from the public corpus or from personal sources (e.g., emails from a mailbox). A message corpus is defined as $mc = \{m_1, m_2, \ldots, m_n\}$, where m_i is the i-th message in the corpus.

S_{p_n} is defined as:

$$S_{p_n} = \begin{pmatrix} m_{1,1}, & m_{1,2}, & \cdots & m_{1,r_1}; \\ m_{2,1}, & m_{2,2}, & \cdots & m_{2,r_2}; \\ \cdots; & & & \\ m_{n,1}, & m_{n,2}, & \cdots & m_{n,r_n}; \\ \cdots; & & & \end{pmatrix}$$

Here, $m_{i,j}$ represents the j-th text at the i-th time. The union of the message streams generated by each profile instance forms the message stream, so it is defined as $S = \cup_{p=1}^{l} S_p \mid p \in P$, where $S_p = \cup_{n=1}^{q} S_{p_n}$.

The general modelling of profile behaviour is based on simulation concepts [20] and the M/M/1 queueing model (Poisson input, exponential service times and single server) [21]. To summarise the physical operation of the system, incoming messages enter the queue, are eventually served, and then leave. It is therefore necessary for the simulation model to describe and synchronise the arrival of messages and the serving of messages. The general behaviour of the profile instance is shown in Figure 2.

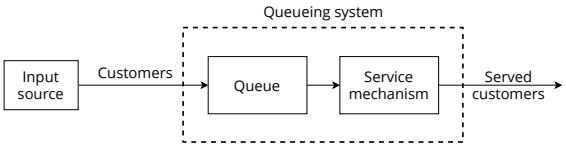

Figure 2. General behaviour of profile instance.

Starting at time 0, the simulation clock records the (simulated) time t that has elapsed during the simulation run so far. The information about the queueing system that defines its current status, i.e., the state of the system, is $N(t)$ = number of messages in the system at time t. The events that change the state of the system are the arrival of messages or a service completion for the messages currently in service (if any). The values of mean arrival rate λ and mean service rate μ are $t_{ms}/h * r$ and t_{ms}, respectively. Each profile instance has independent t_{ms}, r and h values. The state transition formula is:

$$\text{Reset } N(t) = \begin{cases} N(t) + n & \text{if the arrival of } n \text{ texts occurs at time } t \\ N(t) - n & \text{if service completion of } n \text{ messages occurs at time } t \end{cases}$$

The event generation method consists in forming a message batch $mb = \{m_1, m_2, \ldots, m_r\}$, where m_i represents the i-th message in the batch, selecting from n messages of the input source by generating an integer uniform random number for each message and sending it to the queue. The random numbers have an interval of $[1, n]$. This method of event generation allows us to assume that the input source is infinite, since n messages can be selected from each element of mb, generating n^r batches, which tend to be infinite. In the queue system, the queue discipline is first-come-first-served. The service mechanism consists in sending a message batch $mb = \{m_1, m_2, \ldots, m_r\}$ in a μ.

The training examples profile will have a single instance that generates a message stream by selecting message examples and their real classes from the messages sent by the created profile instances. The examples in this stream can be used by the evaluated solutions in the learning process. The latency *lat* is the time between the prediction of an unlabelled instance by a classifier and the availability of its real label by the environment, and is classified as either null, intermediate or extreme latency.

2.3. Message Processing

Message Processing transforms the message input into a high-quality one that is suitable for the learning process to follow, using techniques such as integration, normalisation, cleaning, transformation and reduction. Data processing [22] stands out as a crucial stage in the knowledge discovery process. Although often overlooked in comparison to other phases such as data mining, data processing typically requires a greater investment of

time and effort, accounting for more than 50% of the overall undertaking [23]. Raw data typically contain numerous imperfections, including inconsistencies, missing values, noise and redundancies. Consequently, the effectiveness of successive learning algorithms is inevitably compromised in the presence of poor data quality [24]. It follows that the application of appropriate processing measures has a significant impact on the quality and reliability of the resulting automated insights and decisions.

The message stream processing module consists of a set of text stream preprocessors, denoted as $TsPre = \{tsPre_1, tsPre_2, \ldots, tsPre_n\} | n \geq 1$, which typically employ natural language processing techniques and feature analysis (lexical terms). Commonly used natural language processing techniques include text content extraction, tokenisation and part-of-speech tagging. Following the application of natural language processing techniques, feature analysis is used to reduce the dimensionality of the set of features present in the texts through selection and/or reduction, with the aim of representing the content in a structure that is amenable to classification solutions. This module can be used for the following purposes:

- Evaluate the message stream processing solutions to be developed. To perform this evaluation, several processing solutions are selected that will process the same message stream to obtain the feature sets that will then be used by one or more classification algorithms. Since what would vary in the classification process is the way the message stream is preprocessed, the better the quality of the results, the better the processing solution used.
- Focus the evaluation on the classification process. One of the solutions available in the module is used for all classification solutions used, which means that the quality of the result depends on the classification algorithm used. Therefore, this approach allows the comparison of classification algorithms without the need to preprocess the message stream.
- Identifying the correlation between variants of text processing solutions and classification algorithms. This makes it possible to identify the best-performing combinations of classification algorithms and text stream processing solutions.

The use of this module is optional, as classification solutions may internally include a message stream processing component. This variant is also taken into account in the proposed evaluation framework by disabling the use of this module. The main value of this module is not the text processing algorithms it uses, but the possibility of integrating existing or developing text processing algorithms into the framework through the use of web services and their joint evaluation. This variant makes it possible to distribute the execution load of the sorting processing algorithms to different computing nodes, which process the messages of the stream sent to them and return the sorting result to the framework.

2.4. Classification

Based on learning algorithms, Classification classifies the messages of the generated stream. The classification module can aggregate different classification solutions that exist in the literature or are proprietary. The module consumes these solutions as web services that are available and comply with an input and output format. The solutions of the module to be evaluated classify the generated message streams into different categories. The same message stream can be classified by more than one classification solution, which allows comparing the performance of n solutions with the same message stream generated in the simulation. In this framework, classification solutions that include a text processing component can be evaluated. To train these classifiers, 90% of a set of messages, such as the messages in the SpamAssassin corpus, is used. The main value of this module is not the classification algorithms it uses, but the possibility of integrating existing or developing classification algorithms into the framework through the use of web services and their joint evaluation. This approach allows the execution load of sorting and classification algorithms

to be distributed across different computing nodes. These nodes process the messages within the stream sent to them and then send the sorting results back to the framework.

2.5. Evaluation

Evaluation evaluates the performance of the learners used to classify the stream by measuring accuracy, precision, recall, positive true, positive false, negative true and negative false. The evaluation module receives the output of the classifiers used and the real classes of each message in the stream generated in the simulation. From this information, measures of accuracy, precision, recall, true positives, false positives, true negatives and false negatives are applied to obtain the performance of the different classification solutions. This allows comparisons to be made with other classifiers using the same input parameters.

The evaluation module allows you to design and run experiments to evaluate message stream analysis solutions and analyse the results. Designing experiments involves organising the following initial elements: setting the simulation time, configuring the message stream generator, configuring text processing (if enabled), configuring classification and configuring the evaluation process.

By setting the simulation time it is possible to regulate the duration of the experiment to be carried out. The configuration of the message stream generator consists of defining and/or creating the profile instances that will constitute the source that generates the message streams to be processed in the simulation. Instances may exist previously or be created in the same design of the experiment, although the instance of training examples is unique, as explained in the description of the Message Stream Simulation module. Existing instances can be selected without further adjustment of the *sub* parameters, although they can be changed at the experimenter's discretion. To create a profile instance, it is first named, classified according to profile types, and it is decided which messages will form the corpus. Parameter values are then set to adjust the various *sub* of the profile instance to suit the experimenter.

In the processing configuration, you can choose whether to use the processing module or not. The classification solutions to be evaluated may include the processing stage and therefore the use of this module would not be necessary. On the other hand, the choice to use the processing module may be due to the use of a basic processing that allows to focus only on the evaluation of the classification process, or to evaluate a processing solution by comparing it with other solutions and seeing the classification behaviour when receiving the outputs of the different preprocessors that can be compared by one or more available classification solutions.

3. Case Study: Spam Email Detection

This section presents a case study, Spam Email Detection, where an implementation (https://github.com/Cujae-IF/FrameworkToEvaluateMessageClassification.git accessed on 16 January 2024) of the proposed framework is used to evaluate solutions in spam detection scenarios. The Spam Email Detection case study aims to measure the influence of email streams on the quality (accuracy, precision and recall) of spam email detection. For this purpose, the proposed framework will be implemented to generate mail streams from the SpamAssassin corpus [25] using the Message Stream Simulation module, to classify them using a built-in test classifier (LearningAntiSpamServer) in the Classification module, and finally to evaluate the quality of the classification using the Evaluation module.

While email recipients may have historically viewed spam as nothing more than an annoying intrusion, unwanted advertising or a waste of time, they now commonly associate it with complex and potential threats to their online security, integrity and trustworthiness [26]. Approximately 50% to 85% of the world's daily email traffic is now generated by spam [6], although spam is not exclusive to email and can also be found in other contexts such as social networks [27]. The negative impact of spam has resulted in billions of dollars of economic loss every year. Several proposed spam detection techniques have been developed to determine the authenticity of the emails [6,10,28]. To evaluate the

performance of the filters, various corpus, measures and evaluation methods have been published [6,10]. Although the evaluation of spam filters seems to be quite consolidated from an academic point of view, the current methods do not simulate the real environment correctly. Among the factors that are often not taken into account when evaluating spam filters are the combination in the stream of email clusters with different characteristics (e.g., social networking, marketing and informational), the arrival frequency of the emails to be filtered, the size of the emails, the number of emails per arrival, and the obfuscation of the email body by spammers.

3.1. Materials and Methods

In order to perform an analysis of the features included in the evaluation framework, the Spam Email Detection case study was performed in the Spam Email Detection scenario. The case study focuses on running simulations to analyse the performance behaviour with respect to the factors involved in the generation of the email stream. The LearningAntiSpamServer classifier is initially trained on 500 spam and 500 ham mails from the SpamAssassin corpus [25] in all simulation runs.

In the case study the factors considered for the simulation are Message Send Rate (MSR), Message Batch Size (MBS), Message Size (MS), Message Obfuscation (MO) and Latency (LAT). The levels for each of these factors are MSR: 1-constant or 2-random; MBS: 1-constant or 2-random; MS: 1-Without limit or 2-With lower limit; MO: 1-Without obfuscation or 2-With obfuscation; and LAT: 1-Null, 2-Intermediate or 3-Extreme. Spammer (SP), Social Network (SNP), Personal (PP), Marketing (MP) and Informational (IP) profiles are modelled for the spam detection environment; however, the proposed solution allows the modelling of other profiles for this and other message analysis contexts. These profiles form clusters in which you can group different types of emails that occur in real-world scenarios. The characterisation of the email corpus used by each profile modelled for the spam detection environment is shown in Table 1.

Table 1. Characterisation of email records by profile.

Profile	Source	Instances	Spam	Ham
Spammer	Personal email accounts	150	150	0
Social Network	Personal email accounts	300	300	0
Personal	Personal email accounts	450	425	25
Marketing	Personal email accounts	500	500	0
Informational	Personal email accounts	500	240	260

In order to determine the constant values of the message send rate and message batch size factors to be used in the experiment, a small study was conducted with 30 university students. These students were asked how many emails they had received per hour during the day from sources associated with the defined profiles. In addition, the category other sources (Others) were included for those emails that were not included in the profiles. From the data, we obtained the average number of emails received per hour for each of the sources, which is shown in Table 2.

Table 2. Average number of emails received per hour.

	SNP	IP	MP	PP	SP	Others
Mean	30	14	36	2	6	1

As can be seen, social networks and news media have the highest average email traffic. Taking into account the data collected, the following values of the levels were tested in the simulation for the experiment with the highest number of emails in the shortest time:

- Message send rate constant: 5 min.

- Message batch size constant: The average number of emails received per hour from each source. In the case of source Others it was not taken into account as it was not covered by the profiles.

In addition, the lower limit value of the MS factor was chosen to be 10 KB.

Obfuscation, which is applied to the body of the mail, models one of the behaviours of a spammer and is only applied to mails with a real spam class. The time variation in the sending of training instances aims to evaluate the quality of the response as the arrival time of the [instance, class] pair increases. In the experiment, the time variation has two possible values: higher frequency (sending training instances with a random frequency between [0:1] min) and lower frequency (sending training instances with a random frequency between [5:20] min). This variable is a fundamental aspect for semi-supervised learning solutions. The duration of each of the simulations will be 1 h. Given the factors and their levels, Table 3 represents the experimental units in arrays, where the value is the treatment given to the factor. For example, the value 2 in *MO* means that the treatment for the variable "Message Obfuscation" will be "With Obfuscation". In addition, changes from one experimental unit to another are highlighted.

Table 3. Description of the experimental units.

	Unit Number												
Factors	**01**	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**
MSR	1	2	1	1	1	1	2	1	1	1	1	1	1
MBS	1	1	2	1	1	1	1	2	1	1	1	1	1
MS	1	1	1	2	1	1	1	1	2	1	1	2	1
MO	1	1	1	1	2	1	1	1	1	2	1	1	2
LAT	1	1	1	1	1	2	2	2	2	2	3	3	3

The combination of the extreme level of the Latency factor with the random levels of the Message Send Rate and Message Batch Size factors was not included in these experimental units. These combinations were not included on the assumption that by not learning during classification, the frequency of mail arrivals or the amount of mail received by the classifiers would not affect the quality of classification and would be similar to using the constant level.

3.2. Analysis of the Results

The aim of this section is to analyse the results obtained by classifying the generated email streams into spam and ham. Based on the analysis of the traces generated by the 13 simulations, the email streams are characterised in Table 4.

Figures 3 and 4 show the results obtained for accuracy, precision and recall for each experimental unit.

In Figure 4, it can be seen that as latency decreases, so do the values of all the measures. This suggests an influence of latency on the quality of results, which may be due to a low ability to adapt to possible concept drift with few training instances of the used processor and/or message classifier. A variation of this may be the use of semi-supervised or unsupervised learning approaches to learn from unlabelled instances. On the other hand, in Experimental Units 5, 10 and 13, there is a significant decrease that coincides with the introduction of obfuscation in the emails, suggesting its significant influence on the classification quality. One of the reasons for this reduction in the quality of results may be that terms that the user recognises as having similar or the same meaning, even if they have different spellings, are not recognised by the processor used and are treated as different terms. In addition, the other three factors have a similar level of influence, regardless of the level of sending training instances. For a better understanding of the results, the experimental units are divided into the three levels of the Latency factor as shown in Figure 4 and detailed below:

- Experimental Units 1–5: Null.
- Experimental Units 6–10: Intermediate.
- Experimental Units 11–13: Extreme.

Table 4. Characterisation of simulated email streams.

	Profiles					Training	Total
Stream	**SNP**	**IP**	**MP**	**PP**	**SP**	**Instances**	**Emails**
S01	360	168	432	24	72	1284	2340
S02	270	168	324	18	72	2126	2978
S03	57	168	96	42	72	2713	3148
S04	360	168	432	24	72	905	1961
S05	360	168	432	24	72	1388	2444
S06	360	168	432	24	72	410	1466
S07	390	168	468	26	72	321	1445
S08	48	168	74	56	72	115	533
S09	360	168	432	24	72	86	1142
S10	360	168	432	24	72	529	1585
S11	360	168	432	24	72	-	1056
S12	360	168	432	24	72	-	1056
S13	360	168	432	24	72	-	1056
Total emails	4005	2184	4850	358	936	9877	

Figure 3. Heat map of general results of the experiment.

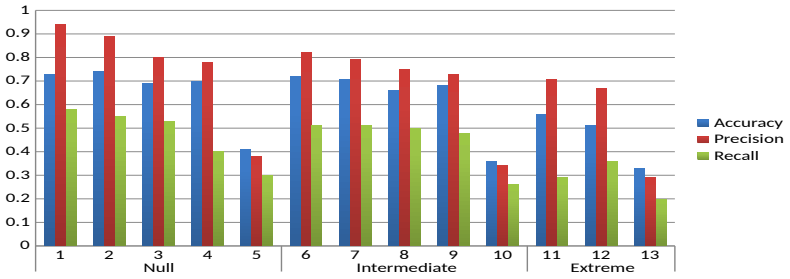

Figure 4. Graphical representation of the overall results of the experiment.

To better see the influence of the factors for each level of the Latency factor, separate analyses are performed. Figure 5 illustrates the behaviour of the quality measures for the experimental units corresponding to the null level.

The first experimental unit is taken as the baseline, as none of the first four factors are changed. In almost all cases there is a tendency for the quality of the result to decrease with respect to the baseline, except for accuracy, where Factor 1 (Experimental Unit 2) shows a slight increase. On the other hand, for most measures, the order of influence of the factors is Factor 4, 3, 2 and 1, with Factor 4 (obfuscation of the body of the mail) having the most significant influence. This order is not true for the accuracy of Factors 2 and 3 (Experimental Units 3 and 4, respectively).

Figure 6 illustrates the behaviour of the quality measures for the experimental units corresponding to the intermediate level.

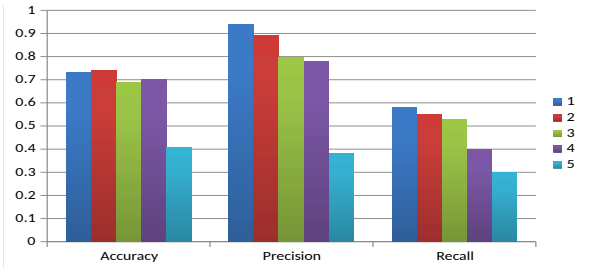

Figure 5. Graphical representation of the null level latency results.

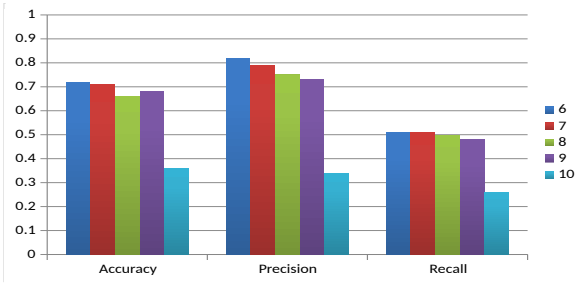

Figure 6. Graphical representation of the intermediate level latency results.

The sixth experimental unit is taken as the baseline, as none of the first four factors are changed. In almost all cases there is a tendency for the quality of the result to decrease with respect to the baseline, except for recall, where Factor 1 (Experimental Unit 7) is the same. On the other hand, for most of the measures, the order of influence of the factors is Factor 4, 3, 2 and 1, with Factor 4 (obfuscation of the body of the mail) having the most significant influence. Factors 2 and 3 (Experimental Units 8 and 9, respectively) do not follow this order for accuracy.

Figure 7 illustrates the behaviour of the quality measures for the experimental units corresponding to the extreme level.

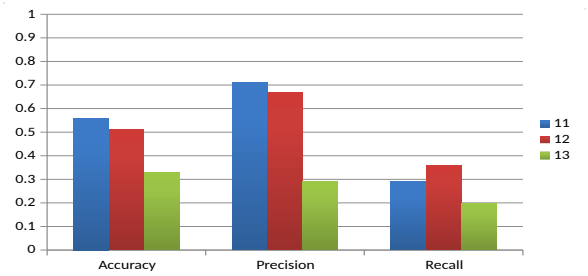

Figure 7. Graphical representation of the extreme level latency results.

The eleventh experimental unit is taken as the baseline, as none of the first four factors are changed. In almost all cases there is a tendency for the quality of the result to decrease with respect to the baseline, except for recall, where Factor 3 (Experimental Unit 12) shows an increase with respect to the baseline. On the other hand, for all measures, the order of influence of the factors is Factor 4 and 3, with Factor 4 (obfuscation of the body of the mail) having the most significant influence. Overall, the experimental results show that in most cases the second levels of the identified factors tend to decrease the classification quality values. Factor 1, however, exceeds the baseline in one case and is equal to the baseline in another, and Factor 3 exceeds the baseline in one case. On the other hand, the order of influence of the first four factors is almost always Factor 4, 3, 2 and 1. From the results obtained, it could be said that the factors with the most notable influence are Factors 4 and 5 (message obfuscation and latency, respectively).

4. Discussion

Current approaches focus on evaluating classification solutions with message streams generated from a cross-validation strategy or with different message corpora. This reduces the ability to adequately model dynamic environments with varying latency and limits the representation of different behaviours of message sources such as spammers. In contrast, the present proposal generates the message streams following a profiling approach of the message sending sources together with the use of simulation techniques and queueing theory. This gives the possibility to model the different behaviours that different sources may have, be it different spammers with different adversarial attack strategies, press media or social network sources. Each of these profile instances generates its own stream of messages, which are then combined into the main stream, which is closer to the real stream. For example, in the study case presented, it was found that the evaluated classification solution degrades in its quality results as the latency increases and when message obfuscation strategies are used by spammers. This suggests that the classification solution should be improved to better adapt to message streams where training message latency increases and spammers use message obfuscation strategies. A possible solution to the latency problem could be the use of a semi-supervised or unsupervised classification approach and, for obfuscated messages, the improvement of the text processor used by means of a term similarity approach. However, these sources may have social and proactive behaviours that influence the generation of the stream and are not covered by the proposed solution. An example of this is the spammers themselves, who may proactively vary their attack strategy so that their target cannot evade them, sometimes forming communities of spammers. Furthermore, the proposed solution only considers obfuscation of the message body as an adversarial attack strategy for this type of source, and there are others.

On the other hand, in the developed study case, the latency in sending training instances and the obfuscation of the body of the messages were identified as the parameters with the greatest impact on the quality of the classification. This case study demonstrated the performance of the proposed framework. However, the dataset used for the evaluation was small, as were the processing and classification solutions used, suggesting a larger-scale evaluation at the level of the volume of messages used and a wider range of classifiers and text processors. Furthermore, the framework was only evaluated in the context of spam detection, although it can be used in other contexts, such as news analysis, where sources are modelled as thematic profiles that may constitute digital news media.

5. Conclusions

In this work, we present a framework based on message stream simulation to evaluate solutions of classification. The use of the queueing theory in the modelling of the user profiles contributed a bigger formalism to the proposed solution. The incorporation of randomness in some of the events of the simulation allowed approximating the generation of the message stream to a real scenario. The evaluation module provides the ability to apply the evaluation measures to analyse the behaviour of the classifiers under the same input conditions. The case study demonstrates an evaluation design that allows identifying the factors present in real contexts that most influence the quality of classification solutions, and thus knowing how to adapt the solution. For the evaluated solution, the factors that most influenced the quality were message obfuscation and latency. Future work is planned to perform a robustness or sensitivity analysis to better understand how changes in simulation parameters affect the reliability of the proposed framework; to develop new test cases and/or scenarios in contexts such as news analysis, unanswered message identification, social media message classification, and others that demonstrate greater versatility and applicability of the proposed framework to a wider range of message classification challenges; to use the proposed framework to evaluate and compare solutions from the literature for semi-supervised text classification in intermediate and extreme latency scenarios; and to make social modelling of message delivery sources to identify

social and proactive behaviours, which can then be implemented using an intelligent multi-agent approach.

Author Contributions: Conceptualization, W.H.-M. and M.M.-E.; methodology, F.M.-P. and J.V.B.M.; validation, M.M.-E., F.M.-P. and J.V.B.M.; formal analysis, W.H.-M., F.M.-P. and M.M.-E; writing—original draft preparation, W.H.-M.; writing—review and editing, I.L.F. and J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available in http://mlkd.csd.auth.gr/datasets.html.

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

References

1. Bularca, M.; Nechita, F.; Sargu, L.; Motoi, G.; Otovescu, A.; Coman, C. Looking for the Sustainability Messages of European Universities' Social Media Communication during the COVID-19 Pandemic. *Sustainability* **2022**, *14*, 1554. [CrossRef]
2. Bui, Q.; Lyytinen, K. Aligning adoption messages with audiences? priorities: A mixed-methods study of the diffusion of enterprise architecture among the US state governments. *Inf. Organ.* **2022**, *32*, 100423. [CrossRef]
3. Hemker, S.; Herrando, C.; Constantinides, E. The Transformation of Data Marketing: How an Ethical Lens on Consumer Data Collection Shapes the Future of Marketing. *Sustainability* **2021**, *13*, 11208. [CrossRef]
4. Anastasiei, B.; Dospinescu, N.; Dospinescu, O. The impact of social media peer communication on customer behaviour—Evidence from Romania. *Argum. Oecon.* **2022**, *1*, 247–264. [CrossRef]
5. Zheng, X.; Li, P.; Wu, X. Data Stream Classification Based on Extreme Learning Machine: Review. *Big Data Res.* **2022**, *30*, 100356. [CrossRef]
6. 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. Rev.* **2023**, *56*, 1145–1173. [CrossRef]
7. Marrs, G.; Hickey, R.; Black, M. The impact of latency on online classification learning with concept drift. In Proceedings of the Knowledge Science, Engineering and Management 2010 (KSEM 2010), Belfast, Northern Ireland, UK, 1–3 September 2010; Bi, Y., Williams, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6291, pp. 459–469. [CrossRef]
8. Souza, V.; Pinho, T.; Batista, G. Evaluating Stream Classifiers with Delayed Labels Information. In Proceedings of the 7th Brazilian Conference on Intelligent Systems (BRACIS), Sao Paulo, Brazil, 22–25 October 2018; pp. 408–413. [CrossRef]
9. Biggio, B.; Roli, F. Wild Patterns: Ten Years after the Rise of Adversarial Machine Learning. *Pattern Recogn.* **2018**, *84*, 317–331. [CrossRef]
10. Dada, E.; Bassi, J.; Chiroma, H.; Abdulhamid, S.; Adetunmbi, A.; Ajibuwa, O. Machine learning for email spam filtering: Review, approaches and open research problems. *Heliyon* **2019**, *5*, e01802. [CrossRef] [PubMed]
11. Pérez-Díaz, N.; Ruano-Ordás, D.; Fdez-Riverola, F.; Méndez, J. SDAI: An integral evaluation methodology for content-based spam filtering mode. *Expert Syst. Appl.* **2012**, *39*, 12487–12500. [CrossRef]
12. Nelson, B.; Biggio, B.; Laskov, P. Understanding the Risk Factors of Learning in Adversarial Environments. In Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence; AISec '11, Chicago, IL, USA, 21 October 2011; ACM: New York, NY, USA, 2011; pp. 87–92. [CrossRef]
13. Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; Roli, F. Evasion Attacks against Machine Learning at Test Time. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, Prague, Czech Republic, 23–27 September 2013; Blockeel, H., Kersting, K., Nijssen, S., Železný, F., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 387–402.
14. Jubatus: Distributed Online Machine Learning Framework. Available online: http://jubat.us/en/ (accessed on 16 January 2024).
15. Bifet, A.; Holmes, G.; Kirkby, R.; Pfahringer, B. MOA: Massive Online Analysis. *J. Mach. Learn. Res.* **2010**, *11*, 1601–1604.
16. Clever, L.; Pohl, J.; Bossek, J.; Kerschke, P.; Trautmann, H. Process-Oriented Stream Classification Pipeline: A Literature Review. *Appl. Sci.* **2022**, *12*, 9094. [CrossRef]
17. Gartner IT Glossary. Frameworks. 2021. Available online: https://www.gartner.com/en/information-technology/glossary/framework (accessed on 5 September 2022).
18. Apache Software Foundation. *Apache Spark—Unified Analytics Engine for Big Data*; Apache Software Foundation: Forest Hill, MD, USA, 2021.
19. Pérez, F.M.; Fonseca, I.L.; Martínez, J.V.B.; Maciá-Fiteni, A. Distributed Architecture for an Elderly Accompaniment Service Based on IoT Devices, AI, and Cloud Services. *IEEE MultiMedia* **2023**, *30*, 17–27. [CrossRef]
20. Hiller, F.; Lieberman, G. *Introduction to Operations Research; Raghothaman Srinivasan*; McGraw-Hill Science: New York, NY, USA, 2010; Chapter Simulation, pp. 934–990.
21. Hiller, F.; Lieberman, G. *Introduction to Operations Research; Raghothaman Srinivasan*; McGraw-Hill Science: New York, NY, USA, 2010; Chapter Queueing Theory, pp. 759–827.

22. García, S.; Luengo, J.; Herrera, F. *Data Preprocessing in Data Mining*, 1st ed.; Springer: Cham, Switzerland, 2015. [CrossRef]
23. Pyle, D. *Data Preparation for Data Mining*, 1st ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1999.
24. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [CrossRef]
25. Katakis, I.; Tsoumakas, G.; Banos, E.; Bassiliades, N.; Vlahavas, I. An adaptive personalized news dissemination system. *J. Intell. Inf. Syst.* **2009**, *32*, 191–212. [CrossRef]
26. Gangavarapu, T.; Jaidhar, C.; Chanduka, B. Applicability of machine learning in spam and phishing email filtering: Review and approaches. *Artif. Intell. Rev.* **2020**, *53*, 5019–5081. [CrossRef]
27. Ali, S.; Islam, N.; Rauf, A.; Din, I.; Guizani, M.; Rodrigues, J. Privacy and Security Issues in Online Social Networks. *Future Internet* **2018**, *10*, 114. [CrossRef]
28. Yang, H.; Liu, Q.; Zhou, S.; Luo, Y. A Spam Filtering Method Based on Multi-Modal Fusion. *Appl. Sci.* **2019**, *9*, 1152. [CrossRef]