A Multi-Agent System for Addressing Cybersecurity Issues in Social Networks

Antonella C. Garcia[1,2,†], Maria Vanina Martinez[3,*,†], Cristhian A. D. Deagustini[1,†] and Gerardo I. Simari[2,4,†]

[1]Fac. de Cs. de la Administración, Universidad Nacional de Entre Ríos (UNER) and Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur (UNS), Argentina

[2]Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur (UNS) & Instituto de Ciencias e Ingeniería de la Computación (UNS–CONICET), Argentina

[3]Artificial Intelligence Research Institute (IIIA-CSIC), Bellaterra, Spain

[4]School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ 85281, USA

Abstract

The constant interaction of individuals on social networks and its implications in their lives, which affects decision-making and can even impact their mental and physical health, sparks interest in studying these environments from different perspectives. Of particular interest is the case of cyberattacks on these platforms, which includes not only low-level hacking activity but also other events like cyberbullying, grooming, and hate speech, among others. In this paper, we investigate the design and implementation challenges faced in the deployment of a multi-agent system that operates in social network platforms to prevent or mitigate cyberattacks through the processing of streaming information using belief revision operations. We instantiate the multi-agent system using the recently-proposed HEIST application framework, which guides the implementation of hybrid socio-technical systems with a focus on explainability, and discuss the main challenges in this process. We propose two possible approaches to building new knowledge dynamics operators: a cautious operator and a credulous operator, and evaluate the implications and challenges in each case. In this preliminary work, we adopt a non-technical approach, focusing on building a roadmap of the problems that need to be solved in order to develop a concrete solution, which is outside the scope of this paper. We conclude by suggesting the first steps towards achieving the objective.

Keywords

Belief Revision, Stream Reasoning, Cybersecurity, Social Platforms

1. Introduction

Currently, people base their daily activities on continuous direct or indirect interaction with information and news from the internet. This interaction generates large volumes of data, which are used for various purposes. Within these purposes, those with malicious intent deserve special attention, since they can cause harm in various areas, affecting the decision-making processes of many people worldwide. This motivates research and development closely related to efforts in cybersecurity, understanding this area according to the general conception[1] that includes not only low-level information security issues, but also human-centered factors such as cyberbullying, hate speech, and attacks against mechanisms that make online trust possible [1].

The U.S. Surgeon General recently published an Advisory on Social Media and Youth Mental Health, which states that:

> *"Extreme, inappropriate, and harmful content continues to be easily and widely accessible by children and adolescents. This can be spread through direct pushes, unwanted content exchanges, and algorithmic designs. In certain tragic cases, childhood deaths have been linked to suicide- and self-harm-related content and risk-taking challenges on social media platforms. This content may be especially risky for children and adolescents who are already experiencing mental health difficulties."* [2]

This underscores the significance of addressing cybersecurity issues in social media.

We now briefly discuss several problems that tend to occur in social platforms, and therefore represent key challenges in cybersecurity that motivate this work; this discussion is based on [1, 3].

Mis/Disinformation: Information in social environments is not always true; the spread of misleading or

ENIGMA-23, September 03–04, 2023, Rhodes, Greece

*Corresponding author.
† These authors contributed equally.
antonella.garcia@uner.edu.ar (A. C. Garcia); vmartinez@iiia.csic.es (M. V. Martinez); ariel.deagustini@uner.edu.ar (C. A. D. Deagustini); gis@cs.uns.edu.ar (G. I. Simari)

[1]Cybersecurity can be informally defined as "the protection of systems (including software, hardware, or humans) connected to the internet".

false information is very common, causing various levels of damage such as fake news, manipulated elections, and stock market bubbles. Misinformation has the ability to exploit vulnerabilities in social systems, and it can thus be classified as a cyber threat [4]. The challenge in these cases is to identify false or highly biased information, as well as users who contribute to its distribution. This is considered to be one of the most challenging threats in social environments, as there are no automated solutions that effectively mitigate this problem.

Cyber Attribution: Determining who is responsible for an attack—a problem commonly referred to as cyber attribution—is often a difficult proposition, and social platforms are no exception. It requires great effort to find and process evidence that can lead to attributions, taking into account that attackers often plant false evidence to cover their tracks or mislead law enforcement. Techniques such as reverse engineering, source tracking, and honeypots, among others [5], are commonly used to address cyber attribution.

Bot/Botnet Detection: Botnets are sets of connected and organized bots (automated software agents operating within a platform typically meant only for human users) designed to fulfill a specific objective. In this particular case, we are interested in malicious bots that, in social environments, have objectives such as trolling, spreading false information, or generating hate speech, among many others. To mitigate these actions, it is essential to determine whether an account on a social platform is being controlled by a person or a bot [6].

Adversarial Deduplication: It is not uncommon for multiple accounts on social media to belong to or be managed by the same real-world entity, often for the purposes of carrying out malicious or inauthentic actions; adversarial deduplication [7] seeks to determine which accounts are controlled by the same real-world entity. Various practices can be grouped under this category, such as sock puppets, Sybil attacks, and other malicious hacker activities. Actors in these scenarios usually seek to remain anonymous, but often also aim to be virtually identifiable in order to maintain a reputation within the community.

These problems, though different in nature, share the fact that solving or addressing them involve an effective use of incomplete, uncertain, and even biased knowledge. Therefore, cybersecurity is a broad area of research and practice that requires leveraging tools to address common issues for different types of attacks. Artificial Intelligence is useful in this context since it provides many basic tools that can be applied on their own or in combination towards the development of an effective and efficient solution. The model we propose in this work is intended to be used in the context of social platforms in general, allowing for systematic processing of a larger amount of data than what humans are capable of. Our model is an instantiation of the HEIST (Hybrid Explainable and Interpretable Socio-Technical Systems) [8] application framework, which provides the foundations developing systems that are capable of offering explanations about the decisions made.

The main goal of our model, a multi-agent system (MAS) of Supervisor Agents, is to supervise social platforms, seeking to detect malicious content and activities and respond so as to avoid or mitigate their effect. We now discuss two motivating domains within social platforms that allow us to introduce the main functional requirements for our approach.

Medical content. In this context a supervising system should be able to distinguish between a post with sexual content and a post that mentions sexual matters in a medical/health context. For example, it should prevent censorship of content related to breast cancer awareness—this would reduce false positives of sexual content on the social network. It could adjust alerts for dangerous/suspicious profiles against accounts that are whitelisted because they are known to disseminate alerts, educational content, awareness campaigns, etc. Currently, campaigns for breast cancer prevention cannot be freely shared as social networks censor any image of a breast, hindering the dissemination of proper self-examination and warning signs.

Parental control. Supervising systems can also be leveraged as tools that can be applied by users themselves in specific platforms to exert personalized control. Such a system could be conceived as an extension to be used "on top of" the social platforms, as is the case with Google's Family Link[2]. For instance, an application for mobile devices could, based on what is displayed on the screen, show alerts or—in the case in which the user is a minor—send notifications to guardians. In this case, the system would have full access to all of the device's activity; if any suspicious behavior is detected, it can activate alerts on the same device or on devices owned by guardians. Another interesting functionality to consider is the possibility that, with prior authorization, the device's supervising system send alerts to devices owned by children/-guardians within the same class/school/interest group. This would generate what we will refer to as a *news item* for all such devices, and each corresponding agent would have the chance to decide what to do with that new knowledge.

The main contributions of this work are the following: (i) The proposal of a multi-agent system designed to address issues related to malicious behavior in social platforms; the system is based on the instantiation of

[2]https://families.google/familylink/

a recently-proposed application framework for XAI in socio-technical systems; (ii) the definition of a novel kind of belief dynamics operator to guide the flow of knowledge within the framework, which we call stream-based belief revision; and (iii) the identification of the hurdles that must be overcome for the development of effective and efficient stream-based revision operators. Though preliminary in nature, this article is meant to serve as a roadmap for ongoing and future research in the area of cybersecurity in social platforms.

2. Preliminary Concepts

We now recall the basic details of two models that we leverage in the development of our Supervisor Agent framework.

2.1. Network Knowledge Bases

The work developed in [9, 10] on Network Knowledge Bases (NKBs) adopts the typical model of social networks as sets of agents with various relationships among them; however, in NKBs each agent (a user in the social network) has its own knowledge base. Each agent is represented as a vertex, and relationships are represented as arcs between the vertices. NKBs can thus be seen as complex multilayer networks that allow representing the individual beliefs of each network node, as well as multiple attributes of the nodes and their relationships, affording the possibility of combining models for more than one social platform. Feeds—the pieces of information that each user sees when engaging with the platform—are modeled as news items that represent the source, content, and an indication of whether the user who posted it is signaling an addition or deletion to their own KB. In [11], the authors show that the model can be used for predicting users' reactions to the content in their Twitter feed.

For the purposes of this work, we will adopt a slightly modified model since we are not assuming that we have full access to users' knowledge bases. Instead of addressing the local belief revision problem (as is done in [9, 10]), we focus on analyzing the activity visible by a companion app as discussed in the examples above, which includes a rich variety of actions and content (cf. Section 3). As an additional novel aspect in this work with respect to the original research line on NKBs, we also take into account time, which is central to effectively tackle problems arising in cybersecurity domains [3].

2.2. The HEIST Application Framework

HEIST (which stands for *Hybrid Explainable and Interpretable Socio-Technical Systems*) [3, 8] is an application framework[3] that aims to guide the implementation of hybrid[4] socio-technical systems that require explainable outputs.

We briefly describe each of the six components, referring the reader to [8] for a full description. For an illustration of the architecture, see Figure 4 (Section 4).

Data Ingestion: Handles the integration of data sources, addressing basic issues like data cleaning, schema matching, inconsistency, and incompleteness management. It also deals with higher-level challenges such as trust and uncertainty management, ensuring the proper handling of heterogeneous data.

Subsymbolic Services: This module focuses on tasks that are best solved using data-driven (machine learning) services. Having such tools isolated in a module helps to identify specific application scenarios for each service, facilitating faster implementation, providing alternative implementations, and the generation of explanations.

Symbolic Reasoning: High-level reasoning is key for addressing general problems. This module, which serves as the core of the framework, leverages preprocessed data from the Data Ingestion Module and outputs from the Subsymbolic Services Module. Rule-based systems are commonly employed here to perform complex tasks like combining low-level data and knowledge, or providing responses based on well-defined reasoning mechanisms over structured knowledge. The reasoning processes implemented in this module are essential for answering user queries.

Explanations: Generates different types of explanations associated with query answers. It leverages outputs from the Symbolic Reasoning module (via the Query Answering module) and the Subsymbolic Services module.

Human in the Loop: In socio-technical environments, the system's effectiveness relies on adequately addressing user demands. This module aims to enhance system performance by incorporating iterative feedback from human users[5]. This feedback includes queries, responses, explanation requests, explanation ratings, and utility-based classification of data sources, among other options.

Query Answering: Focuses on answering user queries by coordinating the execution of all other modules.

Next, we discuss the building blocks that will be used later on (in Section 4) to instantiate HEIST to obtain a specific architectural model for our Supervisor Agent framework.

[3]A general-purpose software structure designed to facilitate the development of applications via instantiations or extensions.
[4]The term "hybrid" refers to the combination of data-driven and symbolic tools.
[5]Reinforcement Learning with Human Feedback [12, 13], is a special case of HITL.

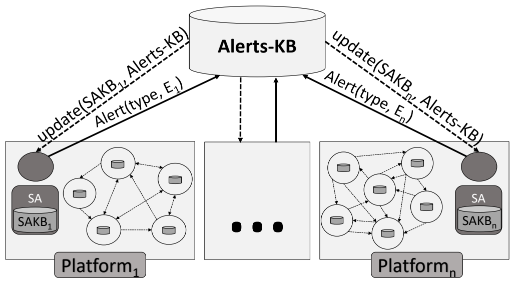

Figure 1: Proposed MAS for addressing cybersecurity issues in social platforms. Each platform has a dedicated *Supervisor Agent* that monitors it and interacts with central repository Alerts-KB.

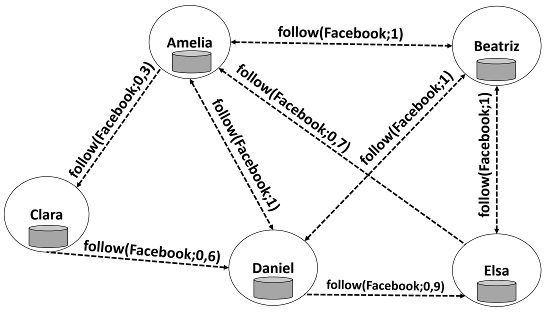

Figure 2: A simple social network.

3. Towards a Supervisor Agent Framework

In this section, we develop the proposed model, a multi-agent system aimed at addressing cybersecurity challenges in social networking environments; Figure 1 provides an overview of the proposed system. Each agent in the system is responsible for supervising a particular social network, and exchanges information through a centralized knowledge base called *Alerts-KB*, which serves as a repository for the agents to have up-to-date information on security alerts arising from different social networks. *Alerts-KB* consolidates knowledge related to cybersecurity events, encompassing various social networks regardless of their specific characteristics. This knowledge repository is reviewed and updated based on the information shared by each agent in the system. As a result, the agents exchange information that enables them to proactively address potential cybersecurity issues that have already been identified in other networks.

Each supervisor agent maintains its own knowledge base about the social network it operates in. As we will discuss below, agents engage in belief revision processes in order to update their own KB. Continuous belief revision processes performed by agents enable them to make informed decisions and implement appropriate actions in a timely manner.

Example 1. *Consider the context of a social network such as Instagram, Twitter, or Facebook. A recurring security issue is the distribution of sexually suggestive images of the human body. There are various security measures in place that filter out images containing certain sensitive content, such as uncovered body parts, including nipples. While these measures aim to address security concerns, they also hinder the positive uses of such images, such as breast cancer prevention campaigns.*

Consider the scenario where a user called "Medical Center", validated as a health authority, posts a video showing a breast as part of a breast cancer prevention campaign. In the proposed model, the intelligent agent can consult the information about this user in the knowledge base and determine that they consistently share legitimate medical content. Based on this knowledge, it would be best not to censor this particular post, since it would allow for more effective prevention campaigns.

3.1. Modeling Social Networks

In order to address cybersecurity issues in social platforms, we must be able to model users and their relationships, for which we first adopt the way social networks are modeled in [10]. According to this definition, a social network is a tuple consisting of four elements: a finite set of vertices, a finite set of edges, a vertex labeling function, and an edge labeling function:

A **Social Network** is a 4-tuple $(V, E, l_{vert}, l_{edge})$ where:

1. V is a finite set whose elements are called *vertices.*
2. $E \subseteq V \times V$ is a finite set whose elements are called *edges.*
3. $l_{vert} : V \to 2^{L_V}$ is a *vertex labeling function*, where L_V is a set of vertex labels.
4. $l_{edge} : E \to 2^T$ is an *edge labeling function*, where $T = \{\langle b, w\rangle \mid b \in L_E, w \in [0, 1]\}$ and L_E is a set of edge labels.

Next, we provide a simple example of a social network.

Example 2. *Consider the social network depicted in Figure 2, where the users are Amelia, Bautista, Clara, Daniel, and Elsa. The graph encodes relationships between different users. For example, Elsa follows Amelia and Bautista, and she is followed by Bautista and Daniel. The relationships may not be reciprocal, as Clara follows Daniel, but Daniel does not follow Clara.*

Considering the specific characteristics of social networks, it is crucial to model the different users and their relationships, i.e., the edges and vertices of the network, as well as the various interactions among them. The latter are events within the social network, and thus we must model a continuous sequence of events that contain relevant information for the system, which we refer to as a *data stream.*

Events within the social network can take different forms. We now mention the most common events in these environments and their specific characteristics. However, a particular social network may have additional types of events beyond those mentioned here.

Post. Each time a user creates new content, a Post event is created. This event reaches the vertices that are connected to the posting vertex. It consists of: `event ID`, `source (publisher)`, `text`, `multimedia element`, `set of tags`, and `timestamp`.

Share. Based on a Post event, a user can generate a Share event, which means sharing the original post with or without adding new data. Sharing increases the reach of the original post to the vertices connected to the Share-generating vertex. It contains the same elements as a Post event, plus a pointer to the ID of the original post: `event ID`, `source (Share generator)`, `text`, `multimedia element`, `set of tags`, `pointer to original post ID`, and `timestamp`.

Reaction. Refers to the reactions to a post, such as "like", "love", or other types of reactions depending on the platform. This event does not increase the reach of the original post but can influence its visibility to a greater number of users in their feeds. It consists of: `event ID`, `source (Reaction generator)`, `reaction type`, `pointer to original event ID`, and `timestamp`.

Comment. A comment is the addition of text to a previously generated post, either by the same user or another user. The event data includes: `event ID`, `source (Comment generator)`, `comment text`, `pointer to original event ID`, and `timestamp`.

Connection. Connections between users can be created or removed—each such occurrence is encoded as a Connection event (if two nodes are already connected, a Connection event encodes the removal of the edge). The event data includes: `event ID`, `source`, `target`, and `timestamp`.

Figure 3 illustrates these events in the context of the network from Figure 2.

As mentioned above, we adapt here the NKB model presented in [9, 10]:

NKBs. A Network Knowledge Bases (NKB) is a 5-tuple $(V, E, l_{vert}, l_{edge}, K)$ where the first four elements comprise a social network, and K is a mapping assigning a knowledge base to each vertex. Given v, $K(v)$ is called the knowledge base associated with vertex v.

In our model, we make a slight modification: whereas in NKBs as proposed by Gallo et al. the KB associated with each vertex is meant to encode the corresponding user's private beliefs, here it will group the posts made by that vertex and the interactions with posts from other vertices; $K(v)$ thus contains the events generated by that specific vertex; as shown in Figure 3, and as we discuss below, these KBs will be referred to as "SAKBs". Another difference with the original NKB model is that here we explicitly model time via the assignment of timestamps to each event, as described above. Each event reaching vertices through their feeds is assumed to do so immediately (based on the timestamp of the original event).

Example 3. *Consider again the network from Figure 2. Based on its structure and available event data, we can observe the reach of events that occur. For example, if Bautista generates a* ***Post*** *event, it will reach the vertices connected to it: Amelia, Daniel, and Elsa. Furthermore, if Daniel shares the post by Bautista, it will also be seen by Amelia and Clara, and Bautista will see the event with the additions made by Daniel.*

With the necessary tools in place for understanding the structure of the social network and characterizing specific aspects of the environment, we can proceed with the definition of our framework.

3.2. Supervisor Agents

We now discuss the requirements for Supervisor Agents (SA, for short) in our model—in Section 4 we provide an architectural design based on the HEIST application framework. Each SA's objective is to provide recommendations and/or make cybersecurity decisions based on the observation of the social network's structure that it is monitoring, and the data stream generated by its events. SAs operate in an alert state within the social network they supervise, and have access both to the NKB model of the corresponding social network and its data stream, so it can access all the events generated by vertices in the network.

Each agent SA_i maintains its own KB, which we will call $SAKB_i$, containing information about the social network and the security alerts occurring in that particular platform. $SAKB_i$ receives information from the NKB model of the social platform i and its data stream. As we will discuss in Section 5, the SA must dynamically keep its KB up to date based on this knowledge. Furthermore, the SA also updates its KB with security alerts from other social platforms sent to *Alerts-KB* by other agents. We propose the application of belief revision operators for these purposes.

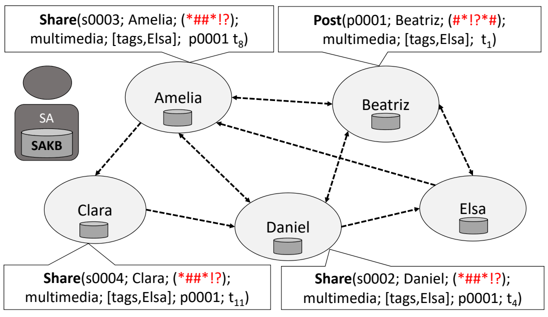
Share(s0003; Amelia; (*##*!?); multimedia; [tags,Elsa]; p0001 t8)
Post(p0001; Beatriz; (#*!?*#); multimedia; [tags,Elsa]; t1)
SA
SAKB
Amelia
Beatriz
Clara
Daniel
Elsa
Share(s0004; Clara; (*##*!?); multimedia; [tags,Elsa]; p0001; t11)
Share(s0002; Daniel; (*##*!?); multimedia; [tags,Elsa]; p0001; t4)

Figure 3: Events in social network example

Example 4. *Consider the network in Figure 3. The agent analyzes the data stream and observes that Beatriz made a post tagging Elsa that contains offensive language.*

Initially, the SA may decide to remain alert without taking action. At a later time, the SA observes that Daniel makes offensive comments against Elsa in the original post by Beatriz. Subsequently, Amelia and Clara share Beatriz's post, adding offensive comments. Based on this behavior, the SA raises an alert and issues warnings to the involved users.

Making the decisions described in Example 4 is the central problem faced by each SA—there is a wide range of possibilities, and exploring this in detail is outside the scope of this paper. Different alternatives can be formalized as policies that the SA can carry out within its network. Examples include simple approaches based on thresholds (for instance, a three-strike rule that issues alerts after allowing two violations of a posting policy), or more complex schemes such as implementing a user classification mechanism, for instance based on user types as described in [9], that can be used to predict behaviors of interest.

Among the tasks that agents must carry out, we have: (i) maintaining an updated SAKB specific to the social network it operates in; (ii) detecting potential threats or suspicious behaviors within the platform; (iii) sending notifications for security-based decision-making, (iv) notifying the individuals involved and the responsible party about security measures taken, (v) sending updates of new security alerts to the *Alerts-KB*, and (vi) revising their knowledge based on updates to the *Alerts-KB* made by other SAs.

Based on the available knowledge, the SA could predict the viral effect of a post and recommend or implement security actions such as detecting negative viral effects, suspending users, managing the relevance level of posts, nullifying posts, or removing fake accounts. Several issues need to be addressed to enable SA's to carry out such tasks. In the following section we provide details regarding the use of HEIST to implement supervisor agents, and then in Section 5 we focus on how the data stream of the social network will be processed, and how the belief revision problem for each SA's KB can be formulated.

4. The HEIST-SA System

We now present HEIST-SA, an extension of HEIST [3, 8] that yields an architectural design that guides the implementation of Supervisor Agents. We choose this model for its flexibility in combining both symbolic and sub-symbolic tools, and because it explicitly considers explanations for query answers, which is central to cybersecurity applications.

In the following, we discuss the modules described in Section 2 in the specific context of a use case based on a typical social platform where users generate events as described in Section 3 that must be processed in HEIST-SA.

Data Ingestion. This component receives all the activity from the social platform, which includes all the events generated by each vertex of the network. The stream activity is continuous and unbounded, so this module must deal with aspects related to stream processing such as windowing, load shedding, etc. [14]. As these tasks are completed, the module divides the data flow into windows, which are fed to the Symbolic Reasoning module.

Sub-symbolic Services. This module provides support in the form of basic services, such as user classification to predict certain behaviors in users [11], determining if posts contain hate speech, predicting virality of posts, etc. This will allow making more relevant security decisions, deploying specific services depending on the context, such as image, audio, or video-based classifiers.

Symbolic Reasoning. This module takes input from the Data Ingestion module and is thus responsible for implementing the stream reasoning [15] aspects that we discuss in more detail in Section 5, as well as maintaining the agent's SAKB. Concretely, the SA must perform stream reasoning-based belief revision in its SAKB as events occur in the social platform, seeking to detect malicious behavior. Specifically, the module receives a window from the stream, with which the NKB model is updated at the same time that the sub-symbolic services are applied to the events of that window. Rule-based approaches such as [16], or other formalisms based on computational logic, are good candidates for implementing such functionalities.

Query Answering. The QA module is responsible for user interaction—it coordinates the other modules to respond to user queries. Different types of users need to be distinguished, including regular users, expert users who are part of the working team, cybersecurity experts, and administrators of various groups within the social network, such as groups or pages, as explored in [3].

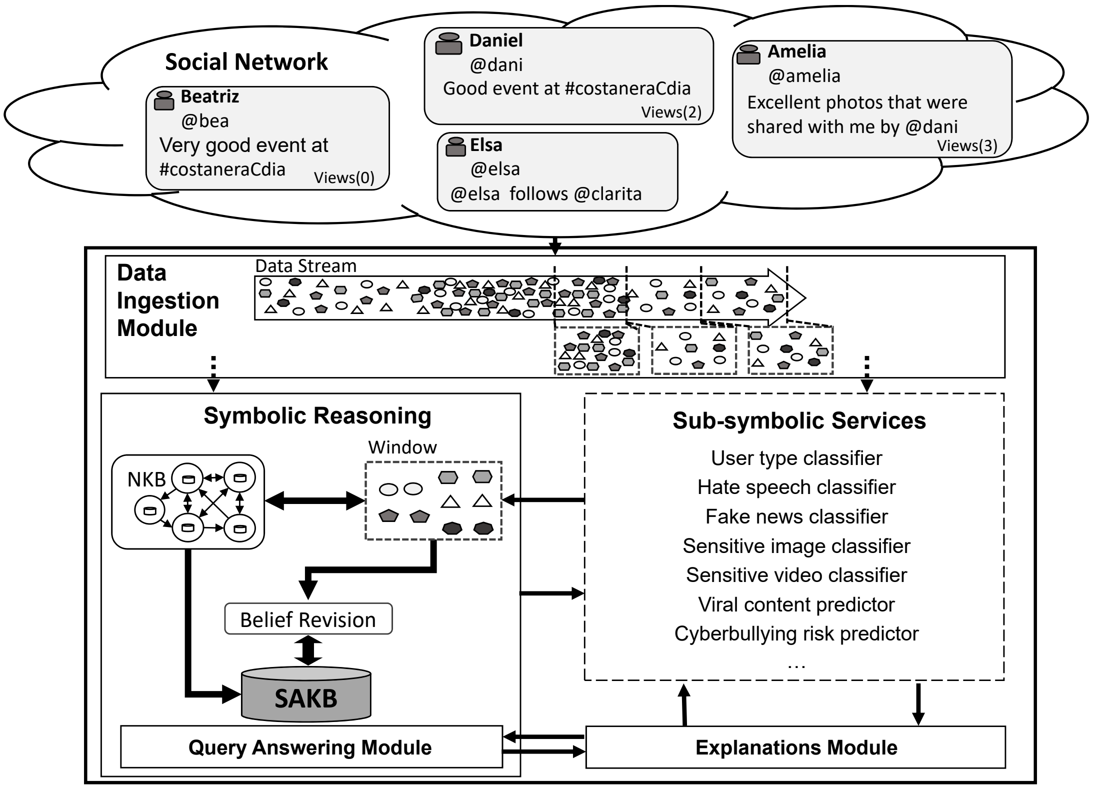

Figure 4: Overview of HEIST-SA

Explanations. This module provides explanations for the decisions made or actions suggested by the SA. As the agent's decision-making is governed by the Symbolic Reasoning module, the Explanations module works in conjunction with the Query Answering module (and the Sub-symbolic Services module if it is used) to derive explanations for the security decisions made so they can be evaluated by different kinds of users.

Human in the loop. This module is responsible for recording and responding to user feedback, which may involve updating the SAKB, reissuing a query, maintaining statistics of interest, etc. It also manages the type of explanations presented to each type of user in the social network, as discussed above.

Next, we will focus on the challenges specific to the Symbolic Reasoning module, which will carry out belief revision tasks based on inputs from the data stream.

5. Stream-based Belief Revision

Belief revision is the problem of deciding how to react to epistemic inputs to a knowledge base [17, 18]. We now discuss how our agents perform belief revision tasks based on epistemic inputs coming from the data stream. First, we provide a more concrete definition of data stream.

Data Stream: The data stream produced by a social network S is a continuous, a priori unbounded, sequence of social network events, where each event is generated by a vertex belonging to S.

These data streams contain information that needs to be processed promptly to extract knowledge as soon as relevant information becomes available [14]. The distinctive feature of data streams is that in general we cannot assume that their elements can be stored for later use. As a first step towards solving this problem, we need to address the processing of the data stream that the Data Ingestion module must perform.

In the rest of this section, we first discuss basic issues that need to be addressed when considering implementations in this setting, then formalize the statement of the problem we wish to solve, discuss challenges that arise, and propose two preliminary proposals for solving our problem.

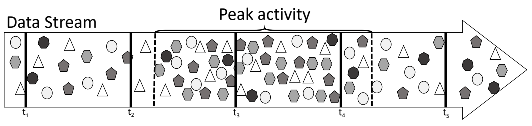

Figure 5: Sliding Pane Logical Window

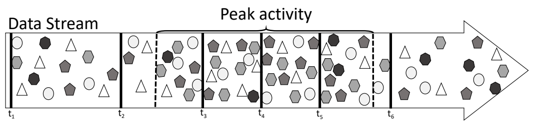

Figure 6: Sliding Pane Physical Window

5.1. Information Stream Processing

Given the nature of social networks and the large volume of data generated in short periods of time, conditions must be established to ensure that the processing of the data can be carried out in the best possible way. To this end, it is necessary to evaluate how stream reasoning [15] can be carried out in such settings so that the agent is capable of making the best possible use of the data stream produced by the associated social network, aiming to gain as much information as possible and respond to events of interest effectively. Next, we will consider different aspects related to handling the data stream from the social network that will enter the Data Ingestion module. As mentioned, data streams may contain inconsistencies, so we need to ensure that these are also processed in a way that provides the best handling of the data for the system's objective. Since we cannot store all the incoming data, we must "discretize" the stream. We now define various aspects of the data stream processing effort.

Firstly, we need to define the **data model**, which refers to how information is represented. The data stream is comprised of events represented as tuples whose structure depends on the specific type of event, as discussed in Section 3. In summary, we have the following five types of event:

- **Connection**: $\langle$*event_id*, *user*$_1$, *user*$_2$, *time*$\rangle$
- **Post**: $\langle$*event_id*, *user_src*, *text*, *media_elem*, *tag_set*, *time*$\rangle$
- **Share**: $\langle$*event_id*, *user_src*, *text*, *media_elem*, *tag_set*, *post_id*, *time*$\rangle$
- **Reaction**: $\langle$*event_id*, *user_src*, *react_type*, *orig_event*, *time*$\rangle$
- **Comment**: $\langle$*event_id*, *user_src*, *text*, *orig_event*, *time*$\rangle$

Once the data model is established, the next relevant aspect for data stream processing in our context is to consider **windows**, which is the construct typically used to discretize streams [14]. In our case, they will allow us to limit the scope of the revision operators.

Window: A subset of events from a data stream selected according to a given criterion.

Windows can be either **logical**, which implement a selection criterion based on bounds over timestamps, or **physical**, which work with prefixed bounds on the number of tuples to be considered. A second factor that needs to be addressed is how the bounds of the windows will be updated, which corresponds to how the window "moves" with time. Here, we will adopt the more general case of the *sliding* window, where both lower and upper bounds advance with the arrival of new items or the passage of time. A special case of this is the *tumbling* window, in which all items change each time the bounds are updated.

We now analyze the strengths and weaknesses associated with two types of windows, both of which are valid options for discretizing the social platform data streams.

Sliding pane logical window: In this case, windows are specified by a fixed time interval, allowing us to know the processing schedule—Figure 5 illustrates this case. A problem with this approach is that windows may become overloaded with data during peak activity, which can result in processing time that is greater than the validity of the window itself.

Sliding pane physical window: In this case, we cannot predict the speed at which the window is updated since it depends on the number of tuples that arrive in the stream (cf. Figure 6). The advantage, of course, is that by knowing the number of items in each window, we can estimate how long it will take to process each window. On the other hand, we do not know now the frequency at which the window will be updated, which can again lead to problems during peak activity, as a large number of elements need to be processed in a short period of time and this may not be possible.

These two cases show that the processing model needs to define a load shedding policy [14], which essentially decides how to deal with data bursts or spikes in the stream by ignoring some of the tuples. In the general case in which several social platforms are involved, it seems unavoidable that such a policy be used since the volume of data streams varies depending on the number of active users on the platform at a given time, resulting in increased data volume during peak activity. In future work, we plan to study more precisely under which conditions each of these discretizations is most suitable for our system, what the impact of each one is in terms of effectiveness, and whether hybrid solutions might be possible.

Other challenges related to stream processing. There are several challenges that need to be addressed in this context. First, we need to establish the relationship between items and the passage of time so that some kind of order among the items can be established. In cybersecurity applications, it is crucial to know when an incident originated and who replicated or amplified it. In our case, though items have timestamps, these may be fixed either at the origin by the social platform itself or, when this is not the case, by the Data Ingestion module as events are read. Related to this issue, in stream processing items may arrive out of order, meaning that we receive an item after receiving others that have a more recent timestamp. This has been studied recently in the Databases community [19], and it is important to study it in this setting as well.

Other important challenges involve providing support for uncertainty. Agents should be able to handle inputs with uncertainty, as they may for instance encounter insults in a post that may not necessarily indicate an attack but rather a playful interaction between two users. Additionally, SAs need to support outputs with uncertainty since alerts generated may not always indicate a real problem.

5.2. Belief Revision: Problem Statement

Let K be an SAKB and w a window belonging to data stream *DS* of the social network *SN*. We wish to define a *stream-based belief revision operator* ϵ as a function that takes K and w and produces a new SAKB K':

$$K' = \epsilon(K, w)$$

That is, K' is obtained by applying operator ϵ to the original K with epistemic inputs from w arising from data stream *DS*.

Assuming a scenario with no computational resource limitations, applying ϵ would result in a consistent and updated K that could be handled with existing belief revision operators. However, this ideal scenario is not possible since in general we may not have enough time to process each window. Our system must therefore have principled mechanisms for deciding which elements in the window will not processed, and for this to be effective we must study how that impacts the result.

In the following section, we discuss several challenges that arise in practice: (i) real-time processing, (ii) out-of-order events, and (iii) event overload during peak activity. Given that classical belief revision operators—such as [17, 20, 21, 22, 23]—are not designed to work in this setting, their direct application would lead to one or more of such requirements to not be met.

5.3. Challenges in Stream-based Belief Revision

Though in abstract the stream-based belief revision problem is straightforward—simply apply a belief revision operator to the current window and continue doing so each time it is updated—things fall apart when we consider the challenges described above that arise when processing data streams. For instance, events in the stream may arrive out of order; applying an operator with incomplete information can generate different results than if we have all the information in a timely manner, and by the time the remaining data arrives it may be too late to correct the mismatch. Therefore, policies for handling out of order events will play a crucial role in deriving effective solutions to this problem, and their properties need to be thoroughly studied.

Since belief revision operators tend to be computationally costly [24], this leads to the problem of overload in windows where the volume of events is large or where windows are updated within short periods of time. While the operator processes the current window, an update may occur and the Supervisor Agent in this case would fall behind, causing a bottleneck in the operator and an outdated knowledge base. There are essentially three ways in which we may deal with this situation. First, as discussed above, we may implement a load shedding policy that simply chooses which elements are ignored so that the operator finishes in time. A second option is to develop a suite of operators, ranging from a lightweight option suitable for heavy loads to an ideal one that may be applied when time is available. Finally, as a compromise solution, we may consider developing something akin to "second order windows" where unprocessed elements are saved for later processing—though additional cost is incurred in terms of space, and results will not be available in a timely manner, correctness is not sacrificed.

Next, we describe a preliminary proposal in the form of two possible options for simple operators to address these challenges.

5.4. First Steps towards a Solution

To tackle the challenges identified above, we may consider two possibilities: (i) ignoring the unprocessed events (i.e., not addressing them at all), and (ii) allowing the KB to accept inconsistency by incorporating the unprocessed events.

These two options represent distinct semantic decisions that we discuss below. Note that depending on the specific system load there may be windows in which all events can be processed resulting in the updated SAKB. We focus on the interesting case that corresponds to situations where the available time for window processing may be insufficient to process all events contained in the

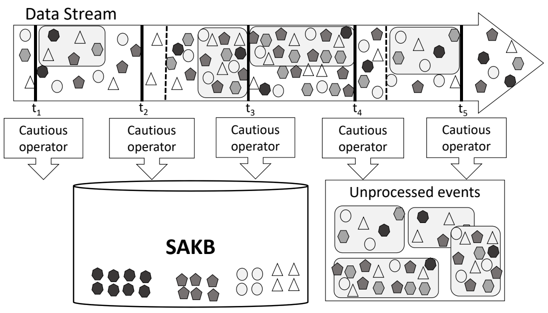

Figure 7: Discarding unprocessed knowledge from window.

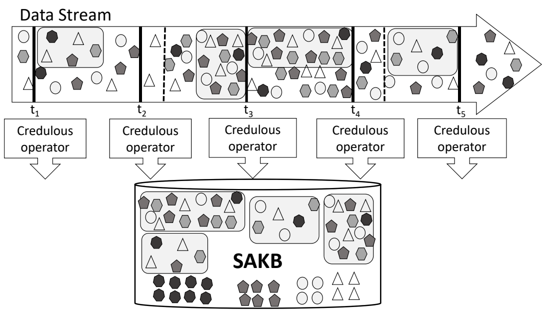

Figure 8: Incorporating unevaluated knowledge into K.

window.

In the following, in order to be able to use classical revision operators, we simplify the knowledge representation model and assume SAKBs and w are formulas in a propositional language $\mathcal{L}$. Furthermore, let $P(w)$ represent the set of elements in window w that have been processed so far, and $U(w) = w \setminus P(w)$ represent the set of elements in window w that have *not* been processed.

Option 1: *Ignore unprocessed events*

Let "$*$" be a classical multiple revision operator; a cautious operator Υ can be defined as follows:

$$\Upsilon(K, w) = K * P(w)$$

If the unprocessed knowledge ($U(w)$) from the current window is discarded, the SAKB will be consistent, and consequently, queries can be resolved using classical reasoning. This simplifies the processing of the SAKB, but it results in partial knowledge, since a significant number of events may be left out. This could include a multitude of attacks or events that could indicate potential threats, which would not be processed by the SA. This is illustrated in Figure 7.

Consider a scenario where a user on the social network is being targeted by other users, such as a case of cyberbullying. Since the SA does not process all of the events, it may only see a fraction of the comments and overlook the attack. Let's say there were 50 comments in the window, but the agent only processed 10 of them, along with other unrelated events. If we consider an agent that takes action when it detects 30 offensive comments, by processing only 10 comments, it would miss identifying the attack. This simple example highlights the drawbacks of this decision.

Option 2: *Allow inconsistency by incorporating the unprocessed events*

Let "$*$" be a classical multiple revision operator, and "$+$" be a classical expansion operator; a credulous operator Φ can be defined as follows:

$$\Phi(K, w) = K * P(w) + U(w)$$

In this case, we incorporate the unevaluated knowledge into the SAKB, and it may therefore become inconsistent as a result (cf. Figure 8). Though this deals with the problem initially, it pushes the issue to the Query Answering module, which must apply costly inconsistency-tolerant methods in order to function properly.

By incorporating unevaluated data into the SAKB, we may include information that appears to be a threat but is actually not. For example, we may have comments in a post that contain inappropriate words, but due to the existing relationship between the involved parties and the stored knowledge, it may be consistent with their way of communicating. These comments could be insults used in a playful manner and therefore do not represent a real attack. Since the SA could not evaluate this event and it was incorporated directly into the SAKB, this incident may play a role it wouldn't have if the window had been processed fully.

In this case, the problem is pushed to the QA module since decisions made by the agent will be "contaminated" by unevaluated data. For instance, it would be necessary to define the level of confidence in the information provided by the AS. We could establish a semantics based on trust for conflict resolution. To achieve this, a measure indicating the level of confidence should be assigned to each piece of information and updated in each application of the revision operator. One possibility would be to consider a form of stratified SAKB [25, 23], where all the processed data forms the "hard" layer with a high level of confidence, and then having layers containing the unevaluated data from each window, potentially with different levels of confidence associated.

These operators could address the issue of event overload during peak activity, allowing (near) real-time processing. However, the policy for handling out-of-order events still needs to be defined. To implement these operators, as future work, we need to define new postulates and constructs that allow us to apply belief revision in these environments. One possibility is to consider the possibility of defining postulates that do not necessarily have to be fully satisfied, but rather think about degrees of satisfaction that provide flexibility to better characterize real-world environments.

6. Conclusions and Future Work

In this work, we discuss the need to address cybersecurity issues in social networks, and develop a preliminary proposal for a multi-agent system based on the instantiation of a recently-proposed application framework. Since the system works with data streams comprised of the events that occur within the social platforms being inspected, we argue that stream-based belief revision operators are at the heart of effective solutions to this problem. We discuss the need to make different decisions regarding the way in which the data stream is processed via window operators, such as the type of window to use and the implications of defining their size and update criteria, and identify several challenges that must be overcome in the path to effective and efficient solutions. We illustrate these challenges by presenting two operators based on classical belief revision, showing how they may not behave as expected.

Future work in this research line involves defining postulates that are especially designed for this setting, as well as constructions of operators based on (subsets) of these postulates. Given the practical motivation of this work, we also intend to design and carry out adequate empirical studies to validate the results obtained. Finally, we believe that this line of research—as well as any software products that are built based on it—would benefit from following guidelines towards achieving *trustworthiness by design*[6].

Acknowledgments

This work was funded in part by Universidad Nacional del Sur (UNS) under grant PGI 24/ZN057, Secretaría de Investigación Científica y Tecnológica FCEN–UBA (RESCS-2020-345-E-UBA-REC), Universidad Nacional de Entre Ríos under grant PDTS-UNER 7066, CONICET under the grant PIP (11220200101408CO), and Agencia Nacional de Promoción Científica y Tecnológica, Argentina under grants PICT-2018-0475 (PRH-2014-0007) and PICT-2020 SERIE A-01481.

[6]http://tailor.isti.cnr.it/handbookTAI/TAILOR.html

References

[1] G. I. Simari, From data to knowledge engineering for cybersecurity, in: S. Kraus (Ed.), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019, ijcai.org, 2019, pp. 6403–6407.

[2] OSG, Social media and youth mental health: The US Surgeon General's advisory, Online (accessed 9-June-2023) – https://www.hhs.gov/sites/default/files/sg-youth-mental-health-social-media-advisory.pdf, 2023.

[3] J. Paredes, J. C. Teze, M. V. Martinez, G. I. Simari, The HEIC application framework for implementing xai-based socio-technical systems, Online Soc. Networks Media 32 (2022) 100239. URL: https://doi.org/10.1016/j.osnem.2022.100239. doi:10.1016/j.osnem.2022.100239.

[4] K. M. Caramancion, An exploration of disinformation as a cybersecurity threat, in: 2020 3rd International Conference on Information and Computer Technologies (ICICT), IEEE, 2020, pp. 440–444.

[5] E. Nunes, P. Shakarian, G. I. Simari, A. Ruef, Artificial intelligence tools for cyber attribution, Springer (2018).

[6] E. Ferrara, O. Varol, C. Davis, F. Menczer, A. Flammini, The rise of social bots, Communications of the ACM 59 (2016) 96–104.

[7] J. Paredes, G. I. Simari, M. V. Martinez, M. A. Falappa, First steps towards data-driven adversarial deduplication, Information 9 (2018) 189.

[8] J. C. L. Teze, J. Paredes, M. V. Martinez, G. I. Simari, Engineering user-centered explanations to query answers in ontology-driven socio-technical systems, Semantic Web (2023) 1–30 (*In Press*). URL: https://content.iospress.com/articles/semantic-web/sw233297. doi:DOI:10.3233/SW-233297.

[9] F. R. Gallo, G. I. Simari, M. V. Martinez, M. A. Falappa, N. A. Santos, Reasoning about sentiment and knowledge diffusion in social networks, IEEE Internet Comput. 21 (2017) 8–17.

[10] F. R. Gallo, G. I. Simari, M. V. Martinez, N. A. Santos, M. A. Falappa, Local belief dynamics in network knowledge bases, ACM Transactions on Computational Logic (TOCL) 23 (2021) 1–36.

[11] F. R. Gallo, G. I. Simari, M. V. Martinez, M. A. Falappa, Predicting user reactions to twitter feed content based on personality type and social cues, Future Generation Computer Systems 110 (2020) 918–930.

[12] W. B. Knox, P. Stone, Augmenting reinforcement learning with human feedback, in: ICML 2011 Workshop on New Developments in Imitation

Learning (July 2011), volume 855, 2011, p. 3.

[13] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, Advances in neural information processing systems 30 (2017).

[14] G. Cugola, A. Margara, Processing flows of information: From data stream to complex event processing, ACM Computing Surveys (2012).

[15] E. Della Valle, S. Ceri, F. Van Harmelen, D. Fensel, It's a streaming world! reasoning upon rapidly changing information, IEEE Intelligent Systems 24 (2009) 83–89.

[16] A. Ronca, M. Kaminski, B. C. Grau, I. Horrocks, The delay and window size problems in rule-based stream reasoning, Artificial Intelligence 306 (2022) 103668.

[17] C. E. Alchourrón, P. Gärdenfors, D. Makinson, On the logic of theory change: Partial meet contraction and revision functions, The journal of symbolic logic 50 (1985) 510–530.

[18] P. Gärdenfors, Knowledge in flux: Modeling the dynamics of epistemic states., The MIT press, 1988.

[19] M. Fragkoulis, P. Carbone, V. Kalavri, A. Katsifodimos, A survey on the evolution of stream processing systems, arXiv preprint arXiv:2008.00842. (2020). doi:https://doi.org/10.48550/arXiv.2008.00842.

[20] S. O. Hansson, Kernel contraction, Journal of Symbolic Logic 59 (1994) 845–859. doi:10.2307/2275912.

[21] L. Amgoud, S. Kaci, An argumentation framework for merging conflicting knowledge bases: The prioritized case, in: Symbolic and Quantitative Approaches to Reasoning with Uncertainty: 8th European Conference, ECSQARU 2005, Barcelona, Spain, July 6-8, 2005. Proceedings 8, Springer, 2005, pp. 527–538.

[22] J. P. Delgrande, D. Dubois, J. Lang, Iterated revision as prioritized merging., KR 6 (2006) 210–220.

[23] M. A. Falappa, G. Kern-Isberner, M. D. Reis, G. R. Simari, Prioritized and non-prioritized multiple change on belief bases, Journal of Philosophical Logic 41 (2012) 77–113.

[24] P. Liberatore, M. Schaerf, Belief revision and update: Complexity of model checking, Journal of Computer and System Sciences 62 (2001) 43–72. URL: https://www.sciencedirect.com/science/article/pii/S0022000000916982. doi:https://doi.org/10.1006/jcss.2000.1698.

[25] G. Brewka, Preferred subtheories: An extended logical framework for default reasoning., 1989, pp. 1043–1048.