SoftwareX 36 (2026) 103135

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

FuzzyDiary: An open-source framework for morphology-aware and traceable linguistic summarization of clinical time series

Juan Francisco Gaitán-Guerrero [a,*], José Luis López-Ruiz [a], Carmen Martínez-Cruz [b], Macarena Espinilla [a]

[a] Computer Science Department, University of Jaén, 23071 Jaén, Spain
[b] Department of Languages and Computer Systems, University of Granada, 18071 Granada, Spain

ARTICLE INFO

Keywords:
Fuzzy linguistic summarization
Time series description
Morphological event detection
Interpretable data modeling

ABSTRACT

Univariate clinical Time Series (TS) carry information that is often inaccessible to non-technical users behind dense numerical streams. We present FuzzyDiary, an open-source Python framework for protoform-based, morphology-aware linguistic summarization of univariate TS, and is domain-independent by design with clinical monitoring as its motivating use case. FuzzyDiary couples Ramer–Douglas–Peucker (RDP) simplification with an extensible catalog of morphological events declared in the configuration (peaks, valleys, sustained rises and falls, plateaus, oscillations, regime changes), and uses fuzzy protoforms to produce graded linguistic descriptions of detected events. Activated protoforms and detected events are then synthesized and rendered as a single cohesive narrative per reporting period. A self-contained interactive report links every textual statement to the underlying segment of the original series, providing full traceability that purely textual outputs lack and supporting reproducible clinical reporting workflows.

Metadata

Code metadata.

Nr.	Code metadata description	Please fill in this column
C1	Current code version	v0.2.0
C2	Permanent link to code/repository used for this code version	GitHub: https://github.com/ASIATIC257UJA/fuzzydiary/tree/v0.2.0 Zenodo: https://doi.org/10.5281/zenodo.22856221
C3	Legal Code License	MIT License
C4	Code versioning system used	Git
C5	Software code languages, tools, and services used	Python, Web browser
C6	Compilation requirements, operating environments & dependencies	Cross-platform. Python 3.10+. NumPy, Pandas, rdp, Pydantic, PyYAML, Plotly, Jinja2.
C7	If available Link to developer documentation/manual	Usage manual in the repository `README`, covering installation, the staged command-line interface and a worked example: https://github.com/ASIATIC257UJA/fuzzydiary#readme
C8	Support email for questions	*jgaitan@ujaen.es*

1. Motivation and significance

Univariate clinical TS are produced continuously by a wide range of monitoring technologies. These signals contain clinically meaningful information that is often hidden behind dense numerical streams and conventional line-chart visualizations, both of which place a substantial cognitive load on practitioners and limit accessibility for non-technical

* Corresponding author.
Email addresses: jgaitan@ujaen.es (J.F. Gaitán-Guerrero), llopez@ujaen.es (J.L. López-Ruiz), cmcruz@ugr.es (C. Martínez-Cruz), mestevez@ujaen.es (M. Espinilla).

https://doi.org/10.1016/j.softx.2026.103135
Received 3 June 2026; Received in revised form 6 October 2026; Accepted 11 October 2026
Available online 14 October 2026
2352-7110/© 2026 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

users such as patients, caregivers, or domain researchers. The growing demand for interpretable, narrative-style reporting in clinical workflows has motivated a long line of research on linguistic descriptions of TS, in which numerical observations are deterministically lifted to short natural-language statements with explicit fuzzy semantics and quantifiable quality measures.

Fuzzy Linguistic Summarization (FLS), originating in the protoform-based framework of Yager [1], provides a principled grounding layer for such reporting. Each statement, for example of the form "Q of the objects in R are S", is associated with a truth degree (and, in extended frameworks, with focus and simplicity measures), ensuring that the resulting narrative is both faithful to the underlying data and traceable. Although originally conceived for describing collections of objects, FLS has been successfully extended to the analysis and summarization of temporal data, enabling the generation of interpretable descriptions of trends, patterns, and events in time series [2]. In contrast to recent generative approaches based on Large Language Models (LLMs) [3], FLS produces deterministic, reproducible statements whose semantics are fully traceable to the input signal. This property is particularly valuable in clinical and industrial settings, where factual reliability is non-negotiable.

Despite the maturity of the underlying theory, the open-source ecosystem for FLS on TS data is fragmented and incomplete. General-purpose fuzzy toolkits, such as scikit-fuzzy [4] and Simpful [5], provide membership functions and fuzzy inference engines, but they do not offer a complete protoform-based pipeline for linguistic summarization. A more recent framework, PyIT2FLS [6], extends this functionality to type-1 and interval type-2 fuzzy sets; however, it remains primarily focused on fuzzy inference and modeling rather than on the generation of linguistic summaries from data. Conversely, a substantial body of research has demonstrated the value of protoform-based summarization across application domains, with a particularly active line of work in health and ambient monitoring: linguistic summaries of heart-rate streams in patients with ischemic heart disease [7], bed-restlessness monitoring for eldercare [8] or behavioral patterns [9], among others. These contributions converge on a clear message: protoform-based linguistic summaries are an effective intermediate representation for clinical and behavioral TS, but each is realized as a domain-specific prototype rather than as reusable software. Recent platforms such as FuzzyIoT [10] move toward generic descriptive analysis of sensor streams through ad-hoc protoforms, yet they remain centered on instantaneous data characterization and do not provide a progressive, event-aware analysis over time with a configurable catalog of clinically relevant patterns. Comprehensive data-to-text frameworks such as ICA2TEXT [11] address the temporal dimension through a rich fuzzy temporal ontology and a full natural language generation pipeline, but it pursues a goal close to ours under constraints that limit its reuse: the system is not distributed as an open-source library, its protoform layer is restricted to concrete quantifiers, and its discourse structure is tailored to a specific reporting template. The relevance of closing these gaps is reinforced by the emerging use of fuzzy linguistic protoforms as a substrate for Explainable Artificial Intelligence, as illustrated by frameworks such as PLENARY [12] or SUMMARIA [13]. Nonetheless, it must be noted that the generation of linguistic descriptions of data has opened new horizons in the task of leveraging LLMs for the description of raw data, as attested in [14,15].

Addressing the latter, FuzzyDiary is presented as an open-source Python framework for protoform-based, morphology-aware linguistic summarization of univariate clinical TS. Although clinical monitoring is its motivating use case, FuzzyDiary is domain-independent by design. FuzzyDiary organizes the analysis around a predefined reporting period and produces graded linguistic descriptions of detected morphological events defined through the configuration file. The framework integrates the RDP simplification algorithm [16], which yields a compact piecewise-linear representation of each TS over which a configurable, extensible catalog of morphological events is applied; events feed an event-aware protoform layer that emits graded reports with truth, focus, and simplicity mechanisms. A subsequent synthesis layer establishes how the resulting statements include or overlap one another, and a generation layer renders them as a cohesive narrative. A self-contained interactive HTML report links every textual statement to the underlying segment of the original series, providing the traceability that purely textual outputs lack. The contribution of FuzzyDiary is therefore not a new fuzzy formalism, but a coherent, reproducible, and monitoring-oriented software architecture that fills a concrete gap between tabular FLS libraries and numerical TS toolkits, and supports interpretable reporting workflows in which natural language outputs remain fully traceable to the underlying signal.

2. Software description

FuzzyDiary follows a deterministic, configuration-driven design philosophy aligned with the principles of reproducible scientific software. Given a univariate clinical TS and a configuration file that defines linguistic variables, morphological events, and protoform templates, the framework produces a reproducible chain of artifacts: a piecewise-linear simplification of the signal, a list of detected events with associated attributes, a graded linguistic summary, a synthesized narrative for each reporting period, and a self-contained interactive HTML report that provides full traceability, linking every textual statement to the original series that supports it. The unit of description is the calendar day, which matches the natural reporting unit of clinical practice and keeps the resulting narrative aligned with how clinicians, patients, and caregivers reason about longitudinal monitoring data. Despite clinical monitoring being the motivating use case, the design is domain-agnostic, since any univariate signal with a meaningful daily structure is handled by the same modules.

2.1. Software architecture

FuzzyDiary is organized as a pipeline of six core modules plus a reporting layer, each with a well-defined input/output contract that allows users to invoke the full pipeline or any intermediate stage in isolation. Fig. 1 provides a schematic overview of the workflow; the modules are summarized below.

io ingests univariate TS from common formats (CSV, JSON, Pandas DataFrame) and standardizes them into a canonical representation indexed by calendar day. It handles resampling to a user-specified sampling rate, imputation of missing values, and segmentation of the continuous signal into daily windows.

simplify reduces each daily window to a compact piecewise-linear representation using the RDP algorithm [16], with a configurable tolerance parameter ϵ and a heuristic for suggesting a sensible default from the observed signal variance. The output is a sequence of segments characterized by start time, end time, slope, amplitude, and duration, which serves as the substrate for morphological event detection.

events detects morphological events over the RDP-simplified signal. Events are declared in the configuration rather than built into the source code. Each event is parameterized by the terms it spans, its minimum and maximum duration, and the gaps that may be merged; events may also be anchored to, or composed from, previously detected ones, enabling domain-specific patterns such as *recovery from low*, a return to *normal* values after a drop. Detected events expose quantitative attributes (intensity, duration, time-of-day) that are subsequently mapped to fuzzy linguistic terms.

describe produces the per-day linguistic report. For each day, it combines the detected events and aggregated daily statistics with the user-defined linguistic variables, fuzzy predicates, qualifiers (R) and quantifiers (Q) to evaluate a set of protoform templates of the canonical form "Q of the X are A", "In R, Q of the X are A", and "Of the X that are A, Q are B". Each instantiated protoform is associated with

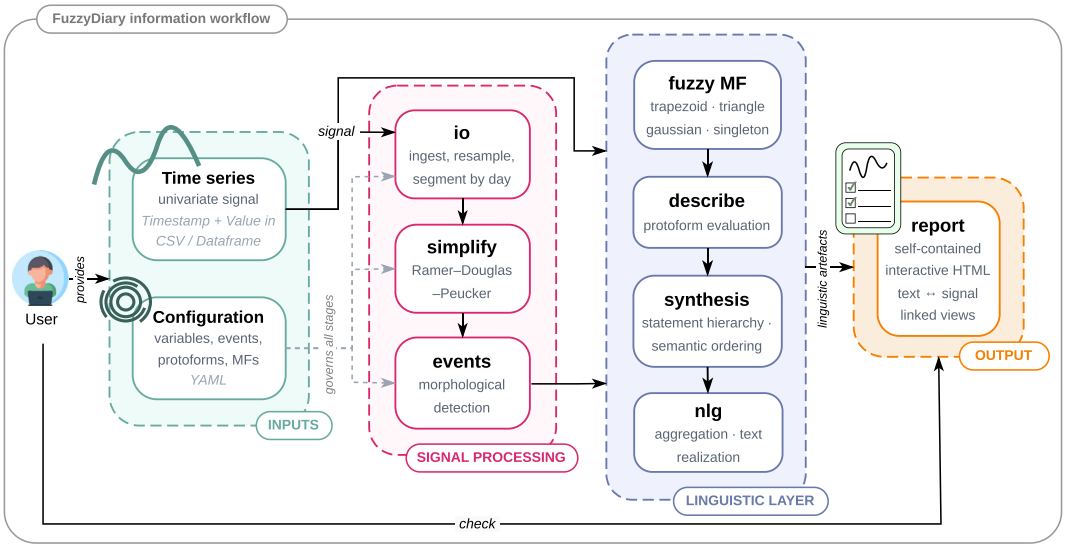


Fig. 1. Overview of the FuzzyDiary architecture and data flow. Inputs (a univariate *TS* and a YAML *configuration*) feed a signal-processing layer (*io*, *simplify*, *events*) and a linguistic layer (*fuzzy MF*, *describe*, *synthesis*, *nlg*); all linguistic artifacts converge in a self-contained interactive HTML *report*. Each module exposes a stable input/output contract and can be invoked independently or as part of the full pipeline.

a truth degree, following the standard quality framework for linguistic summarization [2,8]. The output is an ordered list of statements and a rendered daily narrative.

synthesis establishes how the statements produced by *describe* relate to one another, before any text is generated. It reads the event catalog to determine which linguistic terms each event encompasses, which events specialize others under a stricter temporal constraint, and in which direction along the linguistic scale an event declared over an anchor moves; it derives the ordering of summarizers and quantifiers from the centroids of their membership functions; and it returns, for every temporal scope, a hierarchy in which a more specific statement is attached below the more general statement that covers it. The module computes no new evidence and discards no statement: it only makes explicit the relations that the configuration already declares.

nlg turns that hierarchy into a cohesive paragraph through a natural language generation process: document planning, aggregation, lexicalization and surface realization (detailed in Section 2.2). Every generated clause retains the statements and events it summarizes, so the narrative is as traceable as the graded list it summarizes.

report produces a self-contained interactive HTML document combining the textual summaries, Plotly-based visualizations of the original series with detected events annotated, and a highlighted linked view of the supporting segment of the signal. Self-containment is literal: the plotting library is embedded in the generated file and no resource is fetched at opening time, so the report renders identically offline and can be archived as a single artifact. Every clause of the daily narrative carries the time interval of the segment that supports it; clicking a clause frames that interval on the signal and zooms the view to it, and clicking it again restores the full view. The frame is drawn as an outline rather than as a tint, so the detected events keep the colors the legend assigns to them. Clauses that rest on day-level statements alone are linked to the temporal support of the contextual qualifier under which they were grouped, so no sentence of the report is left without a visual counterpart. The chart is addressed through a fixed element identifier, which additionally makes two runs on the same input byte-identical. The report requires no server and can be archived, shared, or embedded in clinical documentation workflows.

A configurable membership function layer underpins the linguistic variable and, qualifier and quantifier declarations. Membership functions are specified per-term in the configuration file and may be trapezoidal (four breakpoints $a \le b \le c \le d$), triangular (three breakpoints $a \le b \le c$), Gaussian (μ, σ), or singleton (a crisp value with optional tolerance). This design decouples the choice of fuzzy partition from the rest of the pipeline: the event detection, description, and report modules consume membership degrees through a uniform interface and are unaffected by which function type is selected. All four shapes are available for linguistic variables, contextual dimensions or qualifiers, and quantifiers alike.

The whole pipeline is driven by a single human-readable YAML configuration validated against a Pydantic schema, ensuring that any analysis can be exactly reproduced from the configuration file, the input signal, and the library version.

2.2. Narrative synthesis

Protoform evaluation emits one statement for every pair consisting of qualifier and summarizer whose degree of truth reaches the acceptance threshold. Because the terms of a linguistic variable overlap by design, a single sample has membership in more than one term, and several statements may therefore describe the same stretch of signal without anything in the output that expresses how they relate. FuzzyDiary separates what is said from how it is said: an algorithmic stage establishes the relations between the activated protoforms, and a template-based stage realizes them as text.

Given the statements instantiated for one reporting period, the synthesis stage proceeds as follows.

1. *Orderings.* The terms of the linguistic variable and the quantifiers are ranked by the centroid of their membership functions. From the event catalog, each event label yields the set of terms it encompasses; an event specializes another when its term set is contained in that of the other under a stricter minimum duration; and an event declared over an anchor is assigned a direction by comparing the mean rank of the terms it reaches with that of the terms of its anchor.
2. *Scoping.* Statements qualified by a term of the primary contextual dimension form the backbone of the description. The remaining ones are attached to the scope whose membership function is maximal at the centroid of their own qualifier, so that a

Table 1
Surface realization templates. Placeholders in braces are filled from the synthesized structure: R is the scope, A is a summarizer, Q is a quantifier and R' is the scope of a qualification coming from a secondary contextual dimension. The adverbial readings of quantifiers and the natural-language form of every label are taken from an optional `lexicon` section of the configuration, with defaults that read underscores as spaces.

Element	Template
Opening clause	`In the {R}, {signal} was {adv(Q)} {A1 and . . . and An}`
Rise or fall	`it {rose|fell} in the {R}, where it was {adv(Q)} {A1 and . . . and An}`
Continuation	`it remained {adv(Q)} {A1 and . . . and An} in the {R}`
Subordinated summarizer	`{sub(Q)} reaching {A} values`
Qualification, same A and Q	`also around {R'}`
Qualification, same A	`{adv(Q)} around {R'}`
Qualification, new A	`{adv(Q)} {A} around {R'}`
Uncovered sibling	`and also {adv(Q)} {A}`
One detected event	`a period of {kind} was detected`
Several detected events	`{kind1}, . . ., {kindN} were detected`

statement about lunch is reported inside the sentence describing the afternoon.

3. *Ordering by specificity.* Within a scope, statements are sorted by increasing extremity, $\text{ext}(A) = |\text{rank}(A) - (n-1)/2|$, where n is the number of terms, so that the most general description is considered first.
4. *Subsumption.* A statement s is attached below the first root r such that both share a scope, their summarizers belong to the same event family or the degree of subsethood between them reaches a configurable threshold, $\text{ext}(A_s) > \text{ext}(A_r)$, and the quantifier of r is at least as demanding as that of s. Otherwise s becomes a root of its own.
5. *Event nesting.* A detected event is nested below another one that specializes it and contains it in time, so that a long episode is not reported twice under two labels of different granularity.
6. *Ranking.* Roots are ordered by quantifier and then by degree of truth; subordinated statements are ordered by decreasing extremity.

The generation stage then operates on the resulting structure in four steps. *Document planning* visits the scopes in the order declared in the configuration, which makes the narrative chronological, and assigns each of them a rhetorical relation (rise, fall or continuation) by comparing the rank of its main summarizer with that of the preceding scope. *Aggregation* merges adjacent scopes described by the same quantifier and the same summarizer with nothing else to report. *Elision* removes what the sentence has already said: sibling statements sharing the quantifier of the main one are coordinated with it rather than being repeated in a clause of their own, a qualification that repeats both the summarizer and the quantifier of its host is reduced to the scope it adds, and an event whose direction coincides with the rhetorical relation of the sentence is not named, since the verb has already stated it. *Surface realization* instantiates the templates of Table 1, where adv(Q) and sub(Q) denote the adverbial readings of a quantifier in main and subordinate positions.

Both stages are deterministic and depend only on the configuration, the input signal and the library version, so a given analysis always yields the same narrative. The synthesis reorganizes the evidence but never removes it: the detected events and the complete list of instantiated protoforms with their degrees of truth remain in the report, below the narrative, which allows the user to verify that the protoforms are activated exactly as the configuration prescribes.

2.3. Software functionalities

The principal functionalities exposed by FuzzyDiary are:

- **Standardized ingestion** of univariate clinical TS from CSV, JSON, or in-memory Pandas DataFrames, with resampling, imputation, and segmentation into calendar days.
- **Geometric simplification** of each daily signal via RDP, with configurable tolerance and a built-in heuristic for default selection.
- **Morphology-aware event detection** over the RDP approximation, through an extensible YAML syntax in which events are declared as patterns over the linguistic terms of the signal, with duration constraints, anchoring and composition, so that new domain-specific events can be added without modifying the source code.
- **Daily linguistic description** through protoform evaluation, producing ranked statements with truth and simplicity measures.
- **Synthesis of activated protoforms**, which detects the inclusion, specialization and scope relationships that hold between the instantiated statements and between the detected events, using only the relations declared in the configuration.
- **Narrative generation** of a cohesive paragraph per reporting period through document planning, aggregation, lexicalization and template-based surface realization, with every clause linked to the statements and events it summarizes.
- **Self-contained interactive HTML reports** that link every textual statement to the supporting segment of the original series, requiring no server or runtime dependencies beyond a modern web browser.
- **Generalized membership functions**: linguistic variables and quantifiers support trapezoidal, triangular, Gaussian, and singleton membership functions declared in the YAML configuration, so the framework is not restricted to piecewise-linear fuzzy sets.
- **Configuration-driven reproducibility**: every analysis is fully specified by a YAML configuration file validated through a Pydantic schema, so identical inputs always produce identical outputs.

3. Illustrative examples

This section illustrates the practical use of FuzzyDiary through its command-line interface and its Python API. The examples below operate on a generic univariate TS and a YAML configuration. All examples are reproducible from the snippets and configuration files shipped with the repository.

3.1. Installation and command-line use

FuzzyDiary is publicly available on GitHub and can be installed directly from the repository with a single command:

```
pip install git+https://github.com/ASIATIC257UJA/fuzzydiary.git
```

The package ships with a command-line entry point that runs the complete pipeline from a YAML configuration:

```
fuzzydiary run --config config.yaml --series data/signal.csv \
               --output ./report
```

This single call ingests the series, simplifies it with RDP, detects morphological events, evaluates daily protoforms, and writes a self-contained HTML report; the intermediate artifacts can be inspected through the staged commands described below. The `--output` argument accepts either a directory, in which case the report is written to ⟨`directory`⟩`/index.html`, or an explicit `.html` file path. The configuration file declares the linguistic variable, fuzzy quantifiers, the event catalog and any user-defined events, and the protoform templates to evaluate. The minimal configuration below is shipped verbatim with the repository as `example_yaml/config_minimal.yaml`; every remaining

section (`signal`, `io`, `rdp`, `trend`, `contexts`, `lexicon` and `synthesis`) falls back to its defaults:

```
linguistic_variable:
  universe_min: 0
  universe_max: 200
  terms:
    low:    {a: 0,   b: 0,   c: 70,  d: 90}
    medium: {a: 70,  b: 90,  c: 140, d: 160}
    high:   {a: 140, b: 160, c: 200, d: 200}

quantifiers:
  - name: "some"
    membership: {a: 0.2, b: 0.4, c: 0.6, d: 0.8}
  - name: "most"
    membership: {a: 0.6, b: 0.8, c: 1.0, d: 1.0}

events:
  catalogue:
    - label: low_episode
      pattern:
        - level: [low]
      min_duration_minutes: 15
      merge_gap_minutes: 30

    - label: high_episode
      pattern:
        - level: [high]
      min_duration_minutes: 15
      merge_gap_minutes: 30

protoforms:
  - template: "{Q} of the values of the {kind} are {A}"
```

Each entry of the event catalog is a declarative pattern over the linguistic terms of the signal, optionally anchored to previously detected events, rather than a fixed keyword; the catalog of morphological events described in Section 2 is obtained by instantiating these patterns, and new events are added without modifying the source code.

Terms accept four membership function shapes declared via the `type` key:

```
linguistic_variable:
  universe_min: 0
  universe_max: 400
  terms:
    very_low:  {type: trapezoid, a: 0,   b: 0,   c: 54,  d: 70}
    low:       {type: trapezoid, a: 54,  b: 70,  c: 80,  d: 100}
    medium:    {type: gaussian,  mean: 120, sigma: 20}
    high:      {type: triangle,  a: 140, b: 200, c: 250}
    very_high: {type: singleton, value: 300, tolerance: 50}
```

The same pipeline can be invoked stage by stage from the command line (`fuzzydiary simplify`, `fuzzydiary events`, `fuzzydiary describe`, `fuzzydiary narrate`), which is useful for inspecting intermediate artifacts and iterating on the configuration.

The synthesis and generation layers are governed by two optional configuration sections. The `synthesis` section selects the contextual dimension that structures the narrative and the subsethood threshold used when a partition is genuinely nested, and the `lexicon` section supplies the natural-language reading of any label. Both carry defaults, so configurations written for earlier versions remain valid without modification.

3.2. Programmatic use

FuzzyDiary can also be used as a regular Python library, exposing each module as an importable component. The following snippet runs the full pipeline programmatically:

```
from fuzzydiary import (
    load_series, simplify, detect_events,
    describe, render_report,
)

config = "config.yaml"
series = load_series("data/signal.csv", config=config)

simplified = simplify(series, config=config)
events     = detect_events(simplified, config=config,
                           series=series)
daily      = describe(series, events, config=config,
                      simplified=simplified)

render_report(series, events, daily, config=config,
              output="./report/index.html")
```

Users may extend FuzzyDiary by registering custom morphological events through the YAML configuration, by adding new linguistic variables or quantifier shapes to the configuration, or by defining additional protoform templates. The framework does not require modifying the source code for any of these extensions, preserving reproducibility from the configuration alone.

3.3. Output

For a generic univariate TS, the pipeline produces three principal artifacts. Fig. 2 shows the output of the *simplify* and *events* modules: the original signal, its RDP simplification, and the morphological events detected by the defined catalog. Fig. 3 shows a fragment of the self-contained HTML report generated by *render_report*, in which each sentence of the daily narrative is linked to the supporting segment of the original signal.

The day shown consists of real continuous glucose monitoring data from the T1GDUJA dataset [17], collected with the approval of the Ethics Committee of the University of Jaén (reference DIC.21/10.PRY); it contains only timestamps and glucose values, from which the participant cannot be identified. Following an expert-driven approach, the linguistic variable, the event catalog and the protoform templates of the glucose configuration were defined together with clinical experts in diabetes care.

The report opens each day with the synthesized narrative and keeps the detected events and the activated protoforms below it as supporting evidence; hovering over a sentence reveals the statements and events it was built from, and clicking it highlights the corresponding segments on the signal above.

The events behind those highlighted segments are not delimited in the raw samples, but along the piecewise-linear approximation of Fig. 2(b), so their extent, and whether they are detected at all, depend on the RDP tolerance ε. This tolerance is a distance in the day normalized to the unit square, and sets how closely the approximation follows the signal. Table 2 quantifies its effect on the day of Fig. 3: the eight events are preserved up to $\varepsilon = 0.04$, and coarser tolerances progressively remove them until, at $\varepsilon = 0.75$, the day is reduced to a single segment. Over this range, the compression ratio grows from 3.3× to 144× and the root-mean-square reconstruction error increases from 1.4 to 41.5 mg/dL, while the automatic tolerance of the example configuration yields 7.8× with an error of 2.3 mg/dL.

4. Impact

FuzzyDiary contributes to the open-source ecosystem for interpretable TS analysis by addressing a gap that neither tabular fuzzy linguistic summarization libraries nor numerical TS toolkits currently cover: the production of reproducible, morphology-aware natural-language reports from univariate signals, with end-to-end traceability

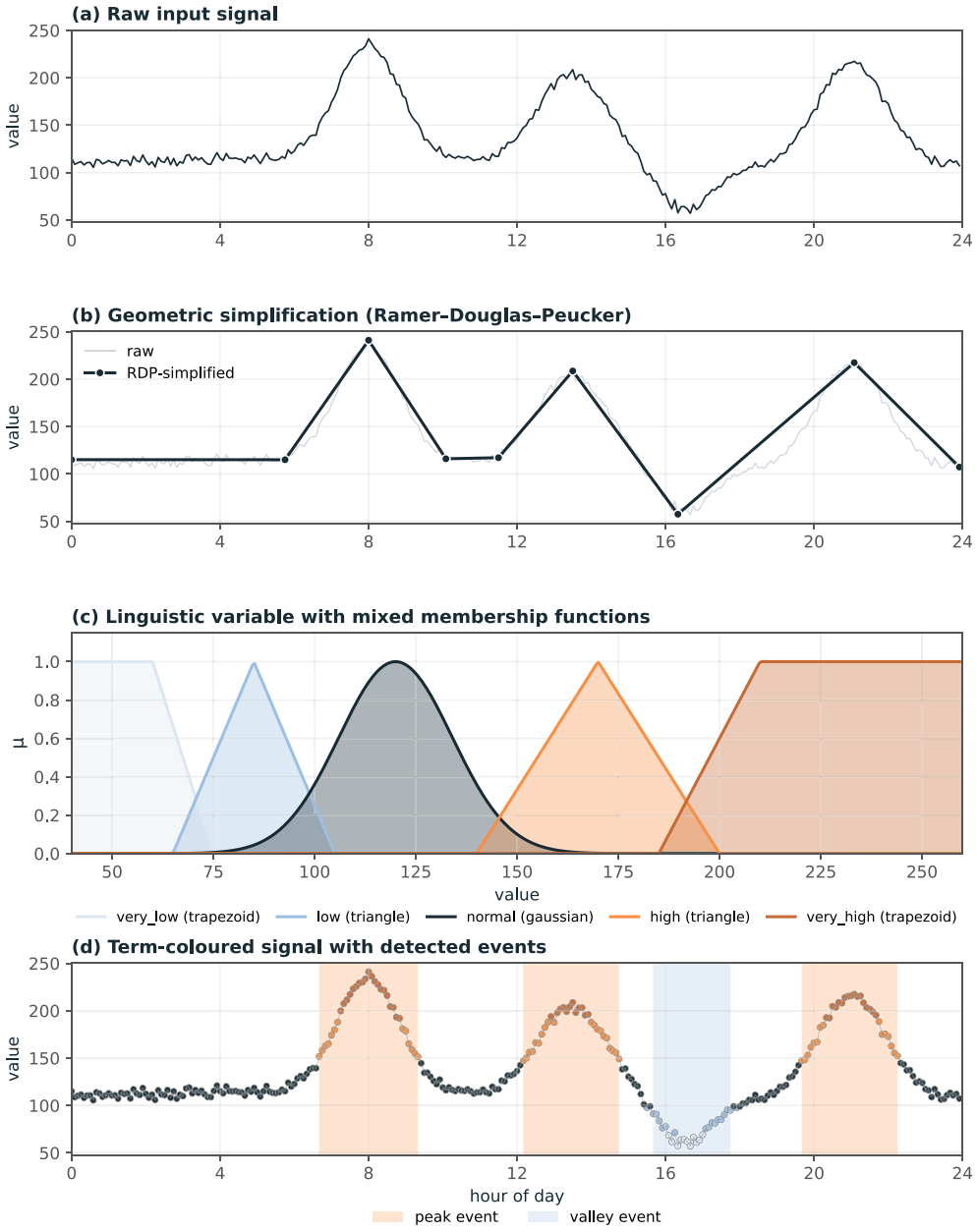


Fig. 2. FuzzyDiary pipeline for a synthetic daily signal: (a) raw input with morning, midday and evening peaks and an afternoon valley; (b) RDP piecewise-linear approximation; (c) linguistic variable with five terms instantiated through trapezoidal, triangular and Gaussian membership functions; (d) term-colored signal with peak and valley events detected by the morphological module shown as shaded bands.

between every textual statement and the underlying observations. We outline below the principal dimensions along which we expect the software to have an impact.

- *Enabling new research questions.* By exposing morphological event detection and fuzzy protoform evaluation as composable modules driven by a single YAML configuration, FuzzyDiary lowers the barrier to empirical work on linguistic descriptions of TS. Researchers can investigate, for instance, how the choice of fuzzy partitions and quantifier shapes affects the truth and focus rankings of linguistic statements, how the granularity of the RDP simplification interacts with the sensitivity of event detection, or how alternative protoform templates change the interpretability of the resulting narratives. Each of these questions previously required ad-hoc implementations; FuzzyDiary makes them tractable through configuration-only experiments.
- *Improving the pursuit of existing research questions.* Within the broader community of fuzzy linguistic summarization and computing with words, FuzzyDiary complements existing tools by occupying a previously uncovered middle layer: structured, morphology-aware description of univariate signals. For practitioners working on clinical reporting, ambient monitoring, industrial telemetry, or any longitudinal sensor stream, FuzzyDiary offers a deterministic, traceable alternative to bespoke reporting scripts, and a faithful intermediate representation that can be consumed by downstream tools, including natural-language post-editors that benefit from receiving verified linguistic facts rather than raw numerical evidence.
- *Changes in daily practice.* The self-contained interactive HTML report —with no server and no runtime beyond a modern browser— is designed to be archived, attached to electronic records, or shared with collaborators in the same way as any other static document. The bidirectional link between every textual statement and the

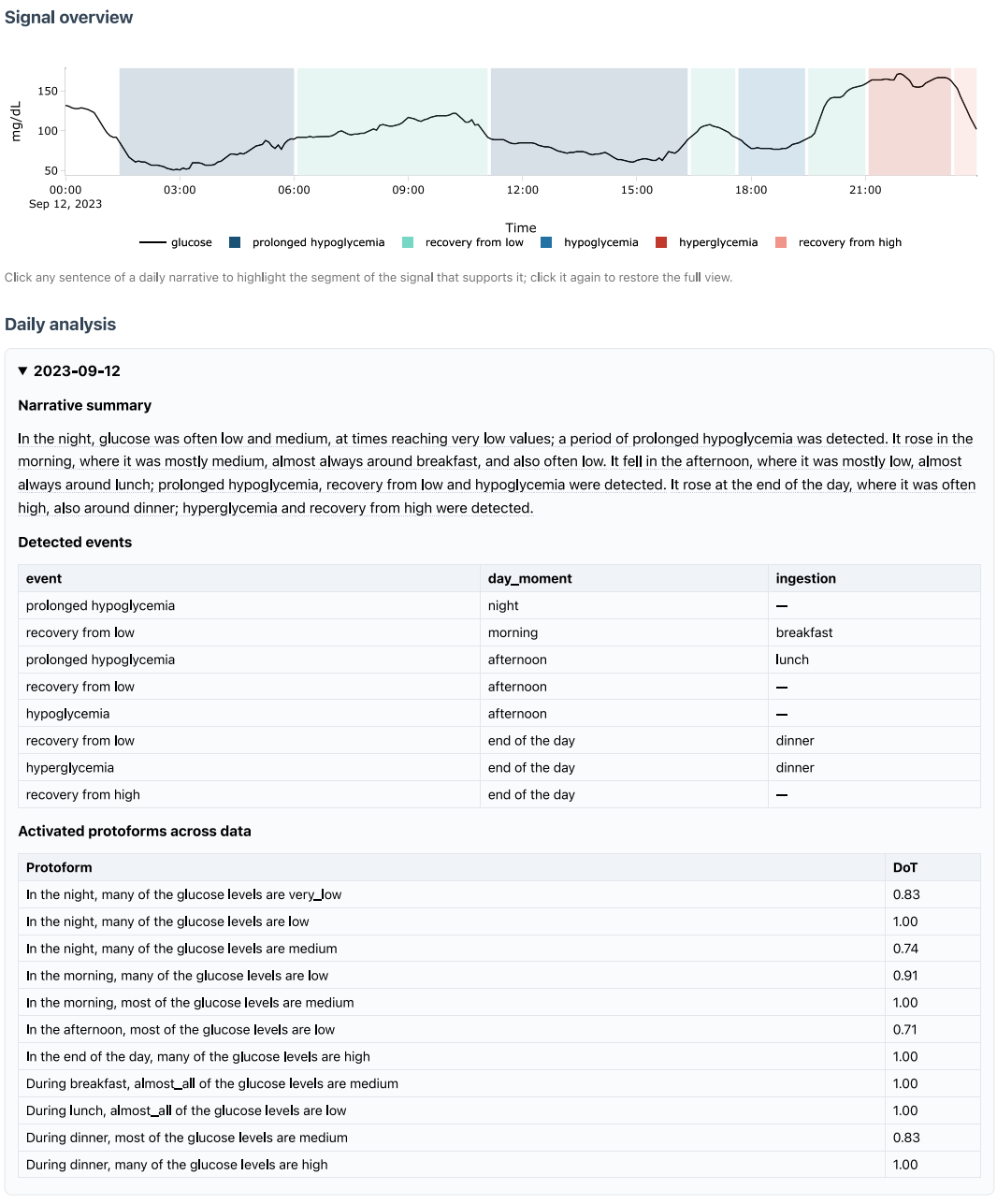


Fig. 3. Fragment of the self-contained HTML report for a single day of continuous glucose monitoring. The signal panel overlays the morphological events detected by the pipeline. The *Detected events* table reports each event with contextual labels (*day_moment*, *ingestion*), and the *Activated protoforms across data* table lists the instantiated protoforms. The *Narrative summary* paragraph above them synthesizes those statements. Clicking any sentence of the narrative highlights the segments on the signal panel that support it.

Table 2
Effect of the RDP tolerance ε on the example series (288 samples). The automatic tolerance of the example configuration is marked as auto.

ε	Vertices	Events	Change
0.005	86	8	–
0.013 (auto)	37	8	–
0.040	14	8	–
0.045	11	8	evening hyperglycemia becomes a peak
0.250	6	6	afternoon hypoglycemia lost
0.400	4	4	second prolonged hypoglycemia lost
0.600	3	2	only the nocturnal episode remains
0.750	2	0	single segment, no events

supporting segments of the original signal supports a traceability requirement that purely textual outputs do not satisfy.

- *Reusability beyond the original domain.* Although the framework is motivated by clinical monitoring and uses the calendar day as its default reporting unit, none of the modules is specific to a clinical context. The event catalog, the linguistic variables, the quantifiers, and the protoform templates are all declared in the configuration; the source code is unchanged when moving to any other univariate signal with a meaningful daily structure.
- *Complementarity with LLM.* FuzzyDiary is intentionally positioned as a deterministic, traceable alternative to generative natural-language reporting based on LLM, where the verbalization step may introduce unverified claims [3]. The narrative layer described

in Section 2.2 shows that a cohesive report does not require a language model when the domain knowledge is already modeled in the configuration, since what has to be resolved is the presentation of facts that protoform evaluation has already established. Rather than competing with LLM-based approaches, the verified linguistic facts produced by FuzzyDiary can serve as a faithful intermediate representation that downstream language models consume as grounded input, opening new horizons in the adaptation of language models for the description of monitoring signals, given their limited performance on the interpretation of numerical data [14,15].

- *Adoption.* FuzzyDiary is released as an open-source Python package under the MIT license, with source code, documentation, and an issue tracker publicly available on GitHub. Distribution through the Python Package Index and the publication of versioned documentation will follow the first stable release. Adoption metrics (stars, forks, citations, contributors) will accrue after publication and will be reported in subsequent updates.

5. Conclusions

This paper introduced FuzzyDiary, an open-source Python framework for protoform-based, morphology-aware linguistic summarization of univariate TS. The framework combines RDP-based geometric simplification, an extensible catalog of morphological events, fuzzy protoform evaluation, and a synthesis and generation layer that resolves the inclusion and scope relationships between the activated protoforms and renders them as a cohesive narrative, into a single configuration-driven pipeline that produces a self-contained interactive HTML report with full traceability through bidirectional links between every textual statement and the supporting segments of the original signal. The design occupies a previously uncovered niche in the open-source ecosystem, complementing tabular fuzzy linguistic summarization libraries and numerical TS toolkits without overlapping with either.

The current version targets univariate signals at a daily granularity; temporal pattern mining over raw signals is outside the scope of this work. Future releases will extend FuzzyDiary along two axes: (i) multivariate signals, supporting linguistic descriptions that span more than one variable through joint protoform templates; and (ii) a richer discourse layer, extending the rhetorical relations and the aggregation rules of the generation stage to the comparison of several reporting periods. Integration with LLMs is deliberately left outside the framework: the synthesized, provenance-annotated facts it produces are well suited as grounded input to an external post-editor, but embedding one would trade the determinism and reproducibility of the output for a variation in phrasing alone.

Moreover, it is highlighted that the current architecture accommodates the first of these extensions without changing its structure. Each variable would be simplified and labeled with its own linguistic variable and event catalog, as the single variable is at present; joint events would extend the composite and anchored events of the catalog, which already build new events from previously detected ones by label and temporal proximity, to components belonging to different variables; and joint protoforms would reuse the qualifier mechanism that currently relies on contextual dimensions such as the moment of the day, with the fuzzy partition of a second variable acting as qualifier, as in *“when heart rate is high, most glucose levels are low”*. Interoperability with the broader time-series ecosystem would further benefit from ingesting the `.ts` format of aeon and sktime; since such files may omit timestamps, which FuzzyDiary needs to delimit reporting periods and evaluate contextual qualifiers, the start time and sampling period would then be declared in the configuration. Similarly, a structured evaluation of the usability of the report and of the informativeness its narratives with clinical professionals is also left for future work.

The framework, its documentation, and the configurations used in this paper are publicly available in the repository listed in the code metadata table.

CRediT authorship contribution statement

Juan Francisco Gaitán-Guerrero: Writing – original draft, Validation, Software, Methodology, Formal analysis, Conceptualization. **José Luis López-Ruiz:** Writing – review & editing, Validation, Supervision, Software, Methodology, Conceptualization. **Carmen Martínez-Cruz:** Supervision, Software, Methodology, Formal analysis, Conceptualization. **Macarena Espinilla:** Resources, Project administration, Methodology, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been partially supported by grant PID2024-156412OB-I00 funded by MICIU/AEI/10.13039/501100011033 and by “ERDF/EU” and by grant M.2 PDC_000756_UJA23 funded by Consejería de Universidad, Investigación e Innovación and by ERDF Andalusia Program 2021–2027.

References

[1] Yager RR. A new approach to the summarization of data. Inf Sci 1982;28(1):69–86.
[2] Marín N, Sánchez D. On generating linguistic descriptions of time series. Fuzzy Sets Syst 2016;285:6–30.
[3] Healey E, Tan ALM, Flint KL, Ruiz JL, Kohane I. A case study on using a large language model to analyze continuous glucose monitoring data. Sci Rep 2025;15(1):1143.
[4] Warner J, Sexauer J, Van den Broeck W, Kinoshita BP, Balinski J, Clauss C, Meggs T, Savio A, Unnikrishnan A, Miretti M, Castelão G, Arruda Pontes F, Uelwer T, Batista F, Germer T, et al. JDWarner/scikit-fuzzy: scikit-fuzzy 0.5.0 (v0.5.0), Zenodo, 2024. https://doi.org/10.5281/zenodo.13372212
[5] Spolaor S, Fuchs C, Cazzaniga P, Kaymak U, Besozzi D, Nobile MS. Simpful: a user-friendly Python library for fuzzy logic. Int J Comput Intell Syst 2020;13(1):1687–98.
[6] Haghrah AA, Ghaemi S, Badamchizadeh MA. PyIT2FLS: an open-source Python framework for flexible and scalable development of type 1 and interval type 2 fuzzy logic models. Softwarex 2025;30:102146.
[7] Peláez-Aguilera MD, Espinilla M, Fernandez Olmo MR, Medina J. Fuzzy linguistic protoforms to summarize heart rate streams of patients with ischemic heart disease. Complexity 2019;2019(1):2694126.
[8] Martinez-Cruz C, Rueda AJ, Popescu M, Keller JM. New linguistic description approach for time series and its application to bed restlessness monitoring for eldercare. IEEE Trans Fuzzy Syst 2021;30(4):1048–59.
[9] Fernandez-Basso C, Díaz-Jimenez D, López JL, Espinilla M. Fuzzy processing applied to improve multimodal sensor data fusion to discover frequent behavioral patterns for smart healthcare. Inf Fusion 2025;123:103307.
[10] Mimbrera FJM, Montoro-Lendínez A, Ruiz JLL, Hermoso MD, Espinilla M. FuzzyIoT-Platform for descriptive analysis of sensor data stream. In: HICSS; 2023. p. 1561–9.
[11] Cascallar-Fuentes A, Gallego-Fernández J, Ramos-Soto A, Saunders-Estevez A, Bugarin-Diz A. Automatic generation of textual descriptions in data-to-text systems using a fuzzy temporal ontology: application in air quality index data series. Appl Soft Comput 2022;119:108612.
[12] Kaczmarek-Majer K, Casalino G, Castellano G, Dominiak M, Hryniewicz O, Kamińska O, Vessio G, Díaz-Rodríguez N. PLENARY: explaining black-box models in natural language through fuzzy linguistic summaries. Inf Sci 2022;614:374–99.
[13] Rodríguez-Rodríguez CR, Zulueta-Veliz Y, Gainza-Reyes D. SUMMARIA—A XAI support methodology by generating composite linguistic summaries of qualitative data. Int J Comput Intell Syst 2025;18(1):171.
[14] Gaitán-Guerrero JF, Martínez-Cruz C, Espinilla M, Díaz-Jiménez D, López JL. A novel fine-tuning and evaluation methodology for large language models on IoT raw data summaries (LLM-RawDMeth): a joint perspective in diabetes care. Comput Methods Programs Biomed 2025;269:108878.
[15] Xu H, Han L, Yang Q, Li M, Srivastava M. Penetrative AI: making LLMs comprehend the physical world. In: Proceedings of the 25th international workshop on mobile computing systems and applications; 2024. p. 1–7.
[16] Saalfeld A. Topologically consistent line simplification with the Douglas-peucker algorithm. Cartogr Geogr Inf Sci 1999;26(1):7–18.
[17] Gaitan-Guerrero JF, Lopez Ruiz JL, Martinez-Cruz C, Espinilla M. T1GDUJA: glucose dataset of a patient with type 1 diabetes mellitus; 2023. https://zenodo.org/records/10713570.