Real-time Wildfire Detection with Semantic Explanations

Author

Phan, Thanh Cong, Quach, Nguyen Duc Khang, Nguyen, Thanh Tam, Nguyen, Thanh Toan, Jo, Jun, Nguyen, Quoc Viet Hung

Published

2022

Journal Title

Expert Systems with Applications

Version

Submitted Manuscript (SM)

DOI

10.1016/j.eswa.2022.117007

Funder(s)

ARC

Grant identifier(s)

DE200101465

Real-time Wildfire Detection with Semantic Explanations

Thanh Cong Phan[a], Nguyen Duc Khang Quach[a], Thanh Tam Nguyen[b,*], Thanh Toan Nguyen[a], Jun Jo[a], Quoc Viet Hung Nguyen[a]

thanhcong.phan@griffith.edu.au, k.quach@griffith.edu.au, nt.tam88@hutech.edu.vn, thanhtoan.nguyen@griffithuni.edu.au, j.jo@griffith.edu.au, quocviethung.nguyen@griffith.edu.au

[a] *Griffith University, Australia*
[b] *Faculty of Information Technology, HUTECH University, Ho Chi Minh City, Vietnam*

Abstract

Wildfire detection is an indispensable component of many resilient platforms, preventing environmental disasters from damaging life. Online detection, which refers to situations in which wildfire events need to be detected in real time, is an essential tool, as the consequences of wildfires might already be irreversible by the time they are detected. Recently, wildfire detection methods based on machine learning and deep learning have been proposed. However, most of these methods take raw data as the input and then aggregate it into predictive features, which requires all of the historical data or a large part of a stream and violates the timeliness requirement of online detection. Moreover, these methods scarcely discuss the explainability (e.g., how the model comes up with the final detection). The lack of explainability reduces human trust in the system and may hinder further applications of the system, especially in high-stakes situations where decisions can have significant consequences. Unlike existing works that treat the timeliness and explainability problems separately, we propose a real-time wildfire detection system that is built upon the streaming capability of complex event processing and the expressiveness of semantic annotation. The system continuously processes raw data streams, transforms them into semantic events, and learns CEP queries that are both predictive and interpretable for humans at the same time. Experiments on four real datasets and one synthetic dataset show that our approach outperforms the baselines in terms of effciency, accuracy, explainability, and adaptivity.

Keywords: wildfire detection, real-time monitoring, explanation discovery, complex event processing

1. Introduction

Remote wildfire detection is the task of automatically identifying natural hazards and potential wildfires using the imagery data of around-the-clock satellites (Yuan et al., 2020). Unlike traditional remote sensing based on field sensors and social media reports, the satellite-aided detection of wildfires provides a timely and non-invasive instrument for the early mitigation of environ-mental changes and damage to life. Due to its promising prospects, remote wildfire detection has received considerable attention (Sousa et al., 2020). For brevitys sake, the terms 'remote wildfire detection' and 'wildfire detection' are used interchangeably. Various types of methods have been proposed for wildfire detection, including spectral indices, machine learning (ML) methods, and deep learning (DL) methods (Sousa et al., 2020).

Although these methods have made comparative progress in wildfire detection, two essential challenges remain: (1) *Explainability* – DL-based models and some ML-based models are not explainable by nature, which is a problem in high-stakes situations where decisions can have significant consequences. They are

*Corresponding author

Email addresses: thanhcong.phan@griffith.edu.au (Thanh Cong Phan), k.quach@griffith.edu.au (Nguyen Duc Khang Quach), nt.tam88@hutech.edu.vn (Thanh Tam Nguyen), thanhtoan.nguyen@griffithuni.edu.au (Thanh Toan Nguyen), j.jo@griffith.edu.au (Jun Jo), quocviethung.nguyen@griffith.edu.au (Quoc Viet Hung Nguyen)

often unable to provide explanations about why these situations occur and about how to avoid such damage in the future. The problem becomes even more critical when the algorithmic output turns out to be a false alarm or missed detection. In the worst case (Sousa et al., 2020), human operators have to validate every alarm, leading to wasted efforts and reduced trust. (2) *Timeliness* – Wildfires happen in real time, while satellite data comes in streams; this situation requires online detection. However, most of the ML- and DL-based methods aggregate predictive features from raw data; thus, the system has to wait for the observations from all of the historical data or a large part of a data stream. This is time-consuming and violates the timeliness restrictions of online detection. Wildfires are often detected after they have already happened and the resulting damage has spread beyond control. As a result, there is a need for a better data stream processing mechanism for real-time wildfire detection.

Recently, two promising paradigms that could overcome the challenges of explainability and timeliness have emerged. First, complex event processing (CEP), an event detection paradigm that does not require machine learning, has recently received considerable attention in a wide range of fields (Boubeta-Puig et al., 2021). As the basic idea of CEP is to detect complex events based on predicates from the data stream, which is an antidote to the timeliness problem in wildfire detection, CEP holds vast potential for detecting wildfire events early. However, in contrast to other event-based tasks, wildfire detection is distinct, because wildfires are dynamic events that cannot be captured by a fixed set of predicates. A wildfires characteristics will naturally shift as time goes on. Second, semantic annotation, the transformation of raw data into human-understandable information, has achieved great progress (Duong et al., 2015). In this paradigm, different views of the same data stream are constructed from various sources of information, such as spatial information, domain knowledge, and environmental information. When different views are incorporated, not only is the performance of detection models enhanced, but also it is easier to explain the output with semantic contexts for human understanding.

Non-trivial obstacles need to be tackled before we can benefit from these paradigms. First, while combining multiple sources of data streams can help to increase human understanding comprehensively, the size of the data increases significantly and places a burden on the processing time. Second, data from different modalities have diverse representations, distributions, scales, and densities. For example, a satellite image is represented by pixel intensities that are real-valued and dense, whereas the temperature is represented by continuous signals over time. It is difficult to provide a common semantic annotation between different data modalities. Third, there are different combinations of predicates that can be used to construct an explanation, and there is a trade-off between the conciseness, consistency, and predictability of these explanations.

In this paper, we design and develop a framework for **F**ast and **A**ccurate wildfire **D**etection with **E**xplanations, or **FADE**, whose goal is to monitor and detect wildfires in real time as well as to provide semantic explanations to facilitate human understanding. The system first transforms raw data from multiple sources into semantic events. Such a transformation not only unifies heterogeneous data into a common semantic model but also significantly reduces the search space, which would be large due to the combinations of raw data streams. The system maintains a list of wildfire-predictive queries that are automatically learned from historical data and user validation. The timeliness of wildfire detection is guaranteed by the complex event processing (CEP) technology, which allows the system to monitor data streams and detect wildfire instances in real time using the learned predictive queries. Finally, an optimal explanation for a detected wildfire is constructed in terms of conciseness, consistency, and prediction power. These explanations bridge the gap between detection and explanation by making the query patterns explainable and predictive at the same time.

To the best of our knowledge, our work is a first attempt to realise real-time wildfire detection with on-the-fly explanation discovery. We focus on the earliness of detection and the real-time processing of data streams, which state-of-the-art wildfire detection systems lack of. We also show that by carefully-designed semantic annotation pipeline and adaptive CEP patterns, we can achieve comparable performance to deep learning based classifiers. In particular, we directly leverages the predicates in the CEP engine to generate explanations for machine learning tasks. Such a new form of explainability agrees with the human reasoning process, which often relies on a set of predictive rules. Moreover, unlike traditional monitoring systems, which rely only on satellite streams, we enrich these data streams with semantic data from spatiotemporal

properties, environmental measurements, and domain knowledge such as atmosphere, weather, and land-use data. More specifically, our contributions are summarised as follows:

- *Explainable streaming wildfire detection:* We design and develop an event processing framework with the ability to discover useful explanations for wildfire detection.
- *Semantic annotation:* We design an annotation module that automatically transforms raw data streams into semantic data streams with a rich feature space. Such semantic annotation enables human-understandable explanations and reduces the search space (Zhang et al., 2017).
- *Explainable CEP engine:* We extend the CEP model to support geo-spatial and semantic information, facilitating the construction of explainable predictive queries.
- *Explanation discovery:* We provide a formal definition of constructing an optimal explanation for wildfire detection as a subset maximisation problem with a class-discriminative reward (Zhang et al., 2017). The explanation is constructed optimally in such a way that the resulting set of predicates is concise, consistent, and predictive.
- *Event-annotated datasets:* We provide the datasets of satellite data streams with rich semantic annotations.
- *Comprehensive evaluations:* We create a sizable collection of empirical benchmarks to evaluate the efficiency, the effectiveness, and the adaptivity of our system.

The remaining sections of the paper are organised as follows. §2 provides background information and related work. §3 describes the model and the problem statement. §4 discusses the principle and provides an overview of our approach. Then, §5, §6, and §7 describes the details of the semantic annotation, the real-time detection, and the explanation discovery components respectively. §8 reports the experimental results. §9 discusses the findings, limitations, and future works before §10 concludes the paper.

2. Background and Related Work

In this section, we review the literature on concepts related to our framework, including wildfire detection, explainable machine learning, and complex event processing.

2.1. Wildfire detection

A plethora of wildfire-sensing satellites have been put into space. The first generation includes mostly polar-orbiting systems such as MODIS (Giglio et al., 2016), AVHRR (Flasse & Ceccato, 1996), and VIIRS (Schroeder et al., 2014). However, these satellites, despite having high spatial resolutions, fly around the poles and thus are limited to daily imaging only (no spot on the Earth's surface can be monitored continuously during a day) (Xu et al., 2010). The second generation consists of geostationary systems that are often used, even though they have smaller resolutions, due to their frequent monitoring capability. Table 1 presents the specifications and detection characteristics of typical satellite systems in the field.

Table 1: Specifications and detection characteristics of wildfire sensors

Satellite System	**Spatial Resolution**[1]	**Input Rate**	**Fire Size**	**Fire Radiative Power**	**False Alarms**
MODIS (1999)[2]	1km	6 hours	$\sim 900m^2$	$\sim$ 15MW	5%–85%
AVHRR (1998)[3]	1.1km	6 hours	$\geq 1km^2$	$\sim$ 1MW	2%–62%
VIIRS (2011)[3]	375m	12 hours	$\geq 5m^2$	$\geq$ 1MW	6%–40%
AHI (2014)[4]	500m	10 minutes	$6km^2$	$\geq$ 40MW	10%–60%
ABI (2016)[4]	1km	15 minutes	9.2 km^2	$\geq 30MW$	3.3%–53.4%

[1] Geographic size of each pixel; [2] Low Earth orbiting; [3] Polar orbiting; [4] Geostationary;

Over the decades, hyperspectral sensors (e.g., MODIS) have been gradually replaced by multispectral sensors (e.g., AHI, ABI) due to their level of complexity (200 continuous channels compared to 16 representative channels per image) and channel redundancy (the differences between consecutive channels are too narrow) (USDA, 2019). In this work, we select the GOES-16 platform as a data source since it operates on the US continent, where the wildfire datasets are available (see §8). GOES-16 is a state-of-the-art geostationary system. The great advantage of GOES-16 is its high spatial resolution and high temporal resolution (i.e., input rate), which are suitable for fast-spreading events like wildfires. GOES-16 also has the advantage of being geostationary, i.e., it can monitor incidents without significant geometric correction of the images, leading to robust spatial analyses (Xu et al., 2010).

Various types of methods have been proposed for wildfire detection. In the early stage, wildfire detection largely depended on the spectral indices of satellite images. For example, the values of the spectral channel 7 from GOES-16 satellites are often used as an indicator of wildfire hotspots (Sousa et al., 2020). However, some studies report that spectral indices are prone to errors due to different atmospheric conditions and locations. Significant research efforts have been given to using traditional ML techniques (e.g. decision tree, SVM), which classify wildfires by hand-crafted features. These features are obtained by analysing the data streams transferred from the satellites to Earth servers. However, hand-crafted features are not robust (Yuan et al., 2020). On the other hand, the advances of DL in many domains, such as computer vision (Ismael & Şengür, 2021), have inspired the decision makers to apply DL to wildfire detection (Sousa et al., 2020; Rashkovetsky et al., 2021). In that, the whole ML feature extraction workflow is replaced by a neural network, such as CNN and RNN (Sousa et al., 2020; Rashkovetsky et al., 2021), to automatically extract latent features from the raw input and generalise them with little human input.

In this work, we focus on the earliness of detection and the real-time processing of data streams, which state-of-the-art wildfire detection systems lack of. However, we show that by carefully-designed semantic annotation pipeline and adaptive complex event processing patterns, we can achieve comparable performance to deep learning based classifiers.

2.2. Explainable systems

Explanations of detection outputs help decision-makers to investigate the root cause of a wildfire instance. Such an understanding can prevent future damage by mitigating the underlying environmental factors or changing the underlying detection mechanism in order to increase true positives and reduce false alarms. Various explanation methods have been developed that consider the original model as a black box and generate reverse-engineered information, such as saliency maps and perturbations (Ma et al., 2021; Angelov et al., 2021). Some works also try to develop intrinsically explainable models by injecting attention layers or prototypical layers into deep neural networks (Ren et al., 2019). However, most of these works are applicable for static settings, which require the storage and use of all of the historical data.

Most wildfire detection models inherently lack human-understandable explanations for their output. On the one hand, complex models like classifiers (Phan et al., 2020) rely on a set of raw features, which are too low-level and noisy to facilitate human understanding. Even with the advance of deep learning methods in which the latent features are learned automatically, these features still lack the ability to generate rich semantic explanations, especially with streaming data. On the other hand, lightweight models (Kim & Yang, 2019) rely on threshold-based predicates over spectral indexes, which are somewhat understandable for domain experts. However, they often yield a lower prediction power in practice. Moreover, these predicates often become obsolete over time in the presence of unknown phenomenon or concept drift. For example, the detection threshold of a spectral index is not the same for every region and could even be different for the same region in the presence of different weather conditions and flammable land materials.

Unlike existing systems, our work defines a new form of explainability in wildfire detection that agrees with the way that humans describe their own thinking via a set of rules. We directly use CEP patterns or predicates as explanations for the detection output. Moreover, explanations are constructed optimally so that they are concise, consistent, and predictive. In this work, we even use these explanations as the medium for users to give feedback on the correctness of the detection predicates and update them accordingly.

2.3. CEP systems

Complex event processing (CEP) is a computational paradigm that supports capturing and monitoring a stream of data to discover the situation of interest in real time. The situation of interest is user-defined in the form of rules or patterns via the SASE query language or other equivalent CEP query languages (Zhao et al., 2021). Below, we present some definitions that are used in our approach.

Event. We define an event as an activity of interest that happens, or could happen, in a system. Examples of events could be an increase in temperature in a particular geographical area.

Event attribute. An event attribute is an element in the event structure and represents a property or characteristic of an event. Each attribute includes a name and a specific type (Integer, Float, Boolean, or String).

Event type. An event type is a collection of event objects belonging to the same class. Each event type is defined by a schema as a set of pre-defined attributes, including timestamp.

Primitive event. A primitive event is an event that cannot be decomposed into other events. A primitive event can also be referred to as an atomic event.

Geographical event. We define a geographical event as a primitive event associated with spatial information about where the event is reported to occur. The spatial information can be a location or a region. Traditional systems follow a post-processing mechanism, where the data stream is mapped back to the geospatial domain after detection. Our approach has the advantages of reducing the search space and enabling more flexible predictive patterns.

Complex event. A complex event is a combination of primitive or geographical events that describe the situation of interest. A complex event is also called a pattern and the system will actively report whenever the pattern is detected.

A few recent studies have dedicated for the automatic generation of CEP rules (Mousheimish et al., 2017). However, the setting is often simplified by assuming that a phenomenon can be captured by one or a few rules, which is also domain-dependent. To the best of our knowledge, our work is a pioneering attempt to learn CEP rules for predictive purposes like wildfire detection.

3. Explainable CEP Approach

In this section, we discuss how we model the data streams for wildfire detection and how to formulate the problem of doing wildfire detection with semantic explanations.

3.1. Model and definitions

Heterogeneous data streams. The raw data stream of a remote sensing measurement, generated by a data source such as a satellite, a weather station, or land use, is recorded as a sequence of spatio-temporal values consisting of $(value, longitude, latitude, timestamp)$ quadruples. Formally, it is a sequence of spatio-temporal values recording the trace of a remote sensing measurement, i.e., $D = \{q_1, \ldots, q_m\}$, where $q_i = (v, x, y, t)$ is a quadruple of a measurement value with the position (longitude x, latitude y) at timestamp t. The set of all raw data streams is denoted as $\mathcal{D} = \{D_1, \ldots, D_k\}$. Raw data streams have different characteristics. For example, the sampling rate of a data stream is often fixed, but different data sources can have different sampling intervals (Phan et al., 2020) (e.g., 15 min for satellite streams and 5-60 min for weather streams). We call all of these different data streams *heterogeneous data streams*, and they are the input for our system.

Note that a traditional wildfire detection framework using multi-modal data (Phan et al., 2020) would require an additional alignment step to align the timestamps of different data streams (e.g., via interpolation) so that each timestamp would have a full set of measurement values for wildfire detection. However, such an alignment step hinders the timeliness of detection since a window of a sufficient amount of data needs to be collected for an accurate alignment (Phan et al., 2020) (e.g., an interpolation often uses many neighbouring values to impute a missing value). Going beyond the state of the art, this alignment step is not needed in our framework thanks to our approach of complex event processing, which we will describe later.

Semantic events. As a first attempt towards semantic satellite streams, we introduce *semantic events* (e) as the semantic counterpart of the satellite data events. They denote remote sensing measurements defined in or inferred from third party data sources that contain information understandable for humans. Table 2 shows the semantic types that are taken into account in this study; our approach is also applicable to other event types.

Table 2: Examples of event types in explainable wildfire detection.

Event type	Meaning	Schema	Semantic features
Location	Spatial information	[timestamp, eventId, boundary]	region, land-use
Imagery	Satellite signal	[timestamp, eventId, channelId, boundary, value]	GOES-16 spectral channels
Weather	Weather signal	[timestamp, eventId, weatherId, boundary, value]	temperature, humidity, air pressure
Fuel type	Fuel type signal	[timestamp, eventId, boundary, value]	forest, grassland
Aerosol	Smoke signal	[timestamp, eventId, boundary, smoke category]	spectral channels 7 and 14
CO2	Emission signal	[timestamp, eventId, boundary, C16 value]	spectral channel 16
Hotspot	Radiation signal	[timestamp, eventId, boundary, radiation value]	spectral channels 7 and 14

Semantic data streams. We annotate for each data stream D into a sequence of n semantic events, $E(D) = \{e_1, \ldots, e_n\}$. Each data stream (representing a data source) is bootstrapped with a fixed number of event types. The details of such annotation will be described in §5.

For brevity, we refer to a unified multi-modal semantic stream, i.e., a set of different semantic data streams, as $\mathcal{E} = \{E(D_1), \ldots, E(D_k)\}$, where k is the number of data sources. A wildfire usually happens during a segment of $\mathcal{E}$, whose length will be captured by the detection component or by user validation if available. In other words, $\mathcal{E}$ will be segmented into a sequence $\mathbb{E} = (\mathcal{E}_1, \mathcal{E}_1, \ldots)$, where each segment $\mathcal{E}_i$ is labelled with a class $c \in \{1, 0\}$ to reflect whether a wildfire happens (1) or does not happen (0) during that segment.

3.2. Problem statement

As formulated above, remote sensing data is often multivariate and represented by a multi-modal data stream. In other words, each observation is a combination of events (referred to as features) from multiple time series (Mousheimish et al., 2017). Our system takes these multivariate time series as input and jointly solves two problems. The first problem is to learn predictive patterns for classifying a wildfire event seamlessly. These patterns will capture various correlations (synchronous, sequential, and time gaps) that might exist between multivariate time series. The second problem is to automatically transform these predictive patterns into a real-time wildfire detection engine with no human intervention.

4. The Overall Architecture

In this section, we discuss how we come up with a framework that combines real-time wildfire detection and explanation discovery. We begin with the design principles of our framework and why we need them. Then we clarify the components of our framework (Fig. 1), including semantic annotator, real-time wildfire detection, and explanation discovery.

4.1. User-centric design principles

We argue that a good solution to wildfire detection with explanations should respect the following criteria:

(C1) *Pro-active monitoring:* It is almost impossible for experts to define a predictive pattern with complicated data inputs regardless their level of expertise, especially with nature phenomena like wildfires. These situations call for a class of proactive monitoring systems that supports pattern learning based on historical data and validated detection output. In other words, the system must be proactive in detecting wildfires and learning the underlying explanations of these detections.

(C2) *Explainability:* A detection result with semantic explanations would be meaningful for human understanding and prevent future instances. However, providing an explanation that is both human-readable and reflective of the root cause is challenging. This requires abstracting raw features into semantic annotations that have a high predictive power.

(C3) *Adaptability:* In practice, the patterns of historical data are no guarantee of future detection accuracy. This is because the distribution of streaming data may show concept drift, i.e., it may be non-stationary (Babüroğlu et al., 2021). As such, old patterns need to be updated and new patterns need to be discovered. The challenge is to update the detection engine without retraining it on the entire dataset to guarantee the timeliness of detection.

4.2. Framework components

Regarding the above criteria, we propose a framework for real-time wildfire detection that combines data stream processing and explainable machine learning paradigms. The framework consists of three components, as depicted in Fig. 1.

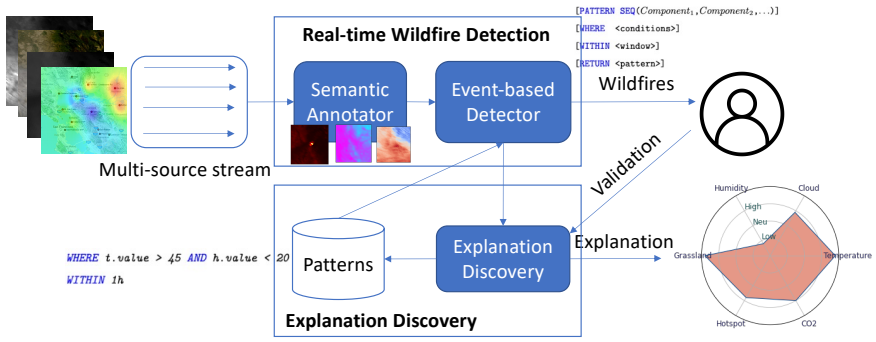

Figure 1: Real-time wildfire detection with semantic explanations. Three important components: 1) *Semantic Annotator* – enriches the multi-source data streams with semantic information, 2) *Event-based Detector* – detects wildfires in real-time based on CEP, 3) *Explanation Discovery* – updates the CEP patterns based on user validation and produces explanations for users. The patterns are CEP conditions themselves and can be integrated directly into the CEP engine for new detections.

Semantic annotator. Discovering complex patterns that are explainable is notoriously hard. Moreover, including all available data may lack semantics and neglect the scalability of the system. Following these ideas, this component extracts and annotates the multi-modal data stream with semantic, high-level events. We further reduce the search space by merging adjacent locations with the same semantics to form larger events associated with a boundary of the region (i.e., a geospatial event). This component is described in more detail in §5.

Real-time wildfire detection. We leverage the stream processing nature of the CEP technology to detect wildfires in real time. Since wildfires originate from a specific location and often spread throughout a region, we propose a geospatial extension for CEP that allows us to monitor and process environmental factors combined with geospatial information. The CEP engine will find wildfire events by matching the data streams against a list of predictive queries, referred to as patterns, which are learned from user validation and historical data. As they are built upon semantic events, these patterns are also used as explanations for the user in the explanation discovery component. This component will be described in detail in §6.

Explanation discovery. This component constructs the best explanation for a wildfire instance by solving the subset optimisation problem to find the set of semantic events (features) that are concise, consistent, and have high predictive power. This requires the definition of a reward function for a feature set and the transformation of these features into query predicates in order to construct the corresponding CEP queries. This component will be described in detail in §7.

5. Semantic Annotation Algorithms

This section explains the details of semantic annotation, considering our objective – algorithms should exhibit high explainability for wildfire detection over a wide range of streams with varying modalities. The core features of these annotations are summarised in the "semantic features" column of Table 2.

5.1. Motivating example

Fig. 2 shows a running example of our semantic annotation. Assume that we are interested in monitoring the continuously flowing data from heterogeneous and distributed sources to detect natural hazards (i.e., wildfires). To this end, raw data are collected in streams with extraordinarily high and unpredictable rates. A combination of these data signals may refer to the situation in which there is a hazardous phenomenon with some probability. For example, a wildfire is more likely to occur in an area where the temperature is high and the humidity drops significantly in comparison with a normal situation.

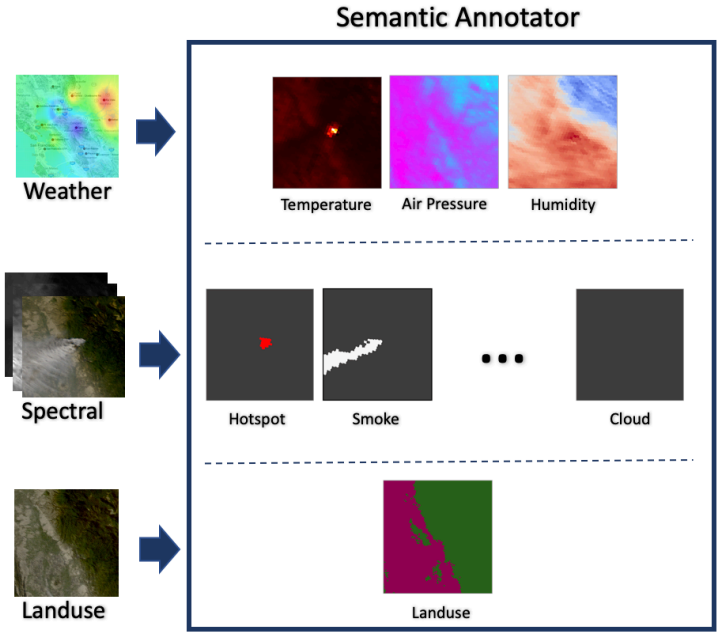

Figure 2: An illustrative example of semantic event streams

In this paper, we consider wildfire detection as a matching of pre-defined patterns against the heterogeneous data streams. However, matching the raw data streams directly would have low explainability, as the patterns are constructed from raw features. A good explanation should be given in the form of human-readable and semantic information. For example, a wildfire may be caused by a sequence of events '*Within 2h:*$(temperature == HIGH \wedge Cloud == HIGH \wedge Humidity == LOW \wedge fuel == grassland)$' and fires rarely occur in a region where $(fuel == water)$. With such a good explanation, users would know how a wildfire has arisen in order to react to, mitigate, or prevent it in the future.

In the following, we provide a set of semantic annotations that are suitable for human understanding as well as real-time wildfire detection.

5.2. Annotation with semantic regions

This procedure enables the annotation of data streams with meaningful geographic regions. It does this by computing topological correlations of the location information of data streams with third-party data sources containing the semantic places of spatial regions.

The topological correlation is measured using a spatial join between the sequence of locations derived from the raw data (i.e., via longitude x and latitude y) and semantic regions. Several forms of spatial predicates are used to compute the correlation, depending on the type of data. These can be a combination of directional, distance, and topological spatial relations (e.g., intersections) (Yan et al., 2011). After finding the appropriate regions, the procedure creates, for a given data stream, a derived data stream with these regions and associated metadata. The derived data stream is denoted as $E_r(D) = \{r_1, \ldots, r_m\}$, where $r_i = (x_1, y_1, x_2, y_2, t)$ is a tuple of the region position (bounding rectangle) at timestamp t. Note that, depending on the requirements, the spatial join can be computed only for high-potential locations (e.g., with a history of frequent wildfires).

The semantic regions also contain land-use information that could provide a causal explanation for a detected wildfire. For example, grassland and forests are more likely to have wildfires than wetlands or riverside areas.

5.3. Annotation with spectral information

Satellite data is acquired as imagery data streams available on Amazon Web Services (AWS) (Blaylock, 2019). The data samples are generated by the ABI of GOES-16 satellite, which captures Earth's radiance in 16 spectral channels (Table 3) via a variety of radiance detectors. Basically, they are digital maps of outgoing radiance values at the top of Earth's atmosphere at visible, infrared, and near-infrared wavelengths. Then, the samples are compressed, packetized, and sent to the ground station, in which they are converted to geo-located and calibrated pixels (Blaylock, 2019), covering the whole America continent. The raw image pixels are kept in Network Common Data Form (netCDF) format, which is descriptive, flexible, standardized among large research projects (UCAR, 2019). Each channel of an image sample is kept in a separate netCDF file for each 15-minute interval.

Table 3: Spectral channels of a GOES-16 image.

Channel	**Central (μm)**	**Detection Utility**	**Type**
1	0.47	Blue	Visible
2	0.64	Red	Visible
3	0.865	Veggie	Visible
4	1.378	Cirrus	Near-IR
5	1.61	Snow/Ice	Near-IR
6	2.25	Cloud Particle Size	Near-IR
7	3.90	Shortwave Window	Infrared
8	6.9	Upper-level Tropospheric Water Vapor	Infrared
9	6.95	Mid-level Tropospheric Water Vapor	Infrared
10	7.34	Lower-level Tropospheric Water Vapor	Infrared
11	8.5	Cloud-Top Phase	Infrared
12	9.61	Ozone	Infrared
13	10.35	Clean Infrared Longwave Window	Infrared
14	11.2	Infrared Longwave Window	Infrared
15	12.3	Dirty Infrared Longwave Window	Infrared
16	13.3	CO2 Longwave Infrared	Infrared

Besides the detection utility and the type of the channel (reflection or radiance), we annotate the data stream of each spectral channel with its value distribution. An example of the radiance value distribution of these channels is shown in Fig. 3. Formally, for a given data stream D, the annotation produces a sequence of spectral events $E_c(D) = \{c_1, \ldots, c_n\}$, where $c_i = \{v_i, \mu_i, \sigma_i, u_i, l_i, t\}$ contains the empirical mean μ, the standard deviation σ, the maximum u, and the minimum l up until the time step i of the data stream. These statistics can be updated incrementally as a new data point v_i arrives as follows:

$$\mu_i = \frac{\mu i - 1 \times (i-1) + v_i}{i}; \quad \mu'_i = \frac{\mu_{i-1} \times (i-1) + v_i^2}{i};$$
$$\sigma_i = \sqrt{\mu'_i - \mu_i^2};$$
$$u_i = \max(l_i, v_i);$$
$$l_i = \min(l_i, v_i).$$

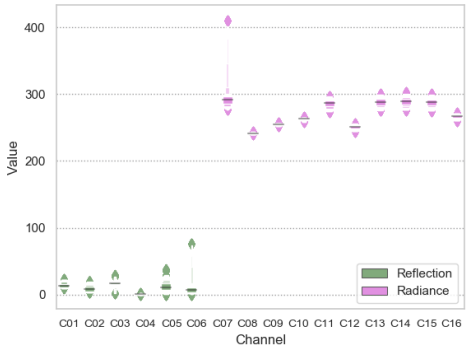

Figure 3: Value distribution of fire pixels

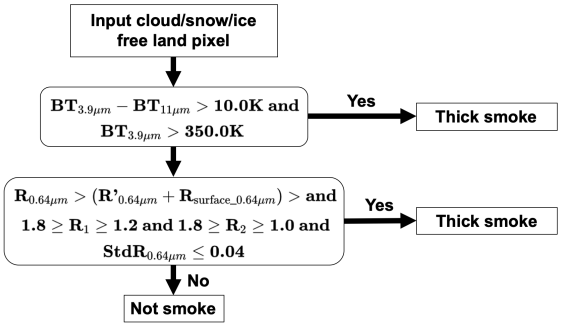

Figure 4: Aerosol (smoke) annotation.

These equations are derived from the definitions; in particular, $\mu = \mathbb{E}[X]$ and $\sigma = \sqrt{\mathbb{E}[X^2] - \mathbb{E}^2[X]}$. These statistics will help to explain for users whether a measurement is low, medium, high, or anomalous.

5.4. Annotation with domain knowledge

Aerosol event. Aerosol amounts must be known for the accurate atmospheric correction of remotely sensed images and are required to accurately gauge the available solar resources (De Laat et al., 2012). In GOES 16, the Advanced Baseline Imager L2+ Aerosol Detection (ABI L2+ ADP) product algorithms are used to detect the presence of aerosol (including smoke) in the atmosphere over land and over ocean, with associated quality flags to indicate the confidence level (low, medium, and high) for the detected smoke (NOAA/NESDIS/STAR, 2018). This aerosol information can be used to explain wildfire scenarios on land, e.g., channel 7 and channel 14 play important roles in smoke detection in the ABI L2+ ADP.

Fig. 4 shows the rule for annotating an aerosol (smoke) event. This is based on the fact that smoke/dust exhibits features of spectral dependence and contrast over both the visible and infrared spectrums that are different from clouds, the surface, and a clear-sky atmosphere. The fundamental principle of the annotation depends on threshold tests, which separate smoke/dust from clouds and clear sky over water and land. More precisely, fire spots are detected by looking at pixels with brightness temperature (BT) at 3.9 µm (channel 7) greater than 350K and a brightness temperature difference (BTD) between 3.9 µm and 11 µm (channel 14) greater than or equal to 10K. Pixels that pass the fire test are assumed to have smoke. The smoke tests over land take advantage of a linear relationship observed between the MODIS reflectance at 0.64 µm (channel 2 in GOES) and 2.13 (channel 6 in GOES), based on the fact that smoke is transparent at 2.13 µm. This relationship gets noisy when the reflectance at 2.13 µm is greater than 20%. When smoke is present in a pixel, there is a larger increase in R0.64 µm than in R2.13 µm.

Hotspot. In GOES-16, the ABI fire algorithm is a dynamic, multi-spectral, thresholding contextual algorithm that mainly uses the 3.9 µm and 11.2 µm bands (ABI channels 7 and 14) to locate hotspots and retrieve their characteristics. We leverage this algorithm for hotspot annotation. During the day, the measurement range of 275-400K and accuracy values of 2.0K represent the ABI channel 7 (3.9 µm) input data that is needed to characterise a hotspot (NOAA/NESDIS/STAR, 2013). At night, hot fires emit radiation with a wavelength of 2.24 µm. This band can detect that emitted energy in the absence of clouds, and it is more sensitive at night when there is no sunlight. In summary, a hotspot event is annotated when the value of ABI channel 7 is high or the difference between ABI channels 7 and 14 is high (Hally et al., 2019). Fig. 5 illustrates an example of channel 7's values over time; the peak period is when a wildfire happens.

CO2. Fires from biomass burning are responsible for emitting large quantities of CO2, CO, and PM2.5 into Earth's atmosphere (Lee, 2007). In the GOES-16 satellite, channel 16 (i.e., the CO2 channel) has a particular wavelength of 13.3 μM which can absorb CO2. For this reason, we annotate the CO2 event according to channel 16.

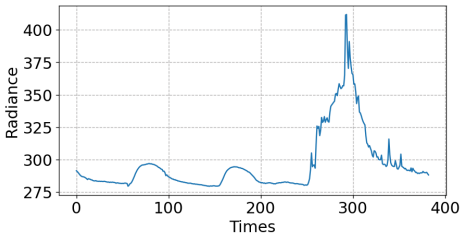

Figure 5: An example of GOES-16's channel 7 overtime.

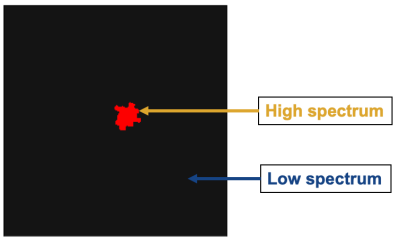

Figure 6: Geospatial event with boundary.

5.5. Annotation with environmental information

We aim to design a flexible framework that is able to easily incorporate domain knowledge (C3) when needed. In this section, we demonstrate how knowledge from other domains is integrated into the system. While we use *weather* and *fuel content* measurements in this study to demonstrate the system's adaptivity, our framework is applicable to other information domains.

Weather. Since wildfires are more likely to happen on hot and dry days, weather information provides an additional degree of confidence for remote detection using satellite images. Moreover, nearby weather stations, if they are close enough, might provide additional evidence of fire, such as increasing temperature and barometric pressure and decreasing humidity, recorded by weather meters within the burning zones (Doolin & Sitar, 2005). In summary, we annotate the weather event as hot if the temperature is high and as dry if the humidity is low.

Fuel moisture content. Existing works have studied the correlation between the fuel content and the occurrence of fires (Ottmar & Alvarado, 2004; Chuvieco et al., 2009). When adding fuel type events into the system, they can be considered as a mask applied over the planet surface to efficiently filter out false alarms, as wildfires cannot occur in many land categories, such as buildings or water surfaces. In this study, we focus on fuel categories that have a high chance of wildfire occurrence, including forests and grasslands. To annotate the fuel type of a region of interest, we apply the Landsat 8 classifier (Jia et al., 2014).

6. Real-time Wildfire Detection

In this section, we describe the system setup for our real-time wildfire detection. Traditional studies on wildfire detection focus on classification settings, i.e., systems are oriented to classify events that have already happened. However, more often than not, wildfires have already had a severe impact before being detected. Even though online classification techniques exist (Cafaro et al., 2021), the detection results are often lagged behind the event. For example, classifiers often need a pre-processed data segment to classify an event as a wildfire (Toan et al., 2019; Sousa et al., 2020). This assumes that the segments are cut perfectly. However in practice, if cutting is done too late, the wildfire may have already caused irreversible damage. Otherwise, there might not be enough information for detection.

In this work, we approach the problem with event-driven processing technologies, in particular complex event processing (CEP), which helps us to proactively react to wildfire events and situations seamlessly. However, even modern proactive CEP systems still require decision makers to have thorough knowledge of the CEP language. Moreover, it is challenging and tedious to define correct patterns that can predict the situations of interest. Moreover, future situations can have new characteristics that previously specified patterns cannot capture.

To overcome these challenges, we take a new approach for defining CEP queries: we automatically learn them from history and new validated scenarios. Our system continuously processes data from multiple sources and detects the situation of interest by looking at the semantic events that have been observed. We also go beyond tradition CEP systems by integrating geospatial events and geospatial patterns into the processing pipeline.

6.1. CEP-based wildfire monitoring system

The input of the wildfire monitoring system is an infinite sequence of events or an event stream. Besides the underlying information, each event is characterised by time annotation and the payload. The former indicates the chronological order of the events, while the latter reveals additional information about their occurrence. Formally, the system processes k event streams of different types $E_1, \ldots, E_k$. Each event stream is defined as: $E = (e_1, e_2, \ldots)$, where e_i has the same event type according to the semantic annotation algorithm that is used (see Table 2).

We consider a CEP engine that monitors these events using wildfire-predictive queries. For common usability, monitoring queries are demonstrated in the popular SASE query language (Zhao et al., 2021).

SASE reminder. The SASE language is a SQL-like structure that supports the *filtering, correlation, and transformation* of events. The SASE language is a declarative language with semantics associated with each clause. The `PATTERN` clause specifies the sequential pattern in the input stream. The `WHERE` clause gives value constraints over the input events to satisfy the pattern. A sliding window for time constraints over the event pattern is further specified in the `WITHIN` clause. Finally, the `RETURN` clause yields a complex event for the final result of the query. The general syntax of CEP queries in SASE is as follows:

```
[PATTERN SEQ(Component_1,Component_2,...)]
[WHERE  <conditions>]
[WITHIN <window>]
[RETURN <pattern>]
```

Example 1. *An example query, Q_1, for wildfire monitoring is shown below.*

```
PATTERN SEQ(Temperature t, Humidity h)
WHERE t.value > 45 AND h.value < 20 AND t.area == h.area
WITHIN 1h
RETURN (t.area)
```

This query detects all pairs (t,h) of temperature and humidity events above 45 °C *and below 20%, respectively, in the same area (t.area = h.area), in which h and t are produced in a 1-hour time frame.*

Processing model. The query plan is comprised of the following basic operators:

- *Sequence scan and construction:* A query specifies a sequential pattern that is predictive of wildfires using the `SEQ` operator, which requires components in the sequence to occur in the specified order. The purpose of this operator is to transform a stream of events into a stream of sequences, in which each sequence is a unique match of the pattern of interest. For example, Q_1 specifies two components: the first component is a single event of the type *Temperature*; the second component is the Kleene closure of a set of events of the type *Humidity*.

- *Selection and predicates:* Similar to relational database queries, the selection operator filters each sequence by applying all the predicates specified by the `WHERE` clause. If all evaluations are successful, the sequence is transferred to the output result.

- *Window:* The window operator examines the constraint of the `WITHIN` clause. It considers each sequence and checks whether the temporal correlation between the relevant events is less than a certain time window, which is specified in advance.

- *Return matches:* A query specifies the matches to return in the `RETURN` clause. Matches comprise a series of events, including a timestamp, with raw or derived attributes. We denote a match by m and M_Q is the set of all matches. Each match corresponds to a wildfire instance.

6.2. Semantic extension for CEP

CEP technologies have proved their efficiency in rules monitoring over data streams (Wu et al., 2006). However, they lack the native support for the geographic information of the events, which is vital for environmental and disaster monitoring. To resolve this shortcoming, we propose an extension that takes the standard metamodel of CEP as a reference and endows it with the geographic component. The proposed component enables extended operators that support geo-operators, such as `Union`, `Intersection`, and `Distance`, among others.

The additional geo-operators for geospatial queries are also supported:

- Geo-arithmetic: They are those that receive n-operands of the geometry type and return another geometry. These are, for example, `Union` and `Intersect`.
- Geo-Boolean: They are those operators that receive n-operands and return a `true` or `false` value. These are the classic operators: `Equals`, `Distance`, and `Contains`.

The former group of operators aims at returning the boundary area where the pattern occurs, while the latter group in particular lets you indicate whether an event occurs at a given location or area (or not). The proposed model makes it possible to represent GeoEvents and GeoOperators of any type, independent of a specific CEP platform.

Example 2. *When new semantic information is needed, the predictive pattern can be adapted by using the semantic exten-sion schema. We demonstrate below how the temperature event is incorporated. A new event schema for temperature is created by inheriting from the geospatial base schema. The threshold for transitioning from `LOW` to `HIGH` is specified by domain experts. The boundary for the event is handled by the geospatial extension for CEP.*

```
FROM <streaming events>
PATTERN P2_fire SEQ(Satellite c6, Satellite c7, Satellite c14, Satellite c16,
Temperature temp))
WHERE c6.level==HIGH AND c7.level==HIGH AND c14.level==HIGH AND c16.level==HIGH
AND temp.level==HIGH
WITHIN 2h
RETURN(c6.time, boundary=INTERSECT(c6.boundary, c7.boundary,c8.boundary,c9.boundary,
temp.boundary)
```

Explainable CEP language. To distinguish the wildfire and non-wildfire classes, we generate two types of basic events: the `LOW`-event captures the activity when a feature value is $< \delta_i$ and the `HIGH`-event depicts the feature with a value $\geq \delta_i$. Following this idea, each primitive event complies with the below schema:

```
primitive_event = (timestamp, eventType, eventId,
level = <HIGH,LOW>)
```

Wildfires and other natural disasters occur in an area rather than at a specific point (see Fig. 6). Thus, we merge adjacent points with same semantics to form a larger event with geographic information (i.e., a geospatial event). The schema for such a geospatial event follows:

```
geospatial_event = (timestamp, eventType, eventId,
boundary, level = <HIGH,LOW>)
```

We show in the experiment that our proposed geoEvent helps to significantly improve the effciency, as the search space is reduced significantly. It is worth noting that the event extraction module is employed at run-time to transform raw data into a stream and then into semantic events, as described in §5.

6.3. Bootstrap with predictive models

Wildfire-predictive patterns could be bootstrapped using machine learning models (Toan et al., 2019). Here we focus on interpretable models to increase the explainability of the detection. We consider each wildfire instance associated with a set of attributes and evaluate the explanations produced by two standard prediction techniques.

Logistic regression. This prediction technique produces a weighted combination over a set of features, which can be used to extract a predictive pattern. In our case, the model would assign non-zero weights to 23 input features. However, such weights are too noisy to be of use and are quite overwhelming for human understanding (the human cognitive load is usually limited to understanding around 7–10 features at a time (Shraga et al., 2020)).

Decision tree. This technique builds a tree, where each non-leaf node is a predicate, while the leaf nodes are predictions. This model is often considered concise in terms of explainability. However, the learned predicates are not very consistent with the ground truth (Zhang et al., 2017). Hence, it is only used to bootstrap the predictive patterns in the CEP engine, which will be replaced later by new discovered patterns that have both high predictive power and high explainability.

6.4. CEP pattern learning

In this work, we focus on explainable patterns that have an understandable format based on predicates.

Definition 1 (Pattern)**.** An explainable pattern e is a Boolean expression in conjunctive normal form (CNF). It contains a conjunction of clauses, where each clause is a disjunction of predicates, and each predicate is of the form $\{f, o, c\}$, where f is a feature value, c is a constant, and o is one of five comparators: $o \in \{>, \geq, =, \leq, <\}$.

Example 3. *A good pattern for detecting wildfires in the Kincade Fire dataset is '*$(temperature == HIGH \wedge Cloud == HIGH)$*', which is a conjunction of two predicates. The conjunction of the two predicates means that the two events happened simultaneously within a certain time frame. It means that a wildfire is likely to happen in a region with a high temperature and high cloud cover, and such a pattern is easy for humans to understand.*

Discovering explainable patterns. Traditional explainable AI techniques often consider the explanation and the detection of wildfires as two separate problems. However, doing so often causes an inconsistency between the explanation and the detection results (Zhang et al., 2017), i.e., the explanation does not truly reflect the ground truth. In this work, we go beyond the state of the art by making CEP patterns explainable and predictive at the same time.

Arriving at such explainable patterns requires balancing conflicting goals: simplicity, which pushes patterns to smaller sizes, and informativeness, which pushes patterns to larger sizes to increase the information content. For example, the pattern '$(temperature == HIGH \wedge Cloud == HIGH)$' is worse than '$(temperature == HIGH \wedge Cloud == HIGH \wedge Humidity == LOW \wedge fuel == grassland)$', because, while it is smaller, it does not cover all characteristics that contribute to a wildfire; thus, it is less consistent with the ground-truth.

Given the importance of such goals, we will describe in details the problem of discovering explainable patterns in §7.

Updating the ranking of discovered patterns. At any point in time, the system maintains a list of patterns (Def. 1) that are ranked by a reward function (Eq. 4), which is computed on the set of features in each pattern. This function will be described in detail in §7. The reason for our design of continuous updates of patterns is the dynamic nature of event streams in general and for wildfire situations in particular. That is, the old patterns can become obsolete, and the new patterns might conform to the ground truth over time.

More precisely, our system allows user to annotate wildfire instances, as well as reference intervals that demonstrate non-wildfire situations. The annotations will be sent to the underlying system as the

user validation of wildfire and non-wildfire instances. Based on the validated instances, a pattern will be constructed. If this pattern is new, it will be added to the pattern list of the system and ranked by its reward value. If this pattern exists in the system, its rank will be updated by merging its wildfire and non-wildfire instances and recomputing the reward value. Moreover, it is worth noting that users can choose to limit the number of active patterns to the top k patterns in order to further improve the efficiency of the detection. However, such an extension is beyond the scope of this paper and is considered future work.

7. Explanation Discovery

In this section, we show how to discover an optimal explanation in terms of information theory. Especially, the explanation discovery is based on a rich feature space including the aforementioned semantic annotations.

7.1. Formalising explanations

While in practice, users can observe the evolution of semantic data streams via time series-like visualisations, they do not provide reasons for the detected wildfire. This is because the number of semantic data streams is still overwhelming for users to manually check (there are 23 of them in our case), not counting the large volume of historical information. Our goal is to automate the explanation process within the detection pipeline.

Fig. 7 describes how the explanation is discovered. The knowledge of domain experts can be accumulated as an archive by annotating the events that have happened in the past (§5) and then, the construction of the explanation is described in §7.5.

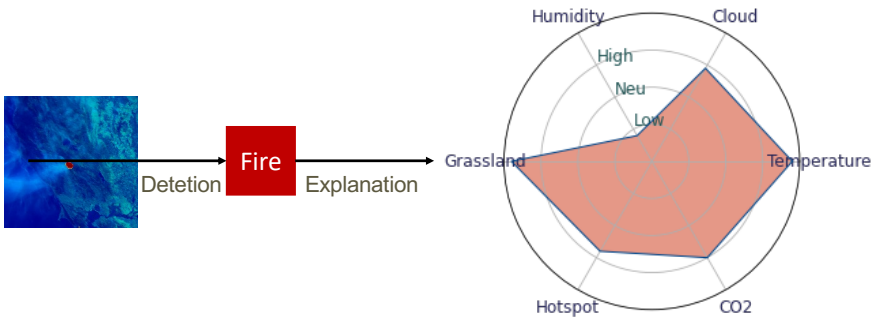

Figure 7: Explanation of fire after detection.

We argue that an optimal explanation needs to satisfy three objectives:

- *Conciseness:* In general, smaller explanations are easier for humans to understand.
- *Consistency:* Explanations should faciliate human interpretation. In practice, explanations should match the nature reasons of a wildfire.
- *Prediction power:* Explanations should be able to predict future wildfires, in order to enable proactive wildfire monitoring.

These objectives can be modelled by a reward function over explanations Formally, given an archive of semantic data streams $\mathcal{E}$ for the CEP engine, a wildfire interval I_1 and a non-wildfire interval I_0, an optimal explanation e is one that maximises a monotonic, submodular information reward R over the chosen intervals:

$$\arg\max_{e,|e|\leq k} R_{I_1,I_0}(e) \tag{1}$$

where $|e|$ is the number of predicates in the explanation (the explanation size). We limit it by k, which can be set to 7 – a common human cognitive load (Shraga et al., 2020), due to the conciseness requirement. Here,

the reward function $R : 2^{\mathcal{F}} \rightarrow \mathbb{R}$ needs to be monotonic, adding predicates to an explanation will increase the reward, due to the predictability requirement. Moreover, the reward function needs to be submodular. This satisfies the consistency requirement (the explanations should not differ drastically for similar situations), because adding predicatesoffers diminishing returns. The reward function will be formalised concretely in Eq. 4. But we first elaborate our intuition to come up with the final formulation.

The reason we base the reward function on a pair of wildfire and non-wildfire intervals is to increase the flexibility of the system. First, this corresponds to the continuous output of the CEP engine, where a match reflects a wildfire interval and other parts of the data stream without a match reflect non-wildfire intervals. As a result, users can request explanations without or before providing validation feedback. In this case, the pattern list in the CEP engine will not be updated. Second, if users choose to validate a stream segment as a wildfire or non-wildfire instance, it will automatically become a wildfire or non-wildfire interval. In this case, the pattern list in the CEP engine will be updated based on the user validation as mentioned in §6.4.

7.2. Reward function

Now the task is to define the reward function $R(e)$ to capture the importance of an explanation (or the explainability of a pattern). Explanations are comprised of predicates on the features and semantic annotations. For instance, in the streaming of raw satellite data described above, we consider each channel as a single feature. Note that the features in our work differ fundamentally from traditional hand-crafted features because they are transformed from raw data streams at a semantic level, as aforementioned in §5. This increases the interpretability of an explanation for human understanding. As such, we argue that a good selection of relevant features should satisfy the following requirements. (1) *Predictiveness:* each feature has the power to discriminate between wildfire and non-wildfire events. Selecting the features with higher predictiveness would provide more chances for users to understand the differences between two classes. (2) *Avoidance of information redundancy:* the selection of correlated features leads to high redundancy due to the conciseness requirement for explanations. Therefore, the selection of a feature should be done related to other variables that have been selected.

To combine the above requirements into a unified reward function, we implement by the following measurements.

Entropy-based predictiveness. The first criterion is to select the features with the lowest entropy, which is inspired by the dis-criminative nature of entropy and an entropy-based discretisation technique (Zhang et al., 2017) that cuts continuous values into value intervals by minimising the class information entropy. Formally, we extend the definition of entropy in the context of data streams as follows. Class entropy is the information needed to describe the class distributions between two stream segments. Consider a pair of stream segments, $\mathcal{E}_1$ and $\mathcal{E}_0$, belonging to the wildfire and non-wildfire classes, respectively. Then, the entropy of a feature f is:

$$H(f) = -p_1 \log p_1 - p_0 \log p_0 \tag{2}$$

where $p_1 = \frac{|\mathcal{E}_1|}{|\mathcal{E}_1|+|\mathcal{E}_0|}$ and $p_0 = \frac{|\mathcal{E}_0|}{|\mathcal{E}_1|+|\mathcal{E}_0|}$ reflect the probability of an event containing the feature f in the stream belonging to the wildfire and non-wildfire classes, respectively.

Redundancy penalty. Considering the reward of a feature set as a sum over the entropy of each feature would ignore the correlation between the features. As a result, the explainability information can be redundant. Therefore, there is a need to minimise the information between the selected features. Formally, the redundancy of a set of features is measured by:

$$s(e) = \sum_{f,f' \in e} w(f) r(f, f') w(f') \tag{3}$$

where the correlation coefficient $r(f, f')$ is the absolute value of the Pearson correlation coefficient. Here, we use the absolute value because the positive and negative correlations are both considered redundant. $w(.)$ serves as a scale factor since some features are more discriminative than others. Here, we use the reverse entropy of the feature itself: $w(f) = -H(f)$. (The negative sign is due to the fact that entropy reflects uncertainty; thus the lower it is, the better.)

Put it all together. Besides the predictiveness and redundancy, the reward function should also incorporate the importance of each feature. The idea behind this is that features stemming from a large group of correlated features often have a high chance of providing more understanding. Specifically, we define $q(f) = -\sum_{f' \in F} r(f, f')H(f')$, where F is the feature space. Putting it all together, we formally design the reward function as a bi-criteria combination of predictiveness and redundancy:

$$R(e) = -w \sum_{f \in e} q(f)H(f) - \sum_{f, f' \in e} H(f)r(f, f')H(f') \tag{4}$$

where w is a positive weight parameter to balance the predictiveness and redundancy components. Note that the negative sign before the second term is due to the fact that entropy reflects uncertainty; thus the lower it is, the better.

While our notion of reward is motivated by predictiveness and redundancy, it also satisfies the monotonicity and submodularity properties of the explanation discovery problem. Proofs are omitted due to space limitations.

7.3. Maximising the reward

Greedy selection. The optimal explanation problem (Eq. 1) is a submodular maximisation problem that is known to be NP-hard (Feige et al., 2011). Submodular optimisation problems are commonly addressed with greedy approximation techniques. They would work in the following way in this setting: given an explanation e, which is initially empty, at each greedy step, we select the feature f that maximally increases the reward $R(e \cup f)$ $R(e)$. This greedy mechanism is known to have an effectiveness of $1 - 1/e \approx 0.63$ (Feige et al., 2011) (i.e., the returned explanation has a reward value that is at least 63% of the optimal reward value).

Alternative reward functions. Other reward functions can also be defined, such as mutual information (Zhang et al., 2017). Mutual information also satisfies the submodularity property. However, the greedy selection based on mutual information is inefficient: the computation of $R(e \cup f) - R(e)$ requires the known conditional entropies $H(e|f)$, which vary for different subsets of features constituting e and are difficult to be updated incrementally with data streams.

7.4. Constructing predicates

Recall that the discovered explanations will be fed to the CEP engine as predictive patterns, which require predicates. In Eq. 1, we only select a set of relevant features. Now we need to define predicates based on the features. That is, we divide each feature into similar partitions so that we can use the boundary values to construct final predicates.

Feature partitioning. Similar partitions are identified from historical data and existing patterns that are monitoring the data stream. A record of partitions is maintained to facilitate fast retrieval.

Partition alignment. Once related partitions are identified, we need to map the validated segments to each similar partition, which can be done based on time or data points. In the former, an annotation is mapped to a partition based on its temporal length. In the latter, an annotation is mapped to a partition based on the ratio of data points that it occupies in the data stream. The final alignment among two methods will be chosen based on the relative difference between two partitions (the smaller it is, the better).

Interval labelling. After the new annotations are mapped to similar partitions, they need to be labelled as wildfire or non-wildfire. We assign labels through hierarchical clustering to ensure consistency in the feature space: an interval that belongs in the same cluster of an annotated wildfire is labelled as wildfire.

History confirmation. The above procedures generate much more labelled data. This, in turns, can be used to filter out false positives by evaluating the entropy of each feature on the historical data (the smaller it is, the better).

7.5. Transforming explanations into predictive patterns

Once the final features are selected, the construction of predictive patterns is straightforward. For each selected feature f_i, the segmentation has suggested the threshold value δ_i to distinguish between the two classes. However, we are only interested in the wildfire predictive patterns. Hence, we leverage the SASE language to fomalise (Gyllstrom et al., 2006) the pattern for wildfire explanations as follows:

```
FROM <streaming events>
PATTERN P1_fire SEQ(Satellite c6, Satellite c7, Satellite c14, Satellite c16))
WHERE AND c6.level==HIGH AND c7.level==HIGH AND c14.level==HIGH AND c16.level==HIGH
WITHIN 2h
RETURN(c6.time, boundary=INTERSECT(c6.boundary, c7.boundary, c8.boundary, c9.boundary)
```

It is worth noting that the pattern is the combination or sequence of fine-grained events that support the situation of interest. The pattern is defined after the `PATTERN` keyword and `SEQ` represents for a sequence of co-occurrence events. Such a definition of the pattern makes it easy for domain experts to adjust the pattern (i.e., adding or removing an event) when needed. The experimental section shows that the prediction pattern yields a good performance. However, with the guidance of experts, the amended pattern could further improve the overall performance. In the `WHERE` clause, we specify the characteristics of the data that favour the situation of interest. THE `WITHIN` and `INTERSECT` clauses help to describe the temporal and geospatial constraints for the pattern when all events occur.

The predictive patterns can be further translated into non-expert explanation, as exemplified below. Indeed, the final explanation is concise and gives better explanations to normal users about why a wildfire occurs.

```
pattern:
    c6.level==HIGH AND c7.level==HIGH AND c14.level==HIGH AND c16.level==HIGH
    AND temp.level==HIGH
  WITHIN 2h
explanation:
    High Aerosol AND High Hotspot AND High CO2 AND Hot-dry weather AND grassland
```

8. Evaluation

In this section, we report a wide range of experiments to validate our approach, including efficiency comparison, end-to-end evaluation, explanation discovery, ablation test, and adaptivity. In these experiments, we cover different empirical aspects of previous works on machine learning (for wildfire detection), data stream processing (for complex event processing), and explainable systems (for explanation discovery) such as training time, event processing time, accuracy and timeliness of detection, consistency, conciseness, and predictability of explanation, and effects of pattern update and concept drifts.

8.1. Experimental setting

Data. We evaluate the framework on the following real-world wildfire datasets. (Their key characteristics are summarised in Table 4).

- *County Fire:* The County Fire started at 2:12 pm on June 30, 2018, at the east end of Lake Berryessa in Yolo County and Napa County. The fire burned approximately 365 km^2 (Wikipedia, 2018a).
- *Camp Fire:* The Camp Fire started on November 8, 2018 in California. The fire covered an area of 620 km^2 (Gov, 2019) and was one of the most destructive fires in the history of the state.
- *Woolsey Fire:* The Woolsey Fire was a destructive wildfire that burned in Los Angeles and Ventura counties in California. The fire began at 2:22 pm PST on November 8, 2018 and burned about 392 km^2 (Wikipedia, 2018c).

- *Kincade Fire:* The Kincade Fire burned in Sonoma County, California, starting at 9:24 p.m. PST on October 23, 2019, and subsequently burned more than 314 km^2 (Wikipedia, 2019).
- *Ferguson Fire:* The Ferguson Fire was a wildfire in the Sierra National Forest, Stanislaus National Forest and Yosemite National Park in California in the USA. The fire was reported on July 13, 2018 and burned above 392 km^2 (Wikipedia, 2018b).

Table 4: Statistics of experimental datasets

Dataset	Genesis	#Events	Class Distribution [1]
County	14:12 30.06.2018	240	164:76
Camp	06:33 08.11.2018	142	65:77
Woolsey	14:22 08.11.2018	240	163:77
Kincade	21:24 23.10.2019	240	128:112
Ferguson	21:36 13.07.2018	112	69 : 43

[1] Ratio of wildfire events to non-wildfire events

Baselines. The following baselines are used for comparison:

- *GOES-AFP:* It is the state-of-the-art method (Xu et al., 2010) for the GOES system; it predicts wildfires at the pixel level according to spectral indexes.
- *Threshold-based:* It is an ensemble of threshold-based methods (Giglio et al., 2016; Schroeder et al., 2014).
- *CNN-based:* It is a common deep learning technique for wildfire detection (Toan et al., 2019; Sousa et al., 2020; Rashkovetsky et al., 2021).

Note that we have reviewed several other baselines as well, such as BAT and ER (Zervas et al., 2011). However, they are system-dependent (e.g., they utilised completely different channels of their own imagery sensors) and do not align with our settings and datasets.

Metrics. We use the following relevant performance metrics:

- *Accuracy:* It is computed from the number of true and false classifications (Toan et al., 2019; Sousa et al., 2020). By definition, $0 \leq acc \leq 1$, and the higher it is, the better. More precisely, in terms of wildfire events, it is the the ratio of correctly classified events over the total number of events, which favours true positives and penalises false positives.
- *Weighted F1-score*: It is a weighted version of the F1-score, where the weight factor is the class ratio (Krizhevsky et al., 2012). Unlike *accuracy*, it captures the imbalanced class distribution, as in our datasets.
- *Timeliness:* The timeliness, on the other hand, is computed from the average number of observations needed before classifying a segment $\mathcal{E}_i$ (i.e., after a wildfire happens and before it is actually detected). Therefore, for a given $\mathcal{E}_i \in \mathbb{E}$ and $\Delta_{\mathcal{E}_i}$ where denotes the set of observations processed before assessing the detection, one can compute the timeliness:

$$timeliness = \frac{1}{|\mathbb{E}|} \sum_{\mathcal{E}_i \in \mathbb{E}} \frac{|\Delta|_{\mathcal{E}_i}}{|\mathcal{E}_i|}$$

  By definition, $0 \leq timeliness \leq 1$, and the smaller it is, the better.
- *Feature overlap ratio:* It measures the overlap between the features appearing in the explanation and the most important features that constitute the detection results. This reflects the consistency of the constructed explanation in capturing a human's way of thinking.

8.2. Efficiency comparison

Event processing time. We compared the performances of the three components (semantic annotation, CEP engine, and explanation discovery) that were used to detect wildfires. In particular, 1000 to 6000 raw events (data signals) were sent through the system and the throughput of processing those events was measured.

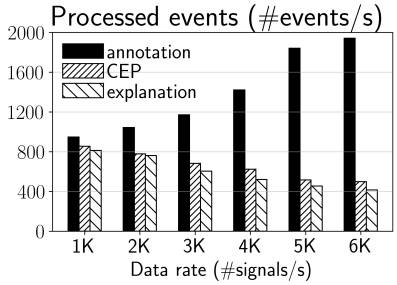

Figure 8: Processing time of framework components.

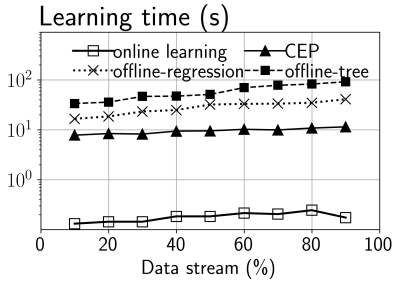

Figure 9: The average time of pattern learning.

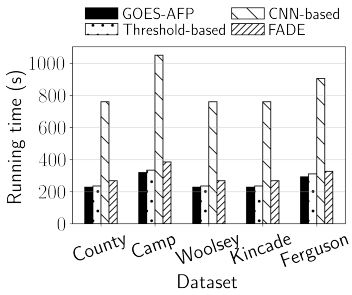

Figure 10: End-to-end running time.

As indicated by Fig. 8, it is evident that increasing the in-coming data rate causes the number of processed events per second to decrease. For the CEP engine, this is because there would be more partial matches to hold and check, making the processing time for each new event longer. For explanation discovery, this is because there would be more predicates to consider while selecting the optimal subset for explanation. Except for the annotation step, the number of processed events increases because the computation for semantic annotations is fast; thus, more data would lead to more event annotations.

Learning time. In this experiment, we evaluated the amount of time taken for the learning and deployment of patterns in the CEP engine after the user validates a detection output. Each validation is evaluated 10 times in order to calculate the average amount of time needed for learning.

Fig. 9 depicts the results of comparing our system with different learning mechanisms: (i) *offline-regression* – the regression model is recomputed when a new pattern is learned; (ii) *offline-tree* – the decision tree model is recomputed when a new pattern is learned; (iii) *online learning* – the predictive model is updated using online learning; and (iv) *CEP* – the CEP engine is updated when a new pattern is learned. The X-axis is the percentage of processed data and the Y-axis is the learning time. In general, the CEP engine updates very fast, taking less than 1 second for every case. Online learning is the first runner-up, as it performs incremental updates for the underlying predictive model. Offline learning is the slowest, as it requires recomputation from scratch, and it becomes even worse as the data stream is processed when more data is incorporated into the system.

Overall running time. We measure the total running time from end to end of our system on top of each dataset, including *County*, *Camp*, *Woolsey*, *Kincade*, and *Ferguson*. The time for explanation discovery is not counted here because it is only performed upon user request. We compare our system with various wildfire detection baselines, including GOES-AFP, threshold-based, and CNN-based methods, which were mentioned in the experimental settings section.

Fig. 10 presents the results; the X-axis represents the real wildfire datasets (*County*, *Camp*, *Woolsey*, *Kincade*, *Ferguson*) and the Y-axis is the running time (in seconds). In general, our framework (FADE) runs very fast and even faster than the lightweight methods (GOES-AFP and threshold-based) in some cases. These methods are fast because they only use indexes and thresholds for wildfire detection. Our framework uses CEP patterns and thus wildfires can be detected as soon as the data satisfies the patterns.

8.3. End-to-end evaluation

Effectiveness. In this experiment, we compare the predictive performance of our system with the wildfire detection baselines. Fig. 11 presents the results; the X-axis represents the datasets (*County*, *Camp*, *Woolsey*, *Kincade*, *Ferguson*) and the Y-axis is the weighted F1-score. It can be seen that our framework (FADE) has a comparable performance to SOTA (CNN-based), which is based on deep neural networks.

Timeliness of detection. In this experiment, we evaluate the timeliness of detection of each system, which is measured as the average time from when the wildfire happens to when the system can detect it. Table 5 presents the results. In general, our system (FADE) detects wildfires quite fast, so it is applicable for near real-time situations.

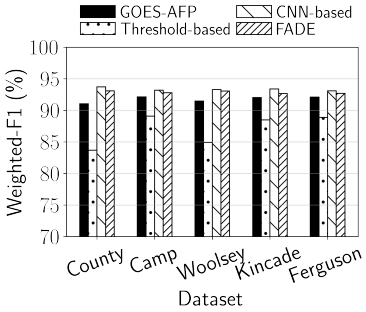

Figure 11: Effectiveness of wildfire detection techniques.

Table 5: Timeliness.

Baseline	Timeliness (h)
GOES-AFP	4 - 8
Threshold-based	4 - 6
CNN-based	2.5 - 4
FADE	1 - 3

8.4. Explanation Discovery

In this experiment, we evaluate the performance of explanation discovery. Our explanation discovery method based on the *reward* function (Eq. 4), is compared with two other selective functions, *mutual information* and *entropy*. We compare these methods using three measures: (i) *consistency* – the selected features in the explanation are compared against ground truth features; (ii) *conciseness vs. predictability* – the trade-off between the number of selected predicates in the explanation (i.e. explanation size) and the prediction accuracy when the explanation is used as a prediction model.

Consistency. First, we compare the computed explanation with the ground truth explanation, which is constructed as the set of the top-k most important features. More precisely, we consider the *CNN-based* baseline as the classifier due to its superior performance in previous experiments. Then we compute the importance score for each semantic feature based on the SHAP value (Marcílio-Jr & Eler, 2021) and select the top-k features with the highest SHAP values (k is set to 7 according to the human cognitive load (Shraga et al., 2020)).

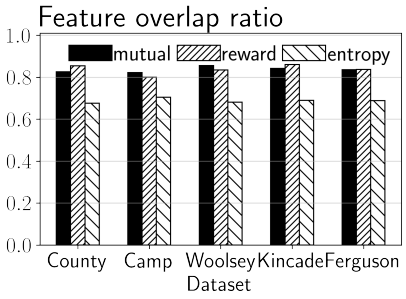

Figure 12: Consistency of explanations.

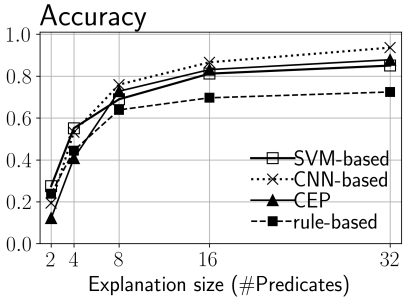

Figure 13: Conciseness of explanations.

Fig. 12 summarises the results; the X-axis represents the datasets (*County*, *Camp*, *Woolsey*, *Kincade*, *Ferguson*) and the Y-axis measures the overlapping ratio between the computed explanation and the ground truth explanation. It can be seen that our reward function achieves a high overlap ratio, which means that our explanation discovery is consistent with the true explanation of wildfire detection. Note that the mutual information function has a comparable performance to ours; however, it would be much slower due to the computation of joint conditional entropies.

Conciseness vs. predictability. Fig. 13 shows the sizes of explanations (the number of predicates) and the corresponding accuracy when using a prediction model based on these explanations. Different prediction models are evaluated: *CEP* – the predicates are used directly in the CEP engine for wildfire detection; *CNN-based* – the predicates are used as the attention mechanism in the classification neural network; *SVM-based* – the predicates are used as the only features for an SVM; and *rule-based* – the predicates are used for a rule-based classifier.

In general, the accuracy increases quickly when the size of the explanation increases, especially from 4 to 8. However, we observe a diminishing return when we increase the explanation size further. Moreover, our *CEP* engine achieves a very competitive performance against the *CNN-based* method, although the former works on streaming data while the latter works on static data.

8.5. Ablation test

In this experiment, we evaluate the importance of each component of our approach, including semantic annotation, the CEP engine, and explanation discovery. To prove that the design choice of using these components together would achieve a great performance, we alter each component with a different approach. More precisely, we either use the semantic events, or not, when the raw data stream goes through the annotation component. For explanation discovery, we consider using either the reward-based function or the mutual information function. Finally, instead of using the CEP engine, we use the CNN-based classifier for wildfire prediction.

Table 6: Importance of each model component.

	Weighted F1	Running time(s)	Earliness (h)
semantic + CEP + reward-based explanation	92.9	297	1-3
raw + CEP + reward-based explanation	93.2	538	1-3
semantic + CEP + information-based explanation	92.6	1986	3-6
semantic + classifier	81.7	878	5-10
raw + classifier	83.6	882	5-10

Table 6 summarises the results, which are averaged over all datasets. The running time is measured as the total running time over the dataset. It can be seen that the CEP engine runs much faster than the classifier and helps to detect wildfires early. The semantic annotation achieves a slightly lower weighted F1-score than its raw version, as the deep neural networks can discover latent features. However, the difference is not significant compared to the fact that semantic annotations provide much better explanations and also reduce the running time. It is worth noting that the information-based version of explanation discovery requires much more time due to the computation of joint conditional entropies.

8.6. Adaptivity

Effects of pattern update. As explained in §6.4, our system allows the patterns to be confirmed and updated by users. This would help the system to be adaptive to data streams. In this experiment, we evaluate this ability by simulating user feed-back via the ground-truth explanation constructed in the previous experiments. More precisely, for each pattern learned, we will randomly confirm if it exists in the ground truth. The ground truth was provided in real datasets (*County*, *Camp*, *Woolsey*, *Kincade*, *Ferguson*).

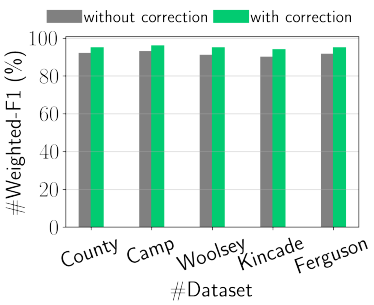

Figure 14: Effects of pattern update.

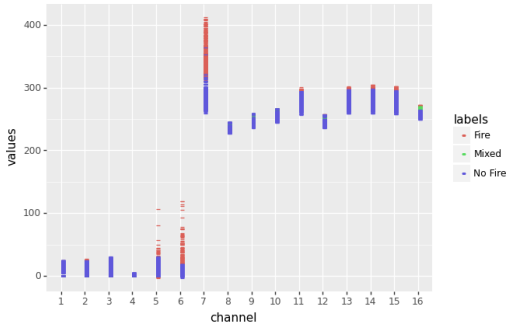

Figure 15: Explainability as the discriminative power of data streams.

Fig. 14 presents the results, in which two versions of our system, with correction (pattern learning with user feedback) and without correction, are compared across datasets in terms of the weighted F1 score. It can be seen that the corrected version is always better than the other version, indicating the adaptivity of our system to user feedback.

Effects of concept drift. Concept drift is the phenomenon in which the distributions and the characteristics of data streams change, making the predictive model prone to error when classifying new wildfire events. To evaluate the robustness of our system against these adverse conditions, we create synthetic data streams by varying the values of real data streams with different distributions, including normal and skewed distributions. We also vary the input rate of data streams from low to high and consider two versions of our system, one with user feedback and one without it. The user feedback is simulated in the same manner that it was for the previous experiment, with the probability of giving feedback set to 50%.

Table 7: Accuracy in adverse conditions

Data dist.	Input rate	Classifier-based	FADE w/o user feedback	FADE
normal	low	0.734	0.844	0.917
	high	0.732	0.845	0.918
skewed	low	0.671	0.824	0.903
	high	0.673	0.823	0.901

Table 7 presents the results: we compare our method with the classifier baseline (the average of all classifiers). It can be seen that the input rate does not affect our system, as the CEP engine can store the partial matches when new data arrives. In general, the skewed data distribution makes the predictive models worse due to the increased appearance of concept drift. However, this difference is not significant in our system, especially with user feedback.

8.7. Qualitative case-studies

To give a better intuition about the explanation predicates, we visualise the channel values in Fig. 15. We order the values of a channel and partition them by colour; red for values in fire intervals, blue for values in non-fire intervals, and green for values in both cases. The predictive analysis of multivariate data streams is typically based on comparing the extracted events to a pre-defined threshold. Patterns can be detected when a measurement is lower or higher than a threshold.

In Fig. 15, we note that some channels show a clear distribution between the fire and non-fire instances. Some other channels had less clear differences between two cases, but in general lower values indicate non-fire instances. Finally, some features are mixed or relevant to only one label. Similar to (Zhang et al., 2017), our reward function favours the apparently separated channels and penalises the mixed ones.

9. Discussions

In this section, we summarise the experimental findings as well as the relevancy of our framework in practical use-cases. Then, we discuss our limitations and future works.

Summary of findings. In general, our framework was able to detect wildfires early due to the real-time nature of complex event processing. Moreover, we also show that our explanation in the form of logical patterns brings usefulness for users in terms of conciseness, consistency, and predictability. Our ablation tests also validate the importance of our semantic annotation pipeline to improve running time and earliness of detection as well as simplify the understanding of wildfire phenomena.

Our framework is relevant to existing systems in the sense that each component of our framework can be reused. For example, the semantic annotation pipeline can be integrated to improve the classification features of existing systems based on online classifiers (Phan et al., 2020). Moreover, our complex event processing component can encode rule-based patterns if domain knowledge for wildfire detection is known

before-hand (Boubeta-Puig et al., 2021). Finally, our framework also adapts to continuous user feedback and concept drifts, showing the flexibility of our system.

Table 8 gives a brief comparison of our system against previous related works.

Table 8: Functionality comparison against previous works

System	Feature design	Real-time processing	Explainability	Performance
Ours	Semantic	Yes	On-the-fly	High
Deep learning classifiers [1]	No	No	No	High
Machine learning classifiers [2]	Spectral only	Near	No	Low/Medium
Explainable ML systems [3]	Model-intrinsic	No	Post-processing	Low/Medium

[1] e.g. (Phan et al., 2020; Sousa et al., 2020) [2] e.g. (Xu et al., 2010; Xu & Zhong, 2017) [3] e.g. (Angelov et al., 2021; Ma et al., 2021)

Limitations. As our framework focuses on the earliness of wildfire detection and real-time processing of data streams, it could introduce false alarms. Moreover, as semantic annotations are rather domain-specific, one may observe more false alarms in other domains. However, in vision of these limitations, we already designed our system to be updated upon new user feedback, making the system more robust overtime.

Future works. We plan to extend our system capability in dealing with high-velocity data streams by means of load shedding (Zhao et al., 2020) as well as in integrating external data sources by means of prefetching (Zhao et al., 2021). We also plan to apply our system to other domains on top of satellite data such as agriculture monitoring (Tam et al., 2020).

10. Conclusion

In this work, we designed and developed a real-time wildfire detection framework with the capability of providing human-understandable explanations for detection results. The frame-work is enabled by the realisation of three important components. First, a semantic annotation component is responsible for transforming raw data streams into semantic event streams that are interpretable. Second, a CEP engine processes these events to find matching instances of wildfire-predictable pat-terns. Third, these patterns are extracted and learned upon user validation via an explanation discovery component, which is also responsible for constructing final explanations for users. Experimental results show that our approach outperforms the baselines, processes data streams at high speeds, detects wildfires in a timely manner, and adapts to several adversarial conditions.

References

Angelov, P. P., Soares, E. A., Jiang, R., Arnold, N. I., & Atkinson, P. M. (2021). Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *11*, e1424.

Babüroğlu, E. S., Durmuşoğlu, A., & Dereli, T. (2021). Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection. *Expert Systems with Applications*, *163*, 113786.

Blaylock, B. (2019). Goes-16 data stream on amazon. `http://home.chpc.utah.edu/~u0553130/Brian_Blaylock/cgi-bin/goes16_download.cgi`.

Boubeta-Puig, J., Rosa-Bilbao, J., & Mendling, J. (2021). Cepchain: A graphical model-driven solution for integrating complex event processing and blockchain. *Expert Systems With Applications*, *184*, 115578.

Cafaro, M., Melle, C., Pulimeno, M., & Epicoco, I. (2021). Fast online computation of the qn estimator with applications to the detection of outliers in data streams. *Expert Systems with Applications*, *164*, 113831.

Chuvieco, E., González, I., Verdú, F., Aguado, I., & Yebra, M. (2009). Prediction of fire occurrence from live fuel moisture content measurements in a mediterranean ecosystem. *International Journal of Wildland Fire*, *18*, 430–441.

De Laat, A., Stein Zweers, D. C., Boers, R., & Tuinder, O. N. (2012). A solar escalator: Observational evidence of the self-lifting of smoke and aerosols by absorption of solar radiation in the february 2009 australian black saturday plume. *Journal of Geophysical Research: Atmospheres*, *117*.

Doolin, D. M., & Sitar, N. (2005). Wireless sensors for wildfire monitoring. In *Smart Structures and Materials 2005: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems* (pp. 477–485). International Society for Optics and Photonics volume 5765.

Duong, T. H., Nguyen, N. T., Truong, H. B., & Nguyen, V. H. (2015). A collaborative algorithm for semantic video annotation using a consensus-based social network analysis. *Expert systems with applications*, *42*, 246–258.

Feige, U., Mirrokni, V. S., & Vondrák, J. (2011). Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, *40*, 1133–1153.

Flasse, S., & Ceccato, P. (1996). A contextual algorithm for avhrr fire detection. *International Journal of Remote Sensing*, *17*, 419–424.

Giglio, L., Schroeder, W., & Justice, C. O. (2016). The collection 6 modis active fire detection algorithm and fire products. *Remote Sensing of Environment*, *178*, 31–41.

Gov, C. (2019). Camp fire. `https://www.fire.ca.gov/incidents/2018/11/8/camp-fire/`.

Gyllstrom, D., Wu, E., Chae, H.-J., Diao, Y., Stahlberg, P., & Anderson, G. (2006). Sase: Complex event processing over streams. *arXiv preprint cs/0612128*, .

Hally, B., Wallace, L., Reinke, K., Jones, S., & Skidmore, A. (2019). Advances in active fire detection using a multi-temporal method for next-generation geostationary satellite data. *International journal of digital earth*, *12*, 1030–1045.

Ismael, A. M., & Şengür, A. (2021). Deep learning approaches for covid-19 detection based on chest x-ray images. *Expert Systems with Applications*, *164*, 114054.

Jia, K., Wei, X., Gu, X., Yao, Y., Xie, X., & Li, B. (2014). Land cover classification using landsat 8 operational land imager data in beijing, china. *Geocarto International*, *29*, 941–951.

Kim, H.-W., & Yang, S.-H. (2019). Region of interest–based segmented tiled adaptive streaming using head-mounted display tracking sensing data. *International Journal of Distributed Sensor Networks*, *15*, 1550147719894533.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Lee, H. (2007). Intergovernmental panel on climate change, .

Ma, L., Ding, Y., Wang, Z., Wang, C., Ma, J., & Lu, C. (2021). An interpretable data augmentation scheme for machine fault diagnosis based on a sparsity-constrained generative adversarial network. *Expert Systems with Applications*, (p. 115234).

Marcílio-Jr, W. E., & Eler, D. M. (2021). Explaining dimensionality reduction results using shapley values. *Expert Systems with Applications*, *178*, 115020.

Mousheimish, R., Taher, Y., & Zeitouni, K. (2017). Automatic learning of predictive cep rules: bridging the gap between data mining and complex event processing. In *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems* (pp. 158–169).

NOAA/NESDIS/STAR (2013). Goes-r advanced baseline imager (abi) algorithm theoretical basis document for fire/hot spot characterization. `http://tiny.cc/goesfireproduct`.

NOAA/NESDIS/STAR (2018). Abi aerosol detection product. `http://tiny.cc/goesaerosol`.

Ottmar, R. D., & Alvarado, E. (2004). Linking vegetation patterns to potential smoke production and fire hazard. In *In: Murphy, Dennis D. and Stine, Peter A., editors. Proceedings of the Sierra Nevada Science Symposium. Gen. Tech. Rep. PSW-GTR-193. Albany, CA: Pacific Southwest Research Station, Forest Service, US Department of Agriculture: 93-96*. volume 193.

Phan, T. C., Nguyen, T. T., Hoang, T. D., Nguyen, Q. V. H., & Jo, J. (2020). Multi-scale bushfire detection from multi-modal streams of remote sensing data. *IEEE Access*, *8*, 228496–228513.

Rashkovetsky, D., Mauracher, F., Langer, M., & Schmitt, M. (2021). Wildfire detection from multisensor satellite imagery using deep semantic segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 7001–7016.

Ren, Z., Wu, B., Sun, Q., & Wu, M. (2019). Simultaneous learning of reduced prototypes and local metric for image set classification. *Expert Systems with Applications*, *134*, 102–111.

Schroeder, W., Oliva, P., Giglio, L., & Csiszar, I. A. (2014). The new viirs 375 m active fire detection data product: Algorithm description and initial assessment. *Remote Sensing of Environment*, *143*, 85–96.

Shraga, R., Scharf, C., Ackerman, R., & Gal, A. (2020). Incognitomatch: Cognitive-aware matching via crowdsourcing. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (pp. 2753–2756).

Sousa, M. J., Moutinho, A., & Almeida, M. (2020). Wildfire detection using transfer learning on augmented datasets. *Expert Systems with Applications*, *142*, 112975.

Tam, N. T., Dat, H. T., Tam, P. M., Trinh, V. T., Hung, N. T., Huynh, Q.-T., & Jo, J. (2020). Monitoring agriculture areas with satellite images and deep learning. *Applied Soft Computing*, (p. 106565).

Toan, N. T., Cong, P. T., Hung, N. Q. V., & Jo, J. (2019). A deep learning approach for early wildfire detection from hyperspectral satellite images. In *2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA)* (pp. 38–45). IEEE.

UCAR (2019). Netcdf. `https://www.unidata.ucar.edu/software/netcdf/docs/`.

USDA (2019). Fire detection maps. `https://fsapps.nwcg.gov/afm/activefiremaps.php?op=info`.

Wikipedia (2018a). County fire. `https://en.wikipedia.org/wiki/County_Fire`.

Wikipedia (2018b). Ferguson fire. URL: `https://en.wikipedia.org/wiki/Ferguson_Fire`.

Wikipedia (2018c). Woolsey fire. `https://en.wikipedia.org/wiki/Woolsey_Fire`.

Wikipedia (2019). Kincade fire. `https://en.wikipedia.org/wiki/Kincade_Fire`.

Wu, E., Diao, Y., & Rizvi, S. (2006). High-performance complex event processing over streams. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data* (pp. 407–418).

Xu, G., & Zhong, X. (2017). Real-time wildfire detection and tracking in australia using geostationary satellite: Himawari-8. *Remote Sensing Letters*, *8*, 1052–1061.

Xu, W., Wooster, M., Roberts, G., & Freeborn, P. (2010). New goes imager algorithms for cloud and active fire detection and fire radiative power assessment across north, south and central america. *Remote Sensing of Environment*, *114*, 1876–1895.

Yan, Z., Chakraborty, D., Parent, C., Spaccapietra, S., & Aberer, K. (2011). Semitri: a framework for semantic annotation of

heterogeneous trajectories. In *EDBT* (pp. 259–270).
Yuan, X., Shi, J., & Gu, L. (2020). A review of deep learning methods for semantic segmentation of remote sensing imagery. *Expert Systems with Applications*, (p. 114417).
Zervas, E., Mpimpoudis, A., Anagnostopoulos, C., Sekkas, O., & Hadjiefthymiades, S. (2011). Multisensor data fusion for fire detection. *Information Fusion*, *12*, 150–159.
Zhang, H., Diao, Y., & Meliou, A. (2017). Exstream: Explaining anomalies in event stream monitoring. In *Proceedings of the 20th international conference on extending database technology (EDBT)*.
Zhao, B., van der Aa, H., Nguyen, T. T., Nguyen, Q. V. H., & Weidlich, M. (2021). Eires: Efficient integration of remote data in event stream processing. In *SIGMOD* (pp. 2128–2141).
Zhao, B., Hung, N. Q. V., & Weidlich, M. (2020). Load shedding for complex event processing: Input-based and state-based techniques. In *ICDE* (pp. 1093–1104).