Goal-oriented Email Stream Classifier with A Multi-agent System Approach

Wenny Hojas-Mazo[1], Mailyn Moreno-Espino[2]
José Vicente Berná Martínez[3], Francisco Maciá Pérez[4], Iren Lorenzo Fonseca[5]
Technological University of Havana "José Antonio Echeverria" – CUJAE, Havana, Cuba[1, 2]
Department of Computer Science and Technology, University of Alicante, Alicante, Spain[3, 4, 5]

***Abstract*—Now-a-days, email is often one of the most widely used means of communication despite the rise of other communication methods such as instant messaging or communication via social networks. The need to automate the email stream management increases for reasons such as multi-folder categorization, and spam email classification. There are solutions based on email content, capable of contemplating elements such as the text subjective nature, adverse effects of concept drift, among others. This paper presents an email stream classifier with a goal-oriented approach to client and server environment. The i* language was the basis for designing the proposed email stream classifier. The email environment was represented with the early requirements model and the proposed classifier with the late requirements model. The classifier was implemented following a multi-agent system approach supported by JADE agent platform and Implementation_JADE pattern. The behavior of agents was taking from an existing classifier. The multi-agent classifier was evaluated using functional, efficacy and performance tests, which compared the existing classifier with the multi-agent approach. The results obtained were satisfactory in all the tests. The performance of multi-agent approach was better than the existing classifier due to the use of multi-threads.**

Keywords*—Email stream classification; goal-oriented requirements; i*; multi-agent system

I. Introduction

Email is one of the most widely used services by Internet users. Moreover, the growth in the number of users makes this service grow as well. [1]. Every user can receive around 40-50 emails per day [2]; but other professional users may receive hundreds or thousands per day. Users spend a lot of time processing the emails they receive on a daily basis. This implies that email management is a major problem in organisations and that it is therefore important to have tools, preferably intelligent ones, to solve this problem. [1]. There are many types of tools for automatic mail management; one of them is the automatic email classifier [3, 4]. An automatic email stream classifier allows for quick and agile classification of emails into discrete sets of predefined categories [1]; for example, to classify an incoming email into professional or personal, spam or desirable, phishing or normal.

There are two levels where the email classification is applied: user and server. Email classification can be considered a goal that serves as a means to satisfy other goals. An example of this is the email filtering that the mail user agent performs in the client application or the spam email detector on the server. This includes other actors who relate to each other to achieve proper functioning, an aspect that could be represented through social modelling [5].

Email is one of the communication media through which most problems and security incidents occur due to spam and phishing [1]. According to [6] up to 80% of the emails sent worldwide are created by spam [6]. The adverse effect caused by spam emails has resulted in the economic loss of billions of dollars annually [7]. Several approaches have been proposed for spam detection [8]. To evaluate the performance of the filters, it has been published diverse corpus [9], different measures [10] and evaluation methods [11] have been used.

Moreover, the classifier, to be deployed in real environments, covers various aspects such as: the email pre-processing, the features selection, the concept drift detection and the classification itself that depends on the other aspects mentioned above. Proposals facing the challenge of increasing the adaptive capacity of email classification solutions tend to focus on specific modules [12, 13, 14, 15, 6]. However, these solutions do not provide a representation that relates the objectives to be achieved with each of the aspects to achieve the email classification. These relationships can lay the foundations for reactive, proactive and social behaviors that allow the classifier to increase his ability to adapt.

The main contribution of this work is a goal-oriented email stream classifier for client and server environment with a multi-agent system approach. Email environment requirements and proposed classifier were modeled with early and late requirements of i*. The email environment was represented with the early requirements and the proposed classifier with the late requirements. The classifier was implemented following a multi-agent system approach supported by JADE agent platform and Implementation_JADE pattern.

The paper is organized as follows: Section 2 offer background and an overview of related work. Section 3 presents proposed solution. Section 4 describes proposed solution evaluation. Section 5 gives conclusions.

II. Background and Related Work

The architecture of the email system consists of two kinds of subsystems [16]: Mail User Agents (MUAs) and the Message Transfer Agents (MTA). MUA is a client application that allows to the users to manage emails of their mailboxes. It can be desktop application (e.g. Thunderbird) or web-based (e.g. Gmail) and includes functionalities such as to compose, to

display, to organize and to filter messages. MTA, informally known as email servers, move the messages from the source to the destination sometimes through Internet, or if the recipient's server has been reached, to the Mail Deliver Agent (MDA). An example of email architecture [4] is showing in Fig. 1.

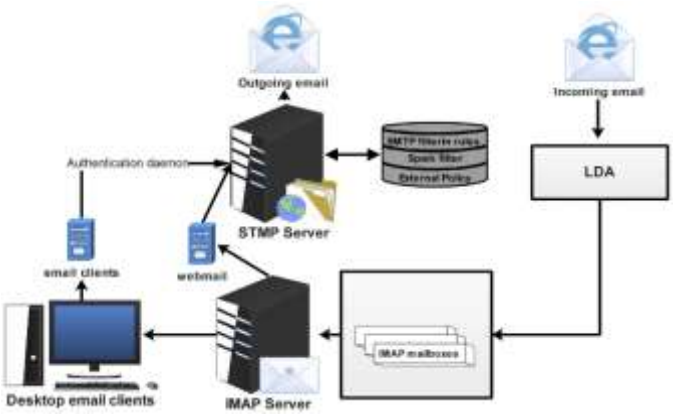

Fig. 1. Example of Email Architecture.

Other components complement the MTA such as user repository and content filter. User repository manages user information such as their username, password and profile information. Content filter evaluates incoming email to determine the probability that the messages are legitimate. This evaluation is supporting with filters such as antivirus filter and spam filter.

Moreover, the email stream classifiers, to be deployed in real environments, cover various aspects such as: the email pre-processing, the features selection, the concept drift detection and the classification itself that depends on the other aspects mentioned above. For example, features selection is intended to identify only those features with the highest discriminatory capacity to improve classifier performance [6]. Proposals, that face the challenge of increasing the adaptive capacity, tend to focus on specific modules [12, 13, 14, 15]. However, these solutions do not provide a representation that relates the objectives to be achieved with each of the aspects to achieve the classification of emails. These relationships can lay the foundations for reactive, proactive and social behaviors that allow the classifier to increase his ability to adapt. This requires that the solutions have adaptive capacities [17] to reduce the negative effects caused by noise, concept drift and the constant appearance of new instances. Requirements capture is an important action for developing an email stream classifier with these characteristics.

Requirements are the first stage when developing a software [18] and might be considered as one of the most important stages [19]. In the requirements analysis stage, the analysts detect the needs of the stakeholders to generate a system description document. At this stage, the goals, functionalities, and constraints (why is the system necessary) of the system are elicited in order to implement the established requirements [20]. Social modelling [21] is an option to capture these requirements.

Social modeling is focusing in system social dimension, it environment and adopts requirements with a goal-oriented approach by seeing intentions and reasons behind a behavior [21]. A detailed analysis of goals reveals desires, which shows expectations or troubles. A goal-oriented model may help to manage changes and allows evaluating alternative solutions by showing strengths and weaknesses [22, 23]. Goals provide criteria, support formal reasoning schemes during requirements engineering and guides to evaluate possible solutions [22, 23] and have been widely used and discussed in literature. Several examples of how to use goal-oriented models and how to apply them in real projects can be found in [21].

We can use the modeling language i* to introduces aspects of social modeling on requirements stage, this modeling follows a goal-oriented approach [24]. In i*, actors are seen as intentional, i.e. they have abilities, goals, beliefs and obligations. Thus, the analysis of each actor focuses on capturing their objectives, considering the relationships between the human actors and the future software system. This analysis allows setting the strategic interests of actors [25].

When we use i*, the requirement stage is divided in two other steeps [24]: step 1 Early Requirements and step 2 Late Requirements. Early requirements identify the actors involved in the context of the problem, their needs and their intentions. Late Requirements models what the futures software system should do and do this description using the most clear form as possible [26]. i* uses the Strategic Dependence model (SD) and the Strategic Rational model (SR) [24], each one with a different level of abstraction. In the SD model, dependency-relations existing among social actors are represented. In the SR model dependencies among objects within an actor are represente. Strength points of goal-oriented approach can be exploiting by agent-oriented [21].

III. Proposed Solution

During the email stream classifier development, three fundamental activities were performed. The first and second activities are focusing in to model in i* by using the tool TAOM4E [27]. The activity third consists in to implement the email stream classifier with a multi-agent system approach.

A. Activity 1

Use early requirements to model the requirements of the email context. The following steps were performed and Table I describes social actors identified:

1) Identify and model the social actors that are involved in the business context. All the actors modeled are show in Table I.

2) Represent the dependencies between actors using the SD model.

3) Identify and represent the objectives of each actor through the RS model.

In Fig. 2 are represented the SD and SR models obtained during the step of early requirements.

TABLE I. SOCIAL ACTORS IDENTIFIED IN EARLY REQUIREMENTS

Social actor	Description
User	Represents the people that use a MUA to manage emails.
MUA	Is a client application that allows to the users manage emails. It can be desktop application (e.g. Thunderbird) or web-based (e.g. Gmail).
MTA	This server application receives emails from MUA, or from another MTA, transfers the mail to another MTA (e.g. using Internet) and if the recipient's server has been reached, transfers the email to the MDA. Postfix is an extended example.
MDA	This server program stores the mail received form servers´s MTA into the mailbox. An example is Dovecot.
Content Filter	Evaluates incoming email to determine the probability that the messages are legitimate. An example is Amavis.
Antivirus Filter	A server program to recognize virus so as to prevent its delivery.
Spam Filter	A server program to recognize spam so as to prevent its delivery.
Internet	A vast collection of different networks that use certain common protocols and provide certain common services.
User Repository	Manages user information such as their username, password and profile information. It can be an SQL database, an LDAP or so on.

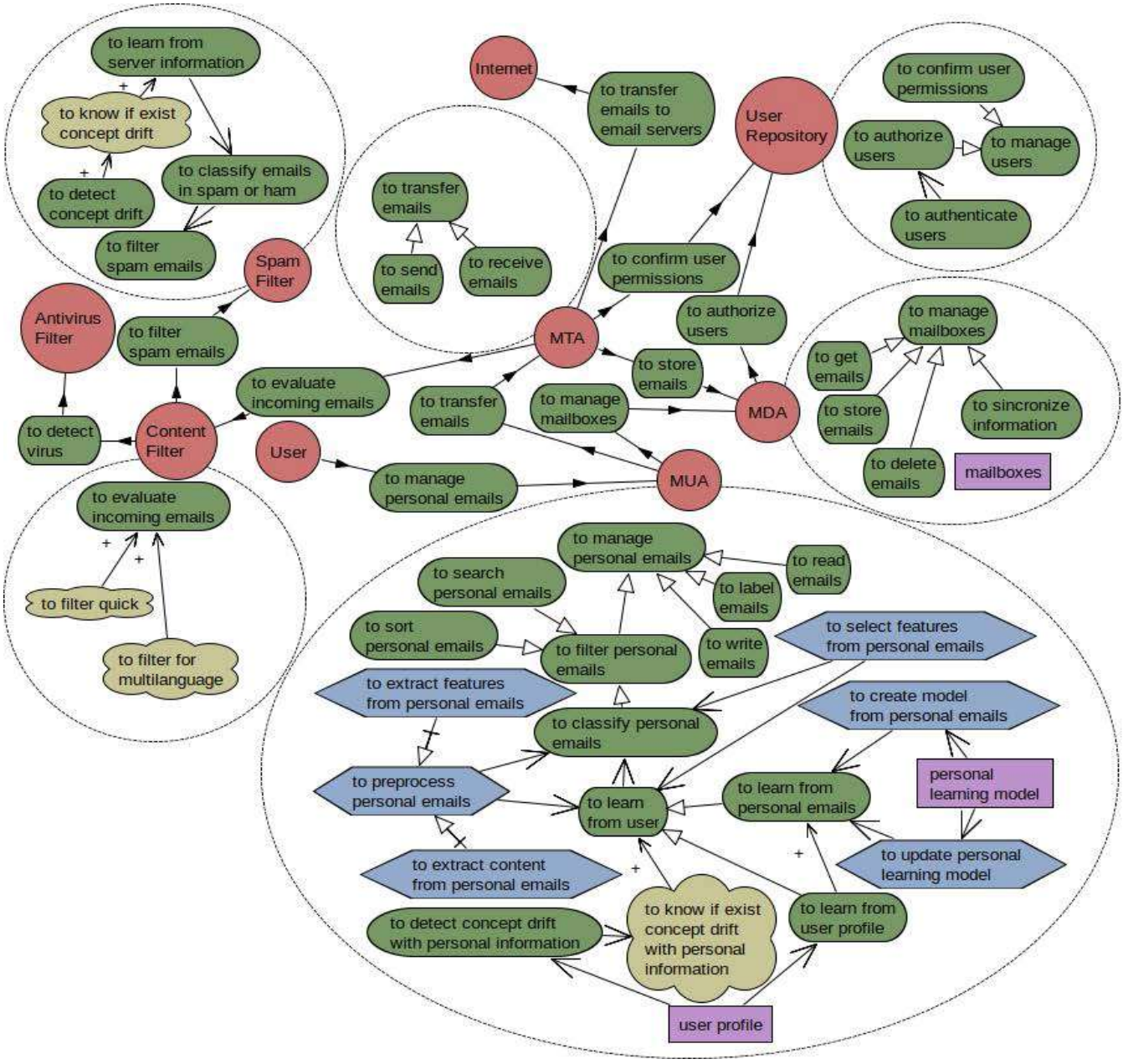

Fig. 2. SD and SR Models of Early Requirements.

B. Activity 2

To model the email stream classifier with late requirements we use the following steps:

1) Modelling social actors within the context of the system.

2) Representing dependencies among actors in an SD model.

3) Representing the goals of each actor in SR models.

Table II shows the social actors identified during late requirements. In Fig. 3 are represented the SD and SR models obtained during late requirements.

C. Activity 3

To implement the email stream classifier with a multi-agent system approach.

In this activity, a Multi-Agent System (MAS) based in late requirement modeling with i* to classify email (spam or ham) is proposed. The MAS is useful for problem resolution in distributed environments [28]. MAS agents are always active and organize so that their behavior emerges from the bottom up. This makes it easier to change the organizational structure when appropriate, or to expand its use, which enhances reusability. The agents of the proposed MAS are the actors represented as systems in the modeling of the late requirements, which can change their behavior without having a negative impact on the rest of the system.

The implementation of the solution was based on an existing spam email classifier, the JADE agent platform [29] and the Implementation_JADE pattern [30]. The classifier was used to take the behavior that each of the agents identified in the late requirements diagram would have. The JADE platform supports the development of the MAS, which provides advantages such as [31]: simplifying the development of a MAS, guaranteeing compliance with FIPA standards; high cohesion and low coupling between the modules; simple liability; independent threads to simultaneously perform different functions; allows programmers to focus on the specific parts of your problem; and adaptive and decision-making capacity. The Implementation_JADE pattern [30] encapsulates the processes and functionalities that can be implemented with the JADE platform and implements and solves those functionalities that are essential so that developers can make use of this agent technology more easily.

TABLE II. SOCIAL ACTORS IDENTIFIED IN LATE REQUIREMENTS

Social actor	Description
MUA	Mail User Agent is a client application that allows to the users manages emails.
Spam Filter	A server program to recognize spam so as to prevent its delivery.
Classifier	A system that classifies emails at MUA and server level.
Preprocesser	A system that preprocesses emails to extract features.
Features Selector	A system that selects the most relevant features.

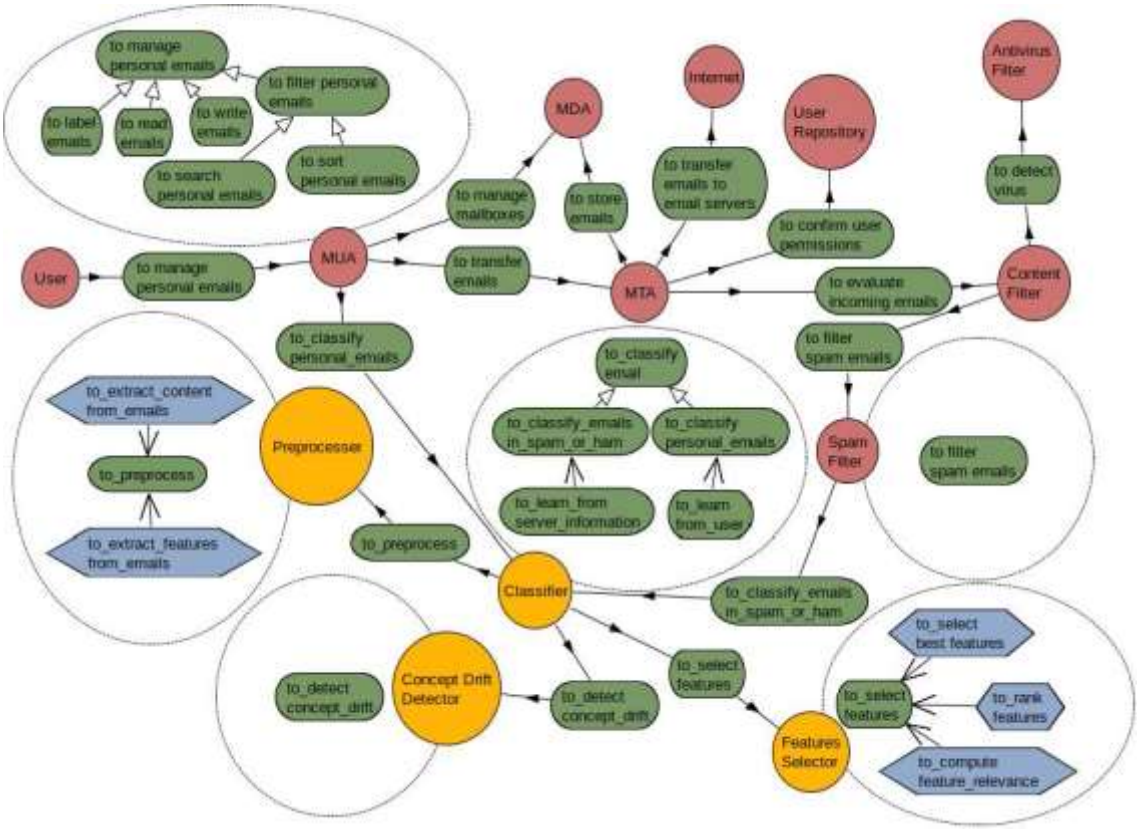

Fig. 3. SD and SR Models of Late Requirements.

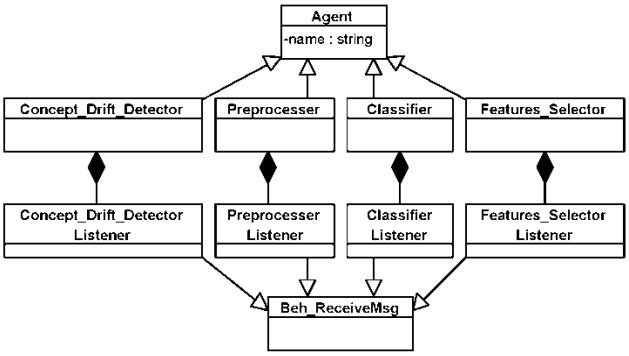

Fig. 4. Class Diagram of Multi-Agent System.

Fig. 4 represents the classes that make up the MAS. The Classifier, Concept_Drift_Detector, Features_Selector, and Preprocesser classes represent agents, while the listener classes represent the fundamental behavior of each agent. In the Listeners the message will be received and later an action will be executed according to what was received in the message or another behavior could also be executed. This implementation makes it easier for agents to add new behaviors to subsequent versions and to fully exploit their capabilities in the system.

IV. Evaluation of Classifier with Multi-Agent System Approach

To evaluate the solution, functional, efficacy and performance tests were carried out, taking as a reference the base spam classifier for the proposed MAS. The functional and efficacy tests consist of classifying 35 emails with the reference classifier and by the proposed MAS, with the aim of verifying that the results of both classifications coincide and therefore that the MAS works correctly. The results for the existed classifier (Classifier) and the proposed MAS (MAS), with respect to accuracy, True Positive Rate (TPR) and False Positive Rate (FPR), are shown in Fig. 5. The coincidence of the results both systems show the correct functioning of the proposed MAS.

Moreover, performance tests consist of obtaining the times in milliseconds (ms) that it takes to classify different amounts of emails, the reference classifier and the proposed MAS, in order to see how the processing time behaves. The results of these tests are shown in Fig. 6 and a decrease in the processing time is evident by the proposed MAS with respect to the reference classifier. This decrease in time is due to the use of multi-threads incorporated by JADE.

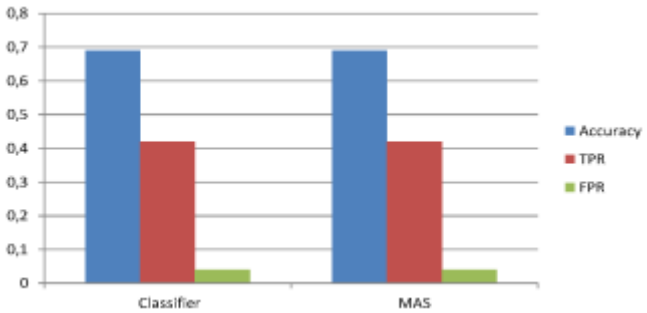

Fig. 5. Efficiency Test Results.

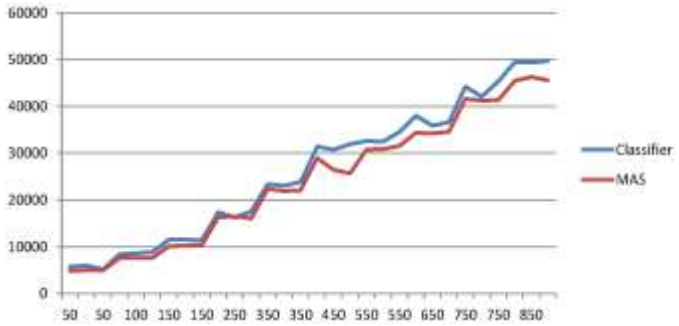

Fig. 6. Performance Tests Results.

V. Conclusion and Future Work

With aspects of social modeling and the language i* was representing actors, goals, tasks, resources and dependency-relations existing among actors of the email environment and designing an email stream classifier by following a goal-oriented approach. The email stream classifier was implemented with a multi-agent system approach. This will allows establishing bases to future achieve reactive, proactive and social behaviors, which allows the classifier to increase his adaptability. The results obtained in functional, efficacy and performance tests were satisfactory. The performance of multi-agent approach was better than the existing classifier due to the use of multi-threads incorporated by JADE. In future work, it is recommended to incorporate behaviors into the multi-agent system of existing solutions with better results and carry out a more exhaustive evaluation of the classifier.

Acknowledgment

This work was performed as part of the Smart University Project financed by the University of Alicante.

References

[1] Mujtaba, G., Shuib, L., Raj, R. G., Majeed, N., & Al-Garadi, M. A. (2017). Email classification research trends: Review and open issues. IEEE Access, Vol. 5, pp. 9044-9064.

[2] Team, R. (2020). Email statistics report, 2020-2024. Technical report, The Radicati Group, Inc. Palo Alto, CA, USA.

[3] Bhowmick, A. & Hazarika, S. M. (2018). E-mail spam filtering: A review of techniques and trends. Kalam, A., Das, S., & Sharma, K., editors, Advances in Electronics, Communication and Computing, Springer Singapore, Singapore, pp. 583-590.

[4] Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. Heliyon, Vol. 5, pp. 1–23.

[5] Luh, R., Marschalek, S., Kaiser, M., Janicke, H., & Schrittwieser, S. (2017). Semantics-aware detection of targeted attacks: a survey. Journal of Computer Virology and Hacking Techniques, 13(1), 47-85.

[6] Sanghani, G. & Kotecha, K. (2019). Incremental personalized e-mail spam filter using novel tfdcr feature selection with dynamic feature update. Expert Systems With Applications, Vol. 115, pp. 287-299.

[7] Rao, J. M. & Reiley, D. H. (2012). The economics of spam. Journal of Economic Perspectives, Vol. 26, pp. 87-110.

[8] Hussain, N., Turab Mirza, H., Rasool, G., Hussain, I., & Kaleem, M. (2019). Spam review detection techniques: A systematic literature review. Applied Sciences, 9(5), 987.

[9] Asdaghi, F., & Soleimani, A. (2019). An effective feature selection method for web spam detection. Knowledge-Based Systems, 166, 198-206.

[10] Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. (2015). Survey of review spam detection using machine learning techniques. Journal of Big Data, 2(1), 23.

[11] Nandhini, S., & KS, J. M. (2020, February). Performance Evaluation of Machine Learning Algorithms for Email Spam Detection. In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE) (pp. 1-4). IEEE.

[12] Bahgat, E. M., Rady, S., Gad, W., & Moawad, I. F. (2018). Efficient email classification approach based on semantic methods. Ain Shams Engineering Journal, Vol. 9, pp. 3259-3269.

[13] Barddal, J. P., Gomes, H. M., Enembreck, F., & Pfahringer, B. (2017). A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. Journal of Systems and Software, Vol. 127, pp. 278-294.

[14] Diale, M., Celik, T., & Van-Der-Walt, C. (2019). Unsupervised feature learning for spam email filtering. Computers and Electrical Engineering, Vol. 74, pp. 89-104.

[15] Méndez, J. R., Cotos-Yañez, T. R., & Ruano-Ordás, D. (2019). A new semantic-based feature selection method for spam filtering. Applied Soft Computing Journal, Vol. 76, pp. 89-104.

[16] Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer networks fifth edition. In Pearson Education, Inc.

[17] Idris, I., Selamat, A., Nguyen, N. T., Omatu, S., Krejcar, O., Kuca, K., & Penhaker, M. (2015). A combined negative selection algorithm–particle swarm optimization for an email spam detection system. Engineering Applications of Artificial Intelligence, 39, 33-44.

[18] Jacobson, I., Booch, G., & Rumbaugh, J. (2012). The Unified Software Development Process. Prentice Hall.

[19] Pressman, R. S. (2010). Software engineering: a practitioner's approach. McGrawHill Higher Education.

[20] Haidar, H., Kolp, M., & Wautelet, Y. (2017). Goal-oriented requirement engineering for agile software product lines: an overview. Louvain School of Management Research Institute Working Paper Series, Louvain, Belgium, 1-36.

[21] Yu, E., Giorgini, P., Maiden, N., & Mylopoulos, J. (2011). Social Modeling for Requirements Engineering. The MIT Press.

[22] Horkoff, J., Aydemir, F. B., Cardoso, E., Li, T., Maté, A., Paja, E., ... & Giorgini, P. (2019). Goal-oriented requirements engineering: an extended systematic mapping study. Requirements Engineering, 24(2), 133-160.

[23] Cailliau, A., & Van Lamsweerde, A. (2012, September). A probabilistic framework for goal-oriented risk analysis. In 2012 20th IEEE International Requirements Engineering Conference (RE) (pp. 201-210). IEEE.

[24] Eric, S. Y. (2009). Social Modeling and i. In Conceptual modeling: Foundations and applications (pp. 99-121). Springer, Berlin, Heidelberg.

[25] AlhajHassan, S., Odeh, M., & Green, S. (2016, October). Aligning systems of systems engineering with goal-oriented approaches using the i∗ framework. In 2016 IEEE International Symposium on Systems Engineering (ISSE) (pp. 1-7). IEEE.

[26] Danesh, M. H., & Yu, E. (2014, June). Modeling enterprise capabilities with i*: reasoning on alternatives. In International Conference on Advanced Information Systems Engineering (pp. 112-123). Springer, Cham.

[27] Bertolini, D., Novikau, A., Susi, A., & Perini, A. (2006). Taom4e: an eclipse ready tool for agent-oriented modeling. issue on the development process. Technical report, Fondazione Bruno Kessler-irst.

[28] Jennings, N. (2000). On agent-based software engineering. Artificial Intelligence, Vol. 117, No. 2, pp. 277-296.

[29] Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). Developing Multi-Agent Systems with JADE. Wiley.

[30] Moreno-Espino, M., Carrasco-Bustamante, A., Rosete-Suárez, A., & Delgado-Dapena, M. D. (2013). Patrones de implementación para incluir comportamientos proactivos. Polibits, Vol. 47, pp. 75-88.

[31] Khamis, M. A., & Nagi, K. (2013). Designing multi-agent unit tests using systematic test design patterns-(extended version). Engineering Applications of Artificial Intelligence, 26(9), 2128-2142.