Journal of Universal Computer Science, vol. 27, no. 4 (2021), 364-386
submitted: 28/1/2020, accepted: 12/3/2021, appeared: 28/4/2021

Spam Detection Based on Feature Evolution to Deal with Concept Drift

Marcia Henke
(Federal University of Santa Maria (UFSM) - Industrial Technical College of Santa Maria (CTISM), Santa Maria, RS, Brazil
https://orcid.org/0000-0003-1437-9309, henke@redes.ufsm.br)

Eulanda Santos
(Federal University of Amazonas (UFAM) - Computer Institute (ICOMP), Manaus, AM, Brazil
https://orcid.org/0000-0002-5671-581X, emsantos@icomp.ufam.edu.br)

Eduardo Souto
(Federal University of Amazonas (UFAM) - Computer Institute (ICOMP), Manaus, AM, Brazil
https://orcid.org/0000-0003-0003-908X, esouto@icomp.ufam.edu.br)

Altair O. Santin
(Pontifical Catholic University of Paraná, Curitiba, PR, Brazil
https://orcid.org/0000-0002-2341-2177, santin@ppgia.pucpr.br)

Abstract: Electronic messages are still considered the most significant tools in business and personal applications due to their low cost and easy access. However, e-mails have become a major problem owing to the high amount of junk mail, named spam, which fill the e-mail boxes of users. Several approaches have been proposed to detect spam, such as filters implemented in e-mail servers and user-based spam message classification mechanisms. A major problem with these approaches is spam detection in the presence of concept drift, especially as a result of changes in features over time. To overcome this problem, this work proposes a new spam detection system based on analyzing the evolution of features. The proposed method is divided into three steps: 1) spam classification model training; 2) concept drift detection; and 3) knowledge transfer learning. The first step generates classification models, as commonly conducted in machine learning. The second step introduces a new strategy to avoid concept drift: SFS (Similarity-based Features Selection) that analyzes the evolution of the features taking into account similarity obtained between the feature vectors extracted from training data and test data. Finally, the third step focuses on the following questions: what, how, and when to transfer acquired knowledge? The proposed method is evaluated using two public datasets. The results of the experiments show that it is possible to infer a threshold to detect changes (drift) in order to ensure that the spam classification model is updated through knowledge transfer. Moreover, our anomaly detection system is able to perform spam classification and concept drift detection as two parallel and independent tasks.

Keywords: Computer Security Network, Machine Learning, Concept Drift
Categories: I.2.1, I.5.1, I.5.4
DOI: 10.3897/jucs.66284

1 Introduction

The continuous increase in the number of devices connected to the Internet has encouraged cyber criminals in their sophistication of software and malicious tools to carry out computer attacks and intrusions. These cyber-attacks are generally mass-spam messages and continue to be one of the main vectors for the propagation of fraudulent activities such as phishing and pharming, denial of service attacks, virus proliferation, worms, and trojans, among other malicious codes.The reason behind this is that the use of emails has its advantages (for the attackers): the email content can use social engineering techniques to lead the user not only to download the malicious file, but also to initialize it. Reports from main security industries show an increase in the number of unsolicited emails containing malicious attachments, which include malicious URLs, classic executable EXE files, and programs written in Visual Basic Script and JavaScript (VBS and JS files, JAR, WSF, and others) [Kaspersky, 2020] [ISTR, 2018]. Therefore, observing spam attacks as potential cybercrime can help in the prevention, by observing changes in attack methods, including the type of malicious code and the presence of criminal networks (i.e. botnets) [Kaspersky, 2020] [Alazab and Broadhurts, 2016].

Given this scenario, email service providers and researchers on security have invested time and resources searching for more effective solutions to respond to spamming strategies proposed by spammers. However, a great part of the existing detecting models is generated based on a set of features observed in a given moment. For instance, the most basic process for message filtering creates a unique hash value (i.e., a message digest) for each known spam message (signature-based systems) [Kocz, 2004]. Because it is a text classification problem, such an approach has been evaded by simple techniques for modifying spam texts, for instance, the use of keywords with different spelling, such as “free” and “Viagra” spelled as “fr33”, “f.r.e.e”, “v.i.a.g.r.a”, “vlagra” and “vi@gr@”. Even spam detection systems (also known as outlier detection or novelty detection), which try to identify text patterns which do not conform to the expected text of an email, are created based on classification models generated from a set of features in a time instant t. However, the behavior and features of the spam attacks treated by these tools change with time. Studies carried out in [Delany, 2005] [Guzella and Caminhas, 2009] [Hayat et al., 2010] [Su and Shen, 2008] show that a majority of applications for email filtering and their classification as spam or legitimate are deficient when the set of data that represents the problem becomes outdated or no longer representative.

However, in machine learning literature it is already possible to find solution proposals for problems which evolve with time, as is the case with most attacks [Blazieri and Bryl, 2009] [Fdez-Riverola and Iglesias, 2007] [koren, 2009] [Caruana and Li, 2012]. These solutions can be divided into two groups: implicit and explicit. In implicit methods, the database is updated and the machine learning algorithm is retrained at pre-defined time intervals t. Therefore, these are methods which present high cost, even though they are dynamic. In explicit methods, their main aim is to detect change occurrences (drifts) in order to update the classification model. However, the detection mechanism is usually based on monitoring the classifier’s error rate. Thus, the classifier’s error rate needs to be abruptly increased for the changes to be detected. This situation certainly implies false positives and false negatives which will be emitted by the system before the change is detected.

In an attempt to minimize the effects of some of these deficiencies, we propose a framework for spam attack detection based on the analysis of changes in data distribution over time (ADDrift), whose process for monitoring the change in features is based on the concept drift detection technique and the update of the classification model is based on

transfer learning [Gama, 2004] [Taylor and Stone, 2009]. The process of concept drift is used to determine the moment in which a significant change in the feature space occurs, which explicitly indicates a change in the features of spam attack. Transfer learning techniques are applied in order to maintain and update the knowledge base about the attacks. Furthermore, ADDrift presents a modular architecture composed of three well-defined and independent steps: 1) classification model training; 2) concept drift detection; and 3) transfer learning. This allows the use of different strategies without affecting the functioning of the proposed architecture, and equally being able to be used in the detection of different types of attacks. To deal with change detection, we present a new strategy: SFS (Similarity-based Features Selection), where the change detection process occurs by means of the analysis of similarities between the features spaces extracted from training data (also referred as source domain) and test data (also referred as target domain). Our experimental results indicate that it is possible to infer a threshold for change detection in order to guarantee that the classification model is always updated by means of transfer learning. Therefore, the main contributions of this paper are:

- We propose a framework for spam attack detection based on concept drift detection, which can be easily adapted for other cybersecurity domains. New concept drift strategies can easily be plugged into the framework, preventing spam attack at an early stage.
- A new detection strategy focusing on observing the feature representations provided by source and target domains. This strategy, called SFS, is based on the analysis of similarities, taking into account as relevance criteria the frequency of each feature in features spaces used to train the model (source domain) and test the model (target domain).
- We design approaches from the framework based on support vector machine. Empirical results show that the framework consistently improves baseline algorithms on two real-world data sets of spam mails, highlighting the efficacy of the proposed approach in providing reliable and robust drift detection.

The remainder of this paper is organized as follows. Section 2 presents related work in spamming detection. Section 3 presents the architecture of the proposed framework and describes its main blocks, including the proposed concept drift strategy (SFS). Section 4 describes the experiments and the results obtained. Finally, Section 5 draws the conclusions and presents suggestions for future work..

2 Related Work

Change detection approaches are concerned with maintaining a dynamic control, seeking to perceive changes between the knowledge base and unknown samples. Research focused on this context has been carried out for approximately two decades. Several methods in the literature have been proposed with a focus on explicit and implicit change detectors [Su and Shen, 2008]. Explicit and implicit change detection methods can be based on probability distribution [Ebner et al., 2007] [Ganti and Ramakrishnan, 2002], the relevance of features [Delany et al., 2006] [Zhang et al., 2014], and the accuracy of the classification model [Gama, 2004] [Goncalves, 2014]. The papers presented in this section address explicit and implicit detection based on the classification model accuracy rate, probability distribution, and the relevance of features on labeled databases.

As a way of detecting the need to generate a new classification model, some authors use reduction of the accuracy rate of the classifier as a flag, as others adopt predefined time intervals (time windows) to generate a new classification model. But both approaches are based on features that represent a spam or legitimate email to form the new training set for the classification model.

2.1 Implicit Change Detection

These are methods that present the construction of a new classification model according to a time interval t to re-train the classification model, regardless of whether or not a change in the data has been identified. Two classic solutions in the area of concept drift research are: STAGGER, proposed by Schlimmer and Granger [Zhou, 2010a] and FLORA, which is a family of algorithms, proposed by Widmer and Kubat [Widmer and Kubat, 1996].

The FLORA framework [Widmer and Kubat, 1996] uses sliding windows for learning a binary classifier. More recent samples are selected to compose the new learning, while older samples are excluded. The framework dynamically uses three rules: one rule covers only positive examples, (in spam filters, for instance, positive samples represent spam e-mails); another rule covers only negative examples (legitimate emails or not spam); and the third rule is a combination considering both positive and negative samples. FLORA has been enhanced with some improvements, giving rise to FLORA2, FLORA3 and FLORA4. FLORA2 is able to adapt the size of the window. FLORA3 improves adaptability and the ability to reactivate obsolete rules. Finally, FLORA4 includes better strategies for noise treatment. The approach used by the FLORA framework to overcome the problem of concept drift is focused on using sliding windows that always consider the most recent samples for learning the classification model.

The STAGGER system [Zhou, 2010a] seeks to describe data by the simplest set of features. Each representation describing the samples has an initial pair of weights. The system involves a projection process that combines the representation of the concept distributed in relation to the new samples. For each characterization, one of the weights is used if the characterization is similar to the sample, while the other weight is used if it is not similar to the sample. The weights are adjusted by the evaluation process, which keeps track of the number of times each characterization was similar or not to the samples. Evaluation is the process of determining the effectiveness of the descriptions of internal concepts. In STAGGER, each representation in the concept description is continuously evaluated, adapting its weights. This assessment reflects learning discoveries, based on the number of times each representation was successful or failed with a prediction. From the weight combinations, concept drift is identified and the change to the new learning is bypassed.

Jackowski [Jackowski, 2014] presents a set of algorithms for Evolutionary Adapted Ensemble - EAE. The author works with recurring change, so the classifier that presents low precision is not discarded, and is only replaced by another one, being able at a later moment to return to the set of algorithms. Public databases are used: one is artificially generated from the UCI Machine Learning Repository, with data on electricity demand, and another is from the SpamAssassin project spam. The updating of the set of classifiers is defined by a fitness function that considers the best classifier which presents the best precision to compose the new set, just as the generation of a new training repository considers all evolutionary algorithms: population initialization, mutation operators, and crossover. The update is done every time the system identifies a new feature context, generating a new classifier and adding it to the classifier pool.

Hayat et al. [Haury et al., 2011] propose a language-based spam filtering system. As the content of the email changes over time, then the trained template becomes obsolete and unable to detect new spam. The proposed approach uses language modeling techniques to represent the distribution of content blocks. A language model indicates a probability for each word in the document. Each word in each block will have a weight. This weight represents the importance of the word in the block. It is important to note that probabilities are based on the content of the block, so since any change affects the distribution of the data, this situation can be considered a change in the content through the language model employed. The way the authors perform the change detection is by the comparison of the accuracy rate of the classification model with a threshold, and the generation of a new classification model.

Delany et al. [Delany et al., 2006] present E-mail Classification Using Examples (ECUE) based on Case-Based Reasoning (CBR) as a solution to the spam filter problem. CBR is considered a lazy learning technique. Examples of training data represent case based for a CBR. In ECUE, each email is a case represented as a feature vector. The approach for detecting changes used is the selection of instances, that is, the case base needs to be updated to include new types of spam and ham e-mail over time. Given the volume of emails that a single user receives per week, there is a need to actively manage training data. Therefore, e-mails incorrectly classified as spam or ham are inserted into the training of the model as a way to update the knowledge base.

2.2 Explicit Change Detection

These are methods that consider the accuracy of the classifier as an indicator of the need to update the classification model, that is, the change is identified from the generalization loss of the classification model. This type of method is the most popular in the literature.

With a hybrid system combining feature selection and a decision tree for spam detection, authors Zhang et al. [Zhang et al., 2014] aim to reduce the false positive error rate using a feature selection method divided into two groups: a filter and a wrapper strategy. The filter considers the content of the information through an objective function evaluating the measures of distance, correlation, statistical dependence, and information theory. The wrapper defines an objective function with a standard classifier, which selects a subset of features through the prediction measure. Thus, they extract the selected features from 6,000 emails collected in 2012 available in the UCI machine learning repository (http: // archive.ics.uci.edu/ml/datasets/Spambase). In order to overcome the disadvantages presented by the wrapper method, which were the slowness in the extraction and the lack of generalization, the global optimization and cross validation methods, respectively, were used for both situations. The authors considered the false positive rate to update the new feature set.

Alqatawna et al. [Alqatawna et al., 2015] propose a framework to improve spam detection. The authors use the analysis of features, but only of the spam class. Malicious spam (emails with attachments) and typical spam (emails without attachments), are named by the authors. To extract the features, the header, body, and attachments of the emails are examined using software in Java. In order to develop a spam detection model, four classification algorithms are used: the C4.5 algorithm, the Multilayer Perceptron Neural Network (MLP), Naïve Bayes, and Random Forest. The four classification models are evaluated through precision, accuracy, and recall metrics. As the classifiers decrease their performance, features are updated.

Gonçalves et al.[Goncalves, 2014] present a comparison among five different concept drift methods: PHD (Page-Hinkley Test), ECDD (Concept Drift Detection) based on

exponentially weighted moving, Average STEPD (Statistical Test of Equal Proportions), DOF (Degree of Drift) and Paired Learners, on Sine (a dataset containing abrupt drift) and Mixed (a dataset with gradual drift). To measure the performance of the compared methods, statistical measures are used, such as: drift detection rate, standard deviation, false alarm rate, and Mahalanobis distance. The objective of the authors is to compare various types of concept drift and analyze how the investigated concept drift detectors behave in the presence of different abrupt and gradual velocities of change. The updating of the classification models occurs changing the configuration of the parameters of each model.

Wenerstrom and Giraud-carrier [Wenerstrom and Giraud-carrier, 2006] present a set of classifiers for change detection using Feature Adaptive Ensemble (FAE). FAE uses Naïve Bayes classifiers incrementally, that is, classifiers adapt to each new added training instance. In addition, each classifier in FAE is an expert in a subset of features and new features can be added at any time, facilitating the task of detecting contextual change. FAE is able to adapt well to changes because it uses several classifiers of different ages and makes decisions about the removal of a classifier from the set based on classifiers individual precision. As the classification model reduces the performance of its classification task based on the accuracy rate, a new classifier is generated to compose the set of classifiers.

Tsymbal et al. [Tsymbal et al., 2008] also use a classifier ensemble technique to handling change detection based on the samples. In a dynamic integration, each base classifier receives a weight proportional to its local precision in the vicinity of the current test instance, instead of using the overall classification accuracy as in traditional weight voting. Most ensemble of classifiers approaches use some criteria to dynamically delete, re-enable, or create new members. These criteria are usually based on the consistency of the model over current data. In addition, instance-based learning is used in order to predict the local performance of base models in a dataset. Therefore, Tsymbal et al. [Tsymbal et al., 2008] use three dynamic techniques based on the same local performance estimation: dynamic selection, dynamic voting, and dynamic voting with selection. However, the authors consider the local accuracy of the classifier members to determine when to generate a new classification model.

Gu-Hsin et al. [Lai et al., 2009] present a collaborative framework for generating rules focused on dealing with drift detection. The framework is built on a genetic algorithm and reinforcement learning using rules. If a rule is rarely used, its weight decreases. Wrong classifications also lead to reduce rules' weight. On the one side, if the rules are highly employed, then these rules are more often used and perform better. On the other, rules rarely employed are assumed to be out of scope or inaccurate. As a consequence, email servers only maintain or allow highly employed rules in order to improve their detection rates and performance. The changes are circumvented by the use of the rules, that is, the rules change as their weight change due to the behavior of the spam attacks. However, this change is defined by the performance of the classifier, i.e. the rules increment or decrement their weights according to the improvement or reduction of the performance of the classification model.

Idre Zliobataite [Zliobaite, 2009]presents an application which performs relevant training samples selection based on the similarity to target samples. Similarities considering space and time are combined. The algorithm determines the size of an optimal training set based on the performance of the classification model. The author uses k-NN (k Nearest Neighbors) for the classification task. Each sample related to the concept drift is identified according to the similarity of space and time and is included in the training dataset as new knowledge. Therefore, the added samples are all those that show

similarity to the training samples. The author works with two algorithms: fixed time window and variable time window.

Lu et al. [Lu et al., 2014] present a method for detecting change in a case-based reasoning system. Instead of measuring the distribution of the current case, a new reliability model is introduced that detects differences through changes in this reliability. Therefore, the method does not require prior knowledge of the distribution of cases and provides statistical guarantees on the reliability of detected changes, as well as significant and quantitative descriptions of these changes. Instead of observing the feature space, reliability measures are observed to detect change. The updating of the classification model is carried out with the detection of cases far from the reliability models.

Taking into account works described in this section, it is possible to notice that most applications employ the approach of selecting instances to cope with changes [Guzella and Caminhas, 2009], [Su and Shen, 2008]. However, this strategy can be costly when the amount of arriving data is high, for instance in spam categorization, since all these data must be labeled. Among the solutions that first try to detect the change in order to adapt the classifier, most of them detect changes when the accuracy rate decreases [Lai et al., 2009] [Zliobaite, 2009] [Zhang et al., 2014]. However, this type of strategy presents a serious problem because it depends on a significant drop in the accuracy rate to identify the occurrence of a change. Thus, in spam detection, phishing, or malware problems, for example, when a fall in the accuracy rate of the classifier is detected, the user has already suffered some damage or loss. Despite this disadvantage, several methods for detecting change have been presented in the literature [Gama, 2004], [Widmer and Kubat, 1996], [Goncalves, 2014] to monitor errors of online classifiers. Most methods rely on the error rate, recall, and accuracy rate of the classifier to measure change detection.

With a differentiated proposal, this work seeks to detect changes using an explicit detection method based on the relevance of the features through a similarity threshold as a sign of the need to generate a new classification model, seeking to keep the classifier always up to date. Our methods are able to identify the need to generate a new classification model without using the drop of performance as the change flag.

3 Framework for Spam Detection Based on Changes in Data Distribution

In this section, we present a framework for attack detection focused on dealing with concept drift, named ADDrif. Specifically, ADDrif analyses changes in data distribution and then updates the classification models when these changes are considered significant. The new knowledge discovered (e.g. adding or removing a feature, changing the value ranges of a feature) is obtained from the comparison of similarity between the feature vectors (instances of the source domain) used to generate the classifier with the feature vectors extracted from the test phase (instances of the target domain). As detailed in Section 3.1, whenever the similarity between the instances of the source domain and a new instance exceeds a certain threshold, that instance is stored in a repository that will compose the new source domain. Maintaining the source domain updated is important because if the changes are significantly sufficient, it is possible to easily reconstruct the classifier to keep a stable hit rate with minimal human interference. An overview of the framework is shown in Figure 1.

The classification model training and construction step consists of finding a model which describes and points out data classes [Gao et al., 2008]. As in the majority of pattern recognition systems, the model is mainly constructed based on the analysis of a set of

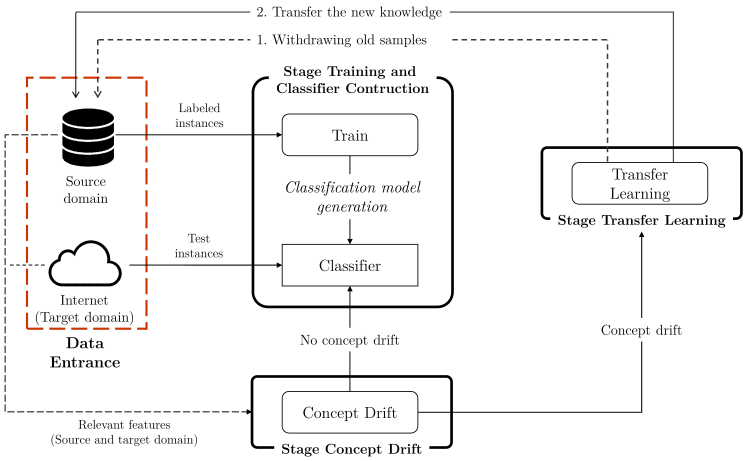

Figure 1: Overview of the ADDrift framework. ADDrif is divided into three stages: i) training and construction of the prediction model; ii) concept drift detection; iii) and a phase of transfer learning.

training data. Next, this model is used to examine the features of a new instance (whose class is unknown) and assign it to a class. This step can be performed using any base classifier such as Decision Tree, Support Vector Machines, k-Nearest Neighbor, Neural Networks, Naïve Bayes, etc., without the need to change the implementation or setting of this framework. Once the classification model is constructed, the change detection phase is initiated, which monitors new data samples with the aim of identifying when changes in the data distribution used for generating the classification model occur. However, detecting changes, even if the classification model's performance is compromised, is still a challenge, given that not all changes in data distribution will degrade the classifier's performance [Gao et al., 2008] [Gama and Kosina, 2014].

To handle this problem, this paper proposes a new strategy for data change detection, called Similarity-based Features Selection (SFS). In SFS, change detection is observed by taking into account the similarity among feature vectors corresponding to the source domain (training data) and the target domain (test data). The last phase of the ADDrift framework corresponds to the reaction step, where knowledge data extracted during the change detection phase are employed to help the learning model to obtain the best performance. In the SFS strategy, transfer learning is conducted by means of similarity measure, which indicates how closely the model elements are related to the new elements (samples).

3.1 Similarity Based Features Selection (SFS)

The SFS change detection strategy aims at identifying behavioral changes in a given attack by observing the representation of the features that compose the source and target domains. This observation is based on a similarity analysis of the features vectors that represent the problem in the moment of the classification model construction compared to the following moment.

Figure 2 presents the proposed change detection strategy. A domain D is defined by two parts, a feature space χ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \chi$ and x_i is the i^{th} feature and n is the number of features in X, χ is the space of all possible feature vectors, and X is a particular learning sample. For a given domain D, a task τ is defined by two parts, a label space γ, and a predictive function $f(\cdot)$, which is learned from the feature vector and label pairs $\{X_i, y_i\}$, where $y_i \in \gamma$. From the definitions above, D_s is defined as the source domain data $D_s = \{(X_{s1}, y_{s1}), ..., (X_{sn}, y_{sn})\}$, where X_{si} is the i^{th} data instance of D_s and $y_{si} \in \gamma_s$ is the corresponding class label for X_{si}. The target domain data D_t is defined as $D_t = \{X_{t1}, ..., X_{tm}\}$, where X_{ti} is the i^{th} data instance of D_t.

SFS calculates the similarity between instances in D_s and instances in D_t using a strategy, inspired by problems in the area of information retrieval, which is based on the use of the vector space model proposed by Salton [Salton et al., 1975] [Hamers et al., 1989]. In this strategy, each sample, represented by a features vector, is considered a word. For instance, in problems related to spam attacks, the vector space model establishes a relation between two vectors $\vec{d} = (d_1, d_2, ..., d_n)$ e $\vec{q} = (q_1, q_2, ..., q_n)$ by means of a similarity measure. Such a measure is an attempt to calculate how closely related are the two analyzed vectors.

The equation for calculating the similarity between two vectors $\vec{d}$ e $\vec{q}$ consists of computing the cosine of the angle between them and it is defined as

$$sim(d, q) = cos\theta = \frac{\sum_{i=1}^{n} d_i \cdot q_i}{\sqrt{\sum_{i=1}^{n} d_i^2} \sqrt{\sum_{i=1}^{n} q_i^2}} \quad (1)$$

The similarity calculation may involve a high computational complexity, since the similarity will be calculated between a sample set, which represents the target domain D_t, and the sample set which represents the source domain D_s. For instance, if the source domain is represented by 1000 samples, the similarity calculation will be done for each document of the target domain in relation to the source domain, giving rise to a computational complexity of $1000 \times n$, where n represents the number of samples of the target domain. This computational complexity is also increased due to the high dimensional data commonly observed in problems related to spam attacks.

In order to reduce computational complexity, two approaches are performed in this work: feature selection and instance selection. First, to improve the proposed models' performance and reduce the feature space, we use the information gain feature selection technique [Quinlan J.R., 1992] to choose the k most relevant features of D_s. Information gain (IG) evaluates features according to a reduction in entropy when splitting on a feature. As a result, IG will rank the features to identify those which have the highest IG. Thus, the k highest ranked features are selected. It is important to mention that spam classification is widely known as a problem presenting sparse data [Dada et al., 2019], meaning that zero is assigned to many of the features. Therefore, taking into account that SFS focuses on identifying changes by comparing the representation of the features that compose D_s and D_t, each instance in D_s and D_t is then represented using a k-dimensional feature set. Precisely, each feature value will be either 0 or 1 whether the i^{th} highest ranked feature is present or absent in a instance, respectively. In this way, the similarity measure can be calculated.

In terms of instance selection, we applied the K-means clustering algorithm on the source domain D_s to extract the centroids which best represent the classes. In this way, the computational complexity is minimized during the calculation of the vector space

model, given that the problem is represented by two classes (for instance, spam and non-spam), making the calculation to be $2 \times n$. Thus, the similarity between each new instance X_{ti} and the source domain centroids is used to estimate data changes. The similarity value of the closely similar centroid is then assigned to X_{ti}.

From calculating the similarity between the target and source domain samples, the aim is to define a threshold to infer the change. This threshold is directly related to a tolerance limit supported by the system, that is, to what extent can the similarity point between the two domains be reduced without the occurrence of performance degradation of the classification model? This point can be an indicator to infer a real change in the features. It is important to mention that there are two types of drift: virtual concept drift, i.e. a drift which does not affect the classifier's decisions; and real concept drift - drift that compromises the classifier's performance. Virtual changes are easily detected by means of the similarity threshold used by the SFS strategy. This is due to the fact that the target domain samples, which are not similar (similarity below the threshold) to those of the source domain, are stored in a virtual change repository (transitory). Thus, after the detection of a real change (permanent), these samples are used to update the classifier's knowledge.

However, the identification of a real change is more complex, because it is important to detect drift in such a way that a performance degradation in the classification model does not occur. In this paper, X_{ti} instances dissimilar to instances in the source domain are stored in the virtual change repository during a fixed maximum time t. The parameter t should be defined experimentally.

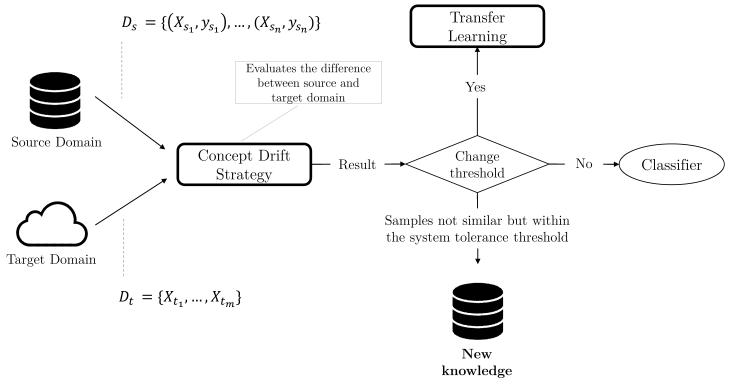

Figure 2: Overview of the SFS change detection strategy.

This model is very flexible; the only premise considered common to all the possibly applied strategies is the action to detect changes. The employed strategy can be modified in accordance with the addressed problem. While t is not reached, the classification task continues to be performed by the original classifier. On the other hand, when a possibly virtual change is detected, samples indicating this change (similarity below threshold) are stored in the virtual change repository. Finally, when a real change is detected (t is reached), the system indicates that a new classification model must be generated. For this, the source domain must be updated by means of the transfer learning step. In this case, all v instances stored in the virtual change repository are used to replace the v

oldest instances of D_s. Finally, a new classifier is trained using the new D_s, then IG is applied and the whole SFS approach is repeated.

4 Experimental Evaluation

This section presents experimental analysis and results of the ADDrift framework on two sets of experiments. First, we investigate how a stable classification model can be maintained over time without any updating. Then, the second set of experiments evaluates the behavior of changes in the features distribution. The goal is to know how much the feature distribution in the source domain is different from that in the target domain. Section 4.3 presents results on such a monitoring strategy considering the variance of feature evolution by observing the level of similarity between relevant features of the training data and features that describe the new samples (SFS Strategy). Experiments were conducted using two real-world datasets, which are described in Section 4.1.

4.1 Experimental Setup

4.1.1 Drift Datasets

Two datasets were chosen to validate our proposed framework: ECUE (Email Classification Using Examples) [Delany et al., 2006] and ETSS datasets. They were pre-processed to have only numeric and binary values, normalized in the range [0.1]. The features of the datasets are shown in Tables 1, 2, 3, and 4. The ECUE dataset is a collection of spam and legitimate e-mails collected from November 2002 to January 2004. Table 1 presents the data distribution of the ECUE training dataset, consisting of 1000 samples, 500 of which were spam-type and 500 samples of the non-spam class (legitimate emails). This set of samples (source domain) is used to train the SVM classifier.

Messages	Source Domain				Total
	2002		2003		
	Nov	Dec	Jan	s/data	
Spam	96	367	37	0	500
NonSpam	83	84	84	249	500
Total	179	451	121	249	1,000

Table 1: Data distribution of ECUE training base. The column 'S/date' date indicates that the sample is not dated.

Table 2 shows the data distribution from the ECUE test dataset, containing 10,865 samples, of which 9,456 are spam-like samples (87% of samples) and 1,409 non-spam samples. This set of samples is used as the target domain to measure the performance of the classifier.

The ETSS dataset consists of a collection of spam and legitimate emails received by eight individuals in the period of June 2012 to October 2013. As in the ECUE database, the ETSS database was split into two sets of samples: training and test datasets. The training dataset corresponds to the collection of emails received in the months of June to

Messages	Target Domain												Total
	2003											2004	
	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	
Spam	142	391	405	459	406	476	582	1,849	1,746	1,300	954	746	9,456
NonSpam	151	56	144	234	128	19	30	182	123	113	99	130	1,409
Total	293	447	549	693	534	495	612	2,031	1,869	1,869	1,053	876	10,865

Table 2: Distribution of precision data base ECUE.

August 2012, while the test dataset corresponds to the collection of emails received in the months of September 2012 to October 2013.

Table 3 presents the distribution of the data of the ETSS training set, composed of 1,000 samples, being 500 samples of spam and 500 non-spam. This dataset is used to train the SVM classifier.

Messages	Target Domain			Total
	2012			
	Jun	Jul	Aug	
Spam	167	167	166	500
NonSpam	167	167	166	500
Total	334	334	332	1,000

Table 3: Data distribution of the ETSS training dataset.

Table 4 shows the distribution of data for the ETSS test dataset, composed of 111,280 samples, being 88,997 spam samples (80% of the samples), and 22,283 non-spam samples.

Message	Target Domain														Total
	2012				2013										
	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	
Spam	4,669	5,427	4,878	4,229	10,281	4,737	6,511	4,039	7,954	3,741	11,016	8,700	7,476	5,293	88,997
NonSpam	1,350	1,715	1,474	1,258	1,282	1,497	1,608	1,915	2,007	1,676	1,910	1,712	1,423	1,454	22,283
Total	6,019	7,142	6,352	5,487	11,563	6,234	8,119	5,954	9,961	5,417	12,926	10,412	8,899	6,747	111,280

Table 4: Data distribution ETSS test base.

4.1.2 Performance Metrics

Table 5 presents the confusion matrix, composed by the following measures: TP (true positive) – amount of spam samples correctly classified as spam; FP (false positive) – number of legitimate emails classified as spam; FN (false negative) - amount of spam classified as legitimate; and TN (true negative) – number of legitimate emails appropriately classified as legitimate.

The Error Rate is used as a metric to calculate the performance of a model trained on a source domain and tested on a target domain. Equation 2 defines the error rate based

Actual Class	Predicted Class	
	Spam	Non-spam
Spam	TP	FN
Non-spam	FP	TN

Table 5: Confusion Matrix.

on false positive and false negative errors:

$$Error\_rate = \frac{(FP + FN)}{(Totalinstances)} \tag{2}$$

4.1.3 Classifier

The SVM classifier is adopted as the base classifier in the proposed approach because it is a well-known good generalization method to solve binary classification problems, such as spam and non-spam email classification. In addition, SVM is a stable classifier, that is, small changes in the data do not significantly affect SVM performance [Buciu et al., 2006]. This characteristics is important to reduce the possibility of false detection of change in problems with dynamic environments. Two initial parameters need to be defined by the user for SVM: the kernel function type and the C regularization parameter. The best results were: Radial Basis Function (RBF) kernel, penalty factor C = 5 with γ = 0.01, for the ECUE database; and RBF (Radial Basis Function) kernel, penalty factor C = 100 with γ = 0.1, for the ETSS database. The training datasets were used to define these parameters, using 10-fold cross-validation.

4.2 Experiments on Concept Drift Detection Using a Single Classification Model

The experiments described in this section are proposed to determine the time period in which the classification model generated remains effective while tested. In particular, two issues are investigated:

1. What is the performance of the classifier considering the error rate in the months following the construction of the classification model?
2. What is the relationship between the error rate of the classifier and the evolution of changes in data distribution?

Two sets of experiments are performed using the ECUE database. In the first series (static model), the classification model is built during the training phase (3 months) and evaluated using the test data (12 months following). In the second series, the model is updated monthly, as simulating a baseline. In this series, the classification model is built and tested monthly using the data that compose the training data. As previously mentioned, we use the information gain feature selection to choose the k best features to represent the data. In our experiments, the ECUE dataset has 166,047 features and after applying information gain 700 features are selected.

In terms of the first series of experiments, the classifier performance behavior measured on the test dataset (Feb/2003 - Jan/2004) can be seen in Figure 3, considering the classification error rate obtained through Equation 1. As can be observed in this figure, the classifier presents an expected behavior, precisely, a gradual and ascending increase of the error rate during the analyzed months. However, some observations require more

careful analysis. For example, why does the month of February, the month following the training dataset, have a high error rate of 3.07%, while the month of April has a 1.09% error rate?

Besides that, Figure 3 shows that ECUE error rate variation reaches 22.03% in January 2004. This behavior leads to advance questioning: the samples that make up the training space used to build the classification model do not have representative features for the problem?

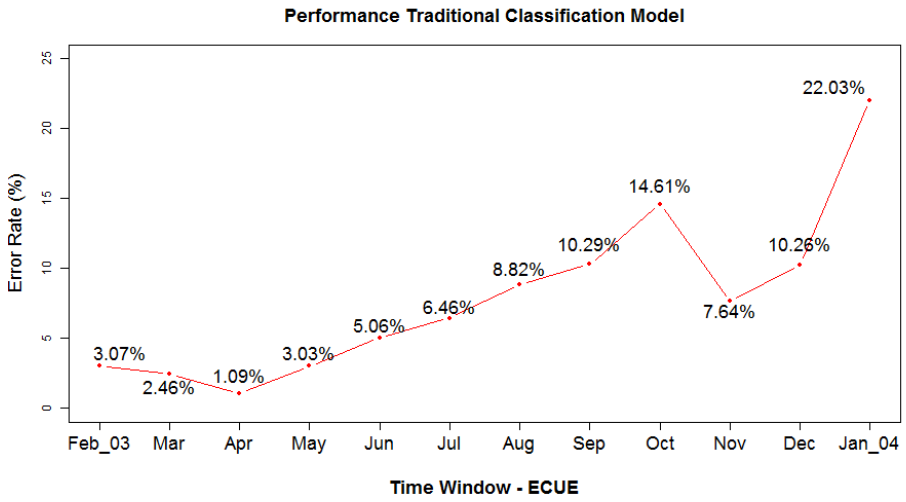

Figure 3: Error rate obtained using a classification model in database ECUE - Series 1.

To investigate these issues and determine the time window in which the classification model generated remains accurate, the second series of experiments was conducted, in which the classification model is built and tested on a monthly basis using the test dataset. It is important to note that the selection of features is also performed every month in these experiments.

Figure 4 shows the comparison between error rates obtained by the classification model built from selected features for three months of training data (called Series 1 or static model experiments) and the error rates obtained for monthly rating models (Series 2 or baseline).

Comparing the error rates of both series, as expected, it can be observed that the generation of a classification model using features monthly selected is always more efficient than the ”static” model used in the first series of experiments. In Figure 4, in all cases of Series 2, the error rate was not higher than 2%. The mean error in Series 1 was 7.90%, while in Series 2 was 0.98%. Besides that, the error rate of 0.34% obtained in the month of February (Series 2) is well below the rate obtained by the static model (Series 1), which was 3.07%. These results from Series 2 demonstrate that (i) features monthly selected are representative for the problem and (ii) there might be an abrupt change in the features that make up the training data in the month of February.

It is important to remember that the variation of the error rate in Series 2 is due to the distribution of the samples in the test data. As shown in Table 3, the months of

March, July, August and December present few samples of the non-spam class. The low representativeness of this class weakens the balanced distribution of samples to apply a cross-validation strategy. For example, the July time window is composed of only 19 non-spam samples. Thus, when applying the 10-fold cross-validation strategy, some folds were composed of fewer than 2 samples of this class. This issue compromises classifier learning, leading to higher error rates.

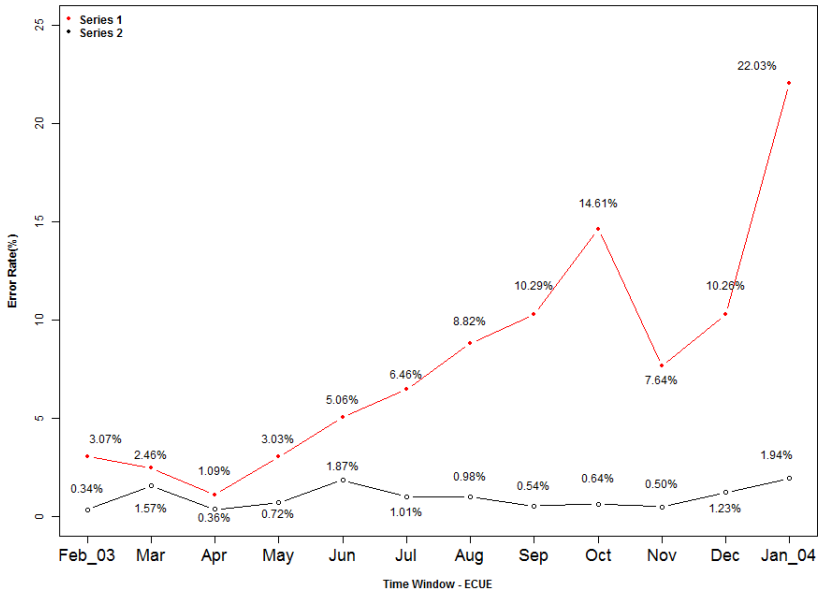

Figure 4: Error rates acquired using a static classification model (Series 1) and a monthly updated model (Series 2).

From the observations made in these initial experiments, another question arises: is there a direct relation between change in data distribution and the error rate of the classifier? To investigate this relationship, an analysis was made on the feature vectors representing training data and test data. This analysis focused on calculating the percentage of features present in the test data and absent in the training data.

As a result of this analysis, Figure 5 shows the percentage of features that were absent in the training data, taking into account monthly time windows, when compared to the test data. The results show that in the first month of the test dataset (Figure 5, February 2003), the percentage of absent features is 43.43%. This change in features has a direct impact on the error rate of the classifier. Considering the results shown in Figure 3, the error rate provided by the classifier in this month was 3.07%. However, this behavior is not linear. For instance, month of April shows a higher change in features vectors (55% of absent features) and a lower error rate (1.09%), when compared to the month of February of the same year.

Therefore, finding the validity of a classification model based on the evolution of

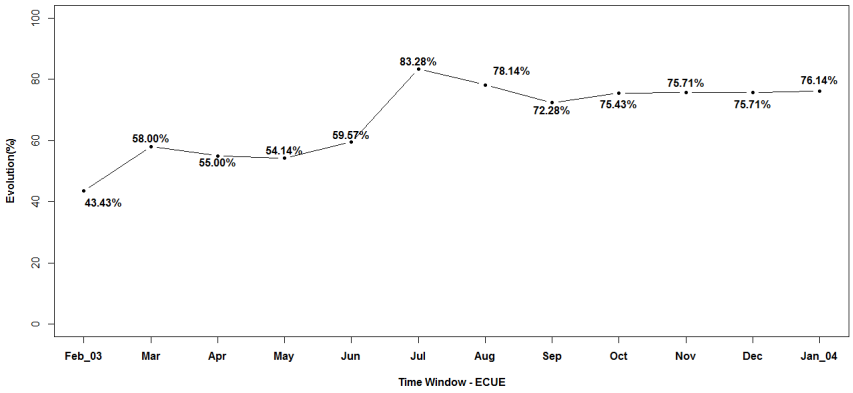

Figure 5: Percentage of missing features by comparing the source and target domains for the ECUE database.

features changing requires a more detailed analysis. Although percentage values of presence or absence of features serve as good indicators for updating the classification model, other factors should be taken into account, such as (i) relevance and (ii) frequency of new features. The results of these initial analyses show that the relationship between changes in features and the classifier error rate is important to identify the efficacy of a classification model. However, from these results new questions arose:

1.The experiments accomplished with Series 1 prove that spam attack detection should not be dealt with static classification models, since it is a dynamic problem. Thus, to obtain better results, it is necessary to keep up changes that occur in spam emails features vectors.

2. The process of feature selection carried out in this work was performed using the IG strategy. The obtained results show that the improvement of the feature selection process can be fundamental for the creation of more efficient classification models.

3. The temporal analysis carried out showed that the percentage values of presence or absence of features may be a good indicator for change detection. However, to update the learning model, other factors must be taken into consideration, such as the relevance and frequency of new features.

In order to answer questions about the relevance and frequency of features, we present below an evaluation of the strategy presented in Section 3.

4.3 Experiments on Concept Drift Detection using ADDrif - SFS

The experiments applying the SFS strategy start with the definition of a threshold. In the light of the issues related to the distribution of the samples obtained in the ECUE database, the evaluation of the SFS strategy will be conducted only with the ETSS database. The analysis of results starts with setting the threshold. Figure 6 shows a scatter plot of the samples on time windows from September to December 2012. The dots represent similarity values obtained from spam samples of the target domain (blue dots) and non-spam (green dots).

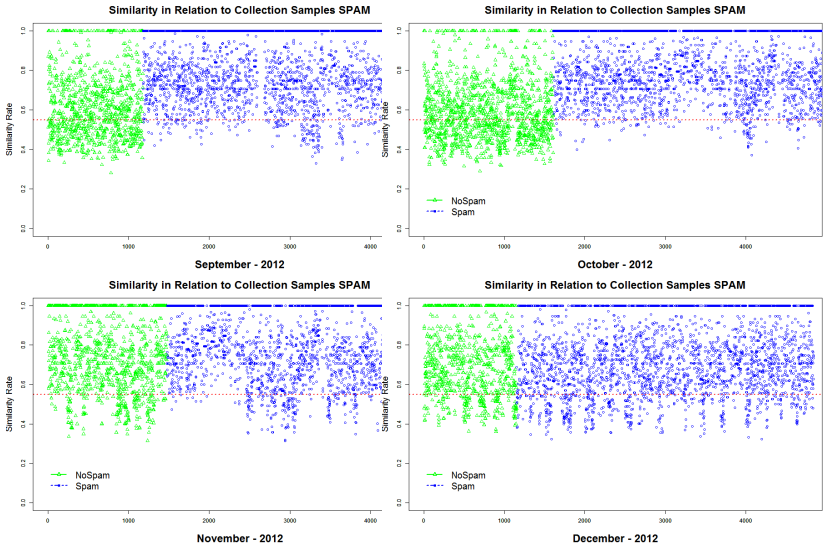

Figure 6: Distribution of email samples in the test window from September to December 2012, compared with samples of e-mail spam in the training data through the similarity measure.

It is observed that most of the samples from the target domain (spam and non-spam) have similarity higher than 0.40 rates for all analyzed time windows. The similarities rates of non-spam sample for months of September and October are concentrated between 0.40 and 0.60. Although, in these same months, similarities rates of spam samples are concentrated between 0.6 and 1.0. Inside two months, those rates are changed, with non-spam and spam samples concentrated above 0.5.

A more detailed analysis shows that in the range between 0.5 and 0.55, there is a focus on more samples of spam than non-spam. For this reason, for the ETSS database, the threshold set was 0.55, i.e. each email whose similarity rate is lower than 0.55 is considered a sample that presents some evolution of change in its features. The experiments follow with the use of the SFS strategy on monthly time windows, considering the limit defined above and a prediction to the classification model.

Figure 7 presents the error rate obtained by the classifier when the SFS strategy is applied considering monthly time windows (Series 3). It is known that the static model is not suitable for dynamic problems such as spam attacks. The series 2 represents an ideal strategy for updating the classification model, that is, the classification model is trained and tested about the best perspective, because in its training the representation closest to the tested environment is used, and it is therefore, a baseline for the spam detection problem. It is important to mention that series 2 is not possible to be performed in real-world application, since it assumes that labeled samples are constantly arriving to train a model monthly. On the other hand, series 3 represents a situation with an approximate result to that of the baseline, updating the classification model monthly using samples stored in temporary repository in order to generate the new learning environment. Figure 7 shows the behavior of this scenario in terms of the models classification error rate, compared to Series 2. It may be noted that the error curve achieved by our method presents

the same behavior as observed in the baseline. In addition, the difference between the error rates from both series of experiments is small (1.03% in average).

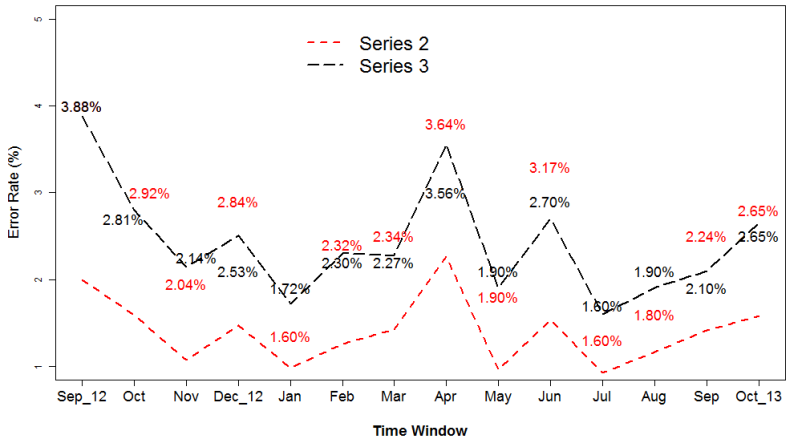

Figure 7: Distribution of email samples in test windows from September to December 2012, compared to samples of e-mail spam in the training data through the similarity measure.

Figure 8 shows graphically the amount of samples transferred to new knowledge bases in Series 3 by means of the SFS strategy. While in Series 2 there is no update, because the space of features used to construct the classifier is the same used for test samples. Series 2 simulates a classifier that guesses the best case, that is, it deals with the features of the environment tested. It is the best situation in which a classifier can be built, based on the features that will be employed by attackers in spam message in the future.

4.4 Discussion

Setting a value for the threshold is not a simple task. The main difficulty is in high dimensionality feature spaces that describe non-spam and spam emails. By observing the similarity rates of the samples of interest considering only the k best ranked features, we could simplify the issue and this helped us to identify the threshold.

Evaluating the time windows, the distancing of the similarities of spam emails samples was found in a gradual manner in relation to the first window. Taking into account that the attackers try to approach more and more email spam to legitimate messages, it generates a great difficulty in keeping the classification models updated. Samples that compose the time windows demonstrate through the extracted features that approximately 42% of the spam features are overlapped with legitimate email features. It is important to emphasize that the series 2 of experiments is the representation of an ideal scenario. However, the average difference between the two series (baseline e SFS) presents an error rate of 1.03%, i.e., approximately 6 time windows present a difference in the error

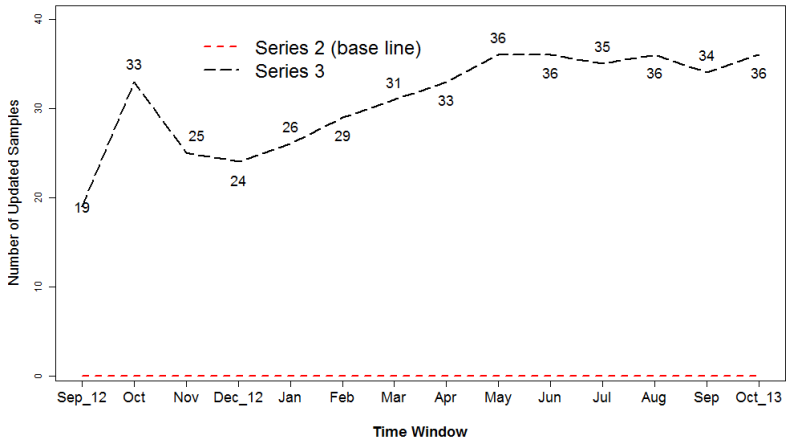

Figure 8: Amount of samples updated in Series 2 (baseline) and in Series 3 (SFS strategy) using the ETSS database.

rate lower than 1%, precisely, January, March, May, July, August and September 2013. The SFS strategy positively returned a monthly update, performed using Series 3, and approached the baseline (Series 2).

5 Conclusion

The challenge of monitoring or detecting changes makes message filters an incremental machine learning problem. This paper addresses this problem by providing a framework for detecting spam attacks based on changes in data distribution over time (ADDrift). The strategy of change detection have been presented and implemented to handle concept drift and automatically transfer the new knowledge identified from the evolution of the features, with the goal of understanding the problem and keeping the training data always up to date in anticipation of the degradation of the classification model.

The SFS (Similarity-based Features Selection) strategy was proposed with the objective of observing the change in the features spaces of spam emails. The new strategy looks at the features that describe the source domain by employing a measure of similarity between the samples composing source and target domains. This measure of similarity made it possible to identify the changes according to the nature of the presented data, whether abrupt or not.

Observing the behavior of the samples using the proposed similarity measure, it was possible to identify a threshold for the database used, capable of determining the moment in which the evolution of the data distribution occurs. An analysis of the distribution of similarity measures of the domain-of-interest samples showed that initially the range for most spam e-mail samples was concentrated between 0.60 and 1.0, as well as the range for e-mail samples (non-spam) was concentrated between 0.40 and 0.60. It is important to note that a sample reaches 100% similarity with the initial domain of knowledge once its similarity is equal to 1, that is, the range that provides analysis of this measure of similarity is between 0 (zero) and 1 (one). A more detailed experimentation of the

samples revealed that a threshold set to 0.55 is adequate to separate (legitimate) spam and non-spam data classes contained in ETSS database.

For the transfer learning phase, when a change is observed, the samples stored in a repository (new knowledge) are used for updating the model. The transfer learning occurs by means of similarity measure, which indicates to how closely the model elements are related, that is, the elements (samples) with little or no similarity, will compose the new knowledge by presenting non-representative features of the training data, most relevant to the test data. The construction of the new knowledge is totally based on the data distribution that describe the problem in the evaluated time frame.

Acknowledgements

This research, according to Article 48 of Decree nº 6.008/2006, was partially funded by Samsung Electronics of Amazonia Ltda, under the terms of Federal Law nº 8.387/1991, through agreement nº 003/2019, signed with ICOMP/UFAM.

References

[Kaspersky, 2020] Kaspersky Security Bulletin 2020. Statistics, Kaspersky Security Bulletin Statistics for 2020.

[ISTR, 2018] Symantec Corporation, 2018 Internet Security Threat Report - ISTR vol. 23 (2018) 1–89.

[Alazab and Broadhurts, 2016] Alazab, M., & Broadhurst, R. (2017). An analysis of the nature of spam as cybercrime. In Cyber-Physical Security. Springer, 251-266.

[Kocz, 2004] Kołcz, A., Chowdhury, A., & Alspector, J. (2004, July). The impact of feature selection on signature-driven spam detection. In Proceedings of the 1st Conference on Email and Anti-Spam (CEAS-2004).

[Delany, 2005] Delany S., Cunningham P., Tsymbal, A., Coyle, L. A case-based technique for tracking concept drift in spam filtering, Knowledge-Based Systems 18 (4-5) (2005) 187–195.

[Guzella and Caminhas, 2009] Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. Expert Systems with Applications, 36(7), 10206-10222.

[Hayat et al., 2010] Zi Hayat, M., Basiri, J., Seyedhossein, L., Shakery, A. Content-based concept drift detection for Email spam filtering, 2010 5th International Symposium on Telecommunications (2010) 531–536.

[Su and Shen, 2008] Su, B., Shen, Y.-d. Modeling concept drift from the perspective of classifiers, 2008 IEEE Conference on Cybernetics and Intelligent Systems (2008) 1055–1060.

[Blazieri and Bryl, 2009] Blanzieri, E., Bryl, A. A survey of learning-based techniques of email spam filtering, Artificial Intelligence Review 29 (1) (2009) 63–92.

[Fdez-Riverola and Iglesias, 2007] Fdez-Riverola, F., Iglesias, E., Díaz, F., Méndez, J. Corchado, J. SpamHunting: An instance-based reasoning system for spam labelling and filtering, Decision Support Systems 43 (3) (2007) 722–736.

[Caruana and Li, 2012] Caruana, G., Li, M., A survey of emerging approaches to spam filtering, ACM Computing Surveys 44 (2) (2012) 1–27.

[koren, 2009] Koren, Y. Collaborative filtering with temporal dynamics, Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’09 (2009) 447–456.

[Gama, 2004] Gama, J., Medas, P., Castillo, G., Rodrigues, P. Learning with drift detection, Brazilian Symposium on Artificial Intelligence 3171 of LN (2004) 286–295.

[Taylor and Stone, 2009] Taylor, M. E., Stone, P. Transfer Learning for Reinforcement Learning Domains : A Survey 10 (2009) 1633–1685.

[Ebner et al., 2007] Ebner, M., O'Neill, M., Ekárt, A., Vanneschi, L., Esparcia-Alcázar, A.I Genetic Programming, Genetic Programming, vol. 4445. Springer Berlin Heidelberg.

[Ganti and Ramakrishnan, 2002] Ganti, V., Gehrke, J., and Ramakrishnan, R. Mining data streams under block evolution, ACM SIGKDD Explor. Newsl., vol. 3, no. 2, 1–10.

[Delany et al., 2006] Delany, S., Cunningham, P., Tsymbal, A. A comparison of ensemble and case-base maintenance techniques for handling concept drift in spam filtering, in: FLAIRS Conference, 2006, pp. 340–345.

[Zhang et al., 2014] Zhang, Y., Wang, S., Phillips, P., Ji, G. Binary PSO with mutation operator for feature selection using decision tree applied to spam detection, Knowledge-Based Systems 64 (2014) 22–31.

[Goncalves, 2014] Gonçalves Jr, P. M., de Carvalho Santos, S. G., Barros, R. S., & Vieira, D. C. (2014). A comparative study on concept drift detectors. Expert Systems with Applications, 41(18), 8144–8156.

[Zhou, 2010a] Zhou, C. V., Leckie, C., & Karunasekera, S. (2010). A survey of coordinated attacks and collaborative intrusion detection. Computers & Security, 29(1), 124–140.

[Widmer and Kubat, 1996] Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. Machine learning, 23(1), 69–101.

[Jackowski, 2014] Jackowski, K. (2014). Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers. Pattern Analysis and Applications, 17(4), 709–724.

[Haury et al., 2011] Haury, A. C., Gestraud, P., & Vert, J. P. (2011). The influence of feature selection methods on accuracy, stability and interpretability of molecular signatures. PloS one, 6(12), e28210.

[Alqatawna et al., 2015] Alqatawna, J., Faris, H., Jaradat, K., Al-Zewairi, M., & Adwan, O. (2015). Improving knowledge based spam detection methods: The effect of malicious related features in imbalance data distribution. International Journal of Communications, Network and System Sciences, 8(05), 118–129.

[Wenerstrom and Giraud-carrier, 2006] Wenerstrom, B., & Giraud-Carrier, C. (2006). Temporal data mining in dynamic feature spaces. In Sixth International Conference on Data Mining (ICDM'06). IEEE, 1141–1145.

[Tsymbal et al., 2008] Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. Information fusion, 9(1), 56–68.

[Lai et al., 2009] Lai, G. H., Chen, C. M., Laih, C. S., & Chen, T. (2009). A collaborative anti-spam system. Expert Systems with Applications, 36(3), 6645–6653.

[Zliobaite, 2009] Žliobaitė, I. Learning under concept drift: an overview, Technical report, Vilnius University, 2009 techniques, related areas, applications (2009) 1–36.

[Lu et al., 2014] Lu, N., Zhang, G., & Lu, J. (2014). Concept drift detection via competence models. Artificial Intelligence, 209, 11–28.

[Gao et al., 2008] Gao, J., Fan, W., Jiang, J., & Han, J. (2008). Knowledge transfer via multiple model local structure mapping. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 283–291.

[Gama and Kosina, 2014] J. Gama, P. Kosina, Recurrent concepts in data streams classification, Knowledge and Information Systems 40 (3) (2014) 489–507.

[Salton et al., 1975] Salton, G., Wong, A., and Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620.

[Hamers et al., 1989] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, A. Vanhoutte, (1989). Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula. Information Processing and Management, 25(3), 315–318.

[Quinlan J.R., 1992] Quinlan, J. R. C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning). Morgan Kaufmann Publishers In, 1992.

[Buciu et al., 2006] Buciu, I., Kotropoulos, C., and Pitas, I. (2006). Demonstrating the stability of support vector machines for classification. Signal Processing, 86(9), 2364–2380.

[Dada et al., 2019] Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., and Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, 5(6), e01802.

[Anderson and Reich 2000] Anderson, K., Reich, S. (eds.): "Proceedings of the Second Workshop on Structural Computing"; Lect. Notes Comp. Sci. 1903, Springer, Berlin.

[Christodoulakis et al. 1999] Tzagarakis, M., Vaitis, M., Papadopoulos, A., & Christodoulakis, D. (1999). The Callimachus approach to distributed hypermedia. In Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots: returning to our diverse roots, 47–48.

[Grønbæk and Trigg 1994] Grønbæk, K., & Trigg, R. H. (1993). Design issues for a Dexter-based hypermedia system. In Proceedings of the ACM conference on Hypertext, 191–200.

[Hicks et al. 1998] Hicks, D. L., Leggett, J. J., Nürnberg, P. J., & Schnase, J. L. (1998). A hypermedia version control framework. ACM Transactions on Information Systems (TOIS), 16(2), 127–160.

[LOC 2000] Library of Congress. Network Development, MARC Standards Office, National Library of Canada, National Library of Canada. Standards, & Support. (2000). MARC 21 Specifications for Record Structure, Character Sets, and Exchange Media. Library of Congress.

[Marshall et al. 1994] Marshall, C. C., Shipman III, F. M., & Coombs, J. H. (1994). VIKI: Spatial hypertext supporting emergent structure. In Proceedings of the 1994 ACM European conference on Hypermedia technology, 13–23.

[Martinez 2002] Martinez, J.: "Coding of moving pictures and audio: Overview of the MPEG-7 standard (version 6.0)" ISO/IEC JTC1/SC29/WG11 N4509 (2001).

[McCall et al. 1990] McCall, R., Bennett, P. R., d'Oronzio, P. S., Ostwald, J. L., Shipman III, F. M., & Wallace, N. F. (1990). PHIDIAS: Integrating CAD Graphics into Dynamic Hypertext, 152–165.

[Nelson 1993] Nelson, T.: "Literary Machines"; Mindful, Sausalito, CA.

[Neveu et al. 2001] Neveu, Y., Guervilly, Y., Wiil, U. K., & Hicks, D. L. (2001). Providing metadata services on the World Wide Web. Technical Report CSE-01-01, Department of Computer Science and Engineering, Aalborg University Esbjerg, Denmark.

[Nürnberg 1999] Nürnberg, P. (ed.): "Proceedings of the Workshop on Structural Computing (SC1)". Technical Report CSE-99-04, Dept. Comp. Sci. & Eng., Aalborg U. Esbjerg, (Feb).

[Nürnberg et al. 1998] Nürnberg, P. J., Wiil, U. K., & Leggett, J. J. (1998). Structuring facilities in digital libraries. In International Conference on Theory and Practice of Digital Libraries, 295–313. Springer, Berlin, Heidelberg.

[Nürnberg et al. 1997] Nürnberg, P. J., Leggett, J. J., & Schneider, E. R. (1997). As we should have thought. In Proceedings of the eighth ACM conference on Hypertext, 96–101.

[Nürnberg et al. 1996] Nürnberg, P. J., Leggett, J. J., Schneider, E. R., & Schnase, J. L. (1996). Hypermedia operating systems: A new paradigm for computing. In Proceedings of the the seventh ACM conference on Hypertext, 194–202.

[Reich et al. 2001] Reich, S., Tzagarakis, M., De Bra, P. (eds.): "Proceedings of the Third Workshop on Structural Computing"; Lect. Notes Comp. Sci. 2266, Springer, Berlin.

[Schraefel 2000] Schraefel, M.: "ConTexts: Adaptable hypermedia"; Proc. Adapt. Hypermedia and Adapt. Web-Based Sys. Int. Conf., Lect. Notes in Comp. Sci. 1892, Springer, Berlin (Aug 2000), 369–375.

[Wiebel et al. 1998] Weibel, S., Kunze, J., Lagoze, C., & Wolf, M. (1998). Dublin core metadata for resource discovery. Internet Engineering Task Force RFC, 2413(222), 132.

[Wiil and Leggett 1996] Wiil, U. K., & Leggett, J. J. (1996). The HyperDisco approach to open hypermedia systems. In Proceedings of the the seventh ACM conference on Hypertext, 140–148.

[Wiil and Leggett 1992] Wiil, U. K., & Leggett, J. J. (1993). Hyperform: using extensibility to develop dynamic, open, and distributed hypertext systems. In Proceedings of the ACM Conference on Hypertext, 251–261.

[Wiil and Nürnberg 1999] Wiil, U. K., & Nürnberg, P. J. (1999). Evolving hypermedia middleware services: Lessons and observations. In Proceedings of the 1999 ACM symposium on Applied Computing, 427–436.

[Dada et al. 2019] Dada, E. G., Bassi, J. S., Chiroma, H., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, 5(6), e01802.