

FuzzyLinguistics: A Python Package for Generating, Simplifying, and Comparing Linguistic Summaries of Data

Brendan Alvey
Dept. of EECS
University of Missouri
Columbia, MO, USA
bja3md@umsystem.edu

James M. Keller
Dept. of EECS
University of Missouri
Columbia, MO, USA
kellerj@missouri.edu

Brendan Young
Dept. of EECS
University of Missouri
Columbia, MO, USA
bmywzx@missouri.edu

Derek T. Anderson
Dept. of EECS
University of Missouri
Columbia, MO, USA
andersondt@missouri.edu

Abstract—Large Language Models (LLMs) can generate polished text, but they are not guaranteed to remain faithful to the underlying numbers. Fuzzy Linguistic Summarization (FLS) converts data into graded protoform statements with explicit fuzzy semantics and quality measures. We present **FuzzyLinguistics**, an open-source Python package for generating, simplifying, and comparing fuzzy linguistic summaries of tabular data. It offers a configuration-driven, Pydantic-validated workflow, vectorized protoform enumeration and scoring, graph-based redundancy reduction with optional compression, and differential summaries for comparing two datasets or model outputs under a shared linguistic vocabulary. Experiments on three minimal examples demonstrate deterministic outputs, strong redundancy reduction, and dataset comparisons, positioning **FuzzyLinguistics** as a practical grounding layer for explainable analysis and LLM-based reporting.

Index Terms—Fuzzy Linguistic Summarization, Fuzzy Logic, Explainable AI, Open-source Software, Computing with Words

I. INTRODUCTION

As AI systems become more capable, the central challenge is increasingly not prediction but communication: turning numerical outputs, model behaviors, and complex datasets into statements that humans can understand and act on. This is especially acute for modern black-box models, where strong performance often comes with limited transparency. Protoform-based linguistic summaries have been used to produce faithful, human-readable explanations in such settings [1] and to support data analysis workflows where practitioners want concise, faithful descriptions rather than large tables of numbers. At the same time, purely generative approaches to data-to-text reporting (including Large Language Models) can produce fluent narratives but may introduce unverified claims when asked to interpret raw numeric evidence.

Fuzzy Linguistic Summarization (FLS) offers a complementary, mathematically grounded alternative. Using fuzzy protoforms and quantifiers, FLS deterministically converts numerical observations into graded natural-language propositions (e.g., “most objects with P are R ”), where each statement is backed by an explicit fuzzy semantics and an associated truth degree. In this respect, FLS can serve as a reliable grounding

layer for downstream explanation systems: it produces verified linguistic facts from data that can be reported directly or supplied as structured inputs to language models. Although protoform-based summarization has been studied since Yager’s early work, there remains no mature, production-oriented Python library focused on end-to-end linguistic summarization of tabular data, including exhaustive protoform generation, ranking, simplification, and comparative summaries.

To address this gap, we introduce **FuzzyLinguistics**, an open-source Python package for generating, simplifying, and comparing fuzzy linguistic summaries of numerical datasets. The package provides a configuration-driven pipeline that (i) defines linguistic variables, predicates, and quantifiers; (ii) enumerates candidate protoform instantiations; (iii) evaluates truth and auxiliary quality measures in a vectorized NumPy/pandas workflow; (iv) reduces redundancy via a graph-based simplification procedure with optional linguistic compaction; and (v) supports differential linguistic summarization to compare two datasets or model outputs under a shared linguistic vocabulary. The design emphasizes deterministic, reproducible execution through structured configuration and exportable artifacts, enabling results to be regenerated exactly from a compact specification.

II. RELATED WORK

Most open-source fuzzy logic libraries are not designed for *data-to-text* linguistic summarization. Python toolkits such as **scikit-fuzzy** [2] and **Simpful** [3] provide membership functions, fuzzy operators, and inference engines, but they do not natively support protoform enumeration, truth-ranked linguistic summaries, or redundancy reduction into compact reports. **FuzzyLogic.jl** [4] offers robust fuzzy primitives in Julia, but its ecosystem differs from the Python-first workflows common in data analysis. The **fuzzy-ml** library [5] targets fuzzy theory for machine learning in PyTorch and includes linguistic summarization among its capabilities, but summarization is not its primary focus and it does not provide a dedicated, configuration-first pipeline for exhaustive protoform generation, graph-based simplification, and differential

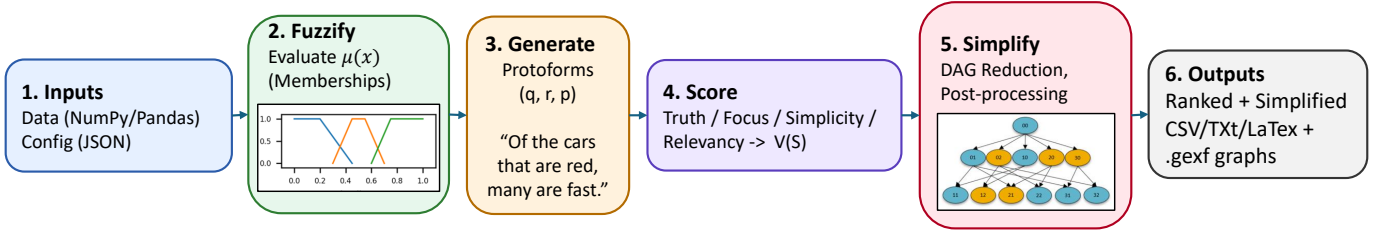


Fig. 1. Overview of the FuzzyLinguistics workflow: configuration-driven fuzzification and protoform generation, vectorized scoring, graph-based redundancy reduction, and export of ranked and simplified summaries, producing deterministic linguistic summaries suitable for reporting or as grounded inputs to LLMs.

summaries. Finally, general NLG/LLM approaches can produce fluent descriptions from data [6], but faithfulness remains a concern in NLG and LLM settings, motivating deterministic linguistic summaries as a grounding layer [7], [8].

Prior work has explored protoform-based linguistic summaries for explainability, redundancy reduction, and comparative analysis; the contribution here is packaging these capabilities into a single, configuration-first Python pipeline aimed at reproducible data-to-text reporting.

III. BACKGROUND AND FORMALIZATION

A. Fuzzy Sets, Linguistic Variables, and Quantifiers

Fuzzy linguistic summarization builds on Zadeh’s notion of a fuzzy set: given a universe of discourse U , a fuzzy set A is characterized by a membership function $\mu_A : U \rightarrow [0, 1]$ that assigns each element a degree of compatibility with the concept [9]. In practice, the semantics of linguistic terms are often encoded by simple parametric shapes (e.g., triangular or trapezoidal), because they are easy to interpret and cheap to evaluate.

A *linguistic variable* L couples a numeric domain with a set of linguistic terms. Following Zadeh [10], a typical representation is $L = (\text{name}, U, \mathcal{T}, \mathcal{M})$, where $\mathcal{T}$ is a term set (e.g., *Low/Medium/High*) and $\mathcal{M}$ maps each term to a fuzzy set on U . In FuzzyLinguistics, each attribute (dimension) is treated as a linguistic variable whose terms are the configured fuzzy predicates.

In protoform-based linguistic summaries, a *summarizer* P is the linguistic description that selects (possibly fuzzy) subsets of the objects (e.g., “clear cloudiness”), while a *qualifier* R describes an additional property of interest (often an outcome or target attribute, e.g., “high outdoor comfort”). In this work, summarizers are built from predicates over designated input dimensions $\mathbf{x}$, and qualifiers are built from predicates over designated output dimensions $\mathbf{y}$; either part may be empty depending on the statement template used (Section III-B).

To capture how much of the data satisfies a linguistic description, quantifier expressions are used, such as *few*, *some*, *most*, and *almost all*. In the proportional setting used by classical linguistic summaries, a quantifier Q is modeled as a fuzzy set on $[0, 1]$, where $\mu_Q(r)$ indicates how well the ratio r matches the intended meaning of the quantifier [11]. This allows the system to convert an empirical (fuzzy) proportion

into a graded truth value that aligns with natural-language quantification.

B. Protoforms for Linguistic Summarization

Let a dataset contain N objects indexed by $i \in \{1, \dots, N\}$, each with observed attributes $\mathbf{x}_i \in \mathbb{R}^{d_{\text{in}}}$ (summarizer dimensions) and optionally $\mathbf{y}_i \in \mathbb{R}^{d_{\text{out}}}$ (qualifier dimensions). A linguistic summary instantiates a *protoform*—a schematic sentence pattern proposed by Yager and later refined in protoform-based summarization frameworks [12]–[14]. In this paper, we use the common two-part form

$$S(q, r, p) : \text{“}Q_q \text{ of the objects with } P_p \text{ are } R_r\text{”}, \quad (1)$$

where P_p is a conjunction of summarizer predicates and R_r is a conjunction of qualifier predicates. Rather than treating R_r as optional within a single sentence form, we treat linguistic summaries as a small *family* of templates: (i) summarizer-only (no R_r), (ii) qualifier-only (no P_p), and (iii) conditional summaries that include *both* P_p and R_r .

For templates where the summarizer part is absent (qualifier-only summaries), we treat the summarizer as *vacuous*: $\mu_P(i) = 1$ for all i . Consequently, its fuzzy support equals the dataset size, i.e., $B_P = \sum_{i=1}^N \mu_P(i) = N$.

For each object i , the degree to which it satisfies a summarizer statement P_p is computed by aggregating the relevant per-attribute predicate memberships. Throughout, $\wedge$ denotes the chosen t -norm (default: Zadeh’s minimum), and $\bigwedge$ denotes iterated application over a set or index. Thus, for the summarizer and qualifier memberships we compute:

$$\mu_{P_p}(i) = \bigwedge_{k \in \mathcal{K}_p} \mu_{P_{k,j(k)}}(x_{i,k}), \quad \mu_{R_r}(i) = \bigwedge_{\ell \in \mathcal{L}_r} \mu_{R_{\ell,j(\ell)}}(y_{i,\ell}), \quad (2)$$

where $\mathcal{K}_p$ (resp. $\mathcal{L}_r$) indexes the dimensions included in the statement and $j(\cdot)$ selects the chosen linguistic term on that dimension.

The proportional semantics reduces each candidate (r, p) to fuzzy cardinalities and a ratio. Define the fuzzy supports

$$A_{r,p} = \sum_{i=1}^N (\mu_{R_r}(i) \wedge \mu_{P_p}(i)), \quad (3)$$

$$B_p = \sum_{i=1}^N \mu_{P_p}(i), \quad B_r = \sum_{i=1}^N \mu_{R_r}(i).$$

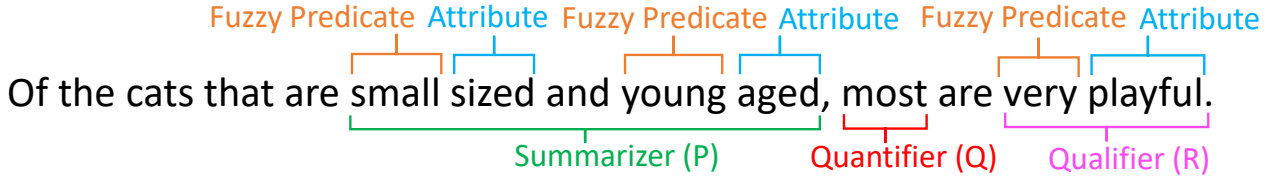


Fig. 2. Linguistic components: fuzzy predicates defined over attributes form summarizers P and qualifiers R , which are evaluated using quantifiers Q .

where $A_{r,p}$ is the fuzzy support for “ R_r among those matching P_p ,” B_p is the fuzzy support of the conditioning part P_p , and B_r is the fuzzy support of the (unconditional) qualifier R_r .

To match the three rendering templates used by the framework, we distinguish conditional and unconditional normalized (fuzzy) support ratios:

$$\begin{aligned} c_{R|P}(r, p) &= \frac{A_{r,p}}{B_p}, & B_p > 0, \\ c_P(p) &= \frac{B_p}{N}, & c_R(r) = \frac{B_r}{N}. \end{aligned} \quad (4)$$

The case where $B_p = 0$ is interpreted as there not being any instances of records matching the given summarizer, which is handled separately.

The protoform truth degree is obtained by quantifier fuzzification:

$$T = \begin{cases} \mu_{Q_q}(c_{R|P}), & \text{“Of the objects with } P_p, Q_q \text{ are } R_r\text{”,} \\ \mu_{Q_q}(c_P), & \text{“}Q_q \text{ of the objects are } P_p\text{”,} \\ \mu_{Q_q}(c_R), & \text{“}Q_q \text{ of the objects are } R_r\text{”.} \end{cases} \quad (5)$$

In particular, “unqualified” summaries (no R_r) are not forced to $c = 1$; instead they use the unconditional coverage ratio $c_P(p) = B_p/N$, so truth reflects how much of the dataset supports P_p .

This protoform view is compatible with broader “computing with words” workflows, where numeric evidence is deterministically lifted into linguistic propositions that can be communicated or consumed downstream [15]. Extensions exist for richer uncertainty representations (e.g., interval-valued and type-2 formulations), but the core proportional protoform remains the most widely used baseline for practical linguistic data summaries [16]. Linguistic labels and quantifiers should respect their intended orderings, and membership-function design can affect protoform truth rankings [17], [18].

C. Quality Measures and Ranking Criteria

Exhaustively instantiating protoforms yields a large candidate set, so linguistic summarization systems typically rank and filter statements using a small family of interpretable quality measures in addition to truth (e.g., support/coverage and conciseness), and interpretability has also been studied explicitly at both the sentence and summary levels [14], [19], [20]. In `FuzzyLinguistics`, each candidate $S(q, r, p)$ is associated with measures that capture (i) correctness, (ii) evidential support, and (iii) structural conciseness.

Truth: Truth is computed by applying the quantifier membership function to the appropriate ratio for the selected template (Eq. 5).

Support / focus: We compute focus as truth weighted by normalized fuzzy support (Eq. 6).

$$F = \begin{cases} T_{q,r,p} \times \frac{B_p}{N}, & \text{“Of the objects with } P_p, Q_q \text{ are } R_r\text{”,} \\ T_{q,p} \times \frac{B_p}{N}, & \text{“}Q_q \text{ of the objects are } P_p\text{”,} \\ T_{q,r}, & \text{“}Q_q \text{ of the objects are } R_r\text{”.} \end{cases} \quad (6)$$

We use ordinary multiplication to down-weight statements that are highly true but supported by negligible mass (i.e., small B_p/N). For qualifier-only templates the conditioning part is vacuous ($B_p = N$), so focus reduces to truth.

Let n_{attr} denote the total number of attributes that may appear in a statement; in our setting $n_{\text{attr}} = d_{\text{in}} + d_{\text{out}}$ (or $n_{\text{attr}} = d_{\text{in}}$ when qualifiers are disabled).

Complexity and simplicity: Let L_p and L_r denote the number of active dimensions used in P_p and R_r . We define the normalized complexity as

$$C(q, r, p) = \frac{L_p + L_r}{n_{\text{attr}}}, \quad (7)$$

and the corresponding simplicity as

$$\text{Sim}(q, r, p) = 1 - C(q, r, p). \quad (8)$$

This biases ranking toward shorter, more readable statements when truth and support are comparable.

IV. FUZZYLINGUISTICS: DESIGN AND IMPLEMENTATION

A. Design Philosophy and Scope

`FuzzyLinguistics` blends fuzzy linguistic theory with everyday Python data workflows. Its goal is to make fuzzy linguistic summarization easy to adopt: it keeps installation lightweight, relies on familiar dependencies, and works with the standard scientific stack (NumPy arrays and pandas tables) [21], [22]. Configuration is explicit but not burdensome. Users can define membership functions for full control, or use automated partitioning to get sensible defaults quickly. Internally, the package prioritizes deterministic, reproducible behavior: given a dataset and fixed fuzzy predicates, it produces the same ranked propositions every time. Efficiency is treated as an engineering constraint, avoiding per-record Python loops where possible for reasonable CPU performance. Output usability is also first-class: instead of returning an exhaustive list of protoforms, a multi-stage simplification

pipeline reduces redundancy while preserving logical content, including structural reduction and merging compatible propositions into compact sentences.

The library targets *data-to-text* summarization of numerical datasets via fuzzy protoforms, producing ranked propositions with quality measures and optional compression. It is not a general fuzzy control toolkit, a full rule-learning framework, or an open-ended NLG system beyond deterministic templates and post-processing. The design emphasizes (i) single-dataset summaries of salient properties, (ii) simplification that reduces cognitive load without discarding key information, and (iii) differential summarization, comparing two datasets (or model outputs) using consistent predicates. This narrower scope enables an accessible interface, transparent implementation, and outputs compact enough for downstream systems, including LLM-based explanation pipelines without relying on the language model to infer numerical facts.

B. Configuration and Reproducibility

A central design goal of `FuzzyLinguistics` is reproducibility from a compact, human-readable configuration. The package therefore exposes configuration-driven entry points and validates inputs with a Pydantic schema [23]. Given the same dataset(s) and fuzzy partitions, the system generates and ranks the same protoform instantiations and produces identical exported artifacts.

Configurations contain four logical components: (i) *datasets* (and, optionally, multiple “models” for differential summaries); (ii) *summarizers* (component P), i.e., fuzzy predicates over input dimensions; (iii) *qualifiers* (component R), i.e., fuzzy predicates over output dimensions (optional); and (iv) *quantifiers* (component Q), i.e., fuzzy quantifiers mapping ratios to linguistic terms. Predicates are encoded as trapezoidal membership functions via four breakpoints, stored as an $m \times 4$ array per dimension.

Users may (a) load a fully specified JSON configuration (best for archival artifacts), (b) construct datasets programmatically while loading membership functions from JSON (fixed vocabulary across datasets), or (c) auto-generate partitions from observed ranges during prototyping and then export/freeze them for deterministic reruns. The library can export ranked summaries (text/CSV/L^AT_EX) and simplification graphs to support repeatable reporting and inspection.

C. Vectorized Inference and Truth Computation

A practical fuzzy linguistic summarizer must evaluate a large number of candidate protoform instantiations over potentially large tabular datasets. `FuzzyLinguistics` therefore treats *vectorization* as the default execution model: all record-wise operations are performed on NumPy arrays, avoiding per-record Python loops. Concretely, each dataset category is represented as dense matrices $\mathbf{X} \in \mathbb{R}^{N \times d_{\text{in}}}$ (summarizer inputs) and $\mathbf{Y} \in \mathbb{R}^{N \times d_{\text{out}}}$ (qualifier outputs), and inference proceeds by precomputing membership-degree tensors that are later reused across all candidate statements. Memberships are evaluated with standard trapezoidal functions using vectorized

NumPy operations (masked linear interpolation), and reused across all candidate statements.

a) Asymptotic scaling: Let N be the number of records, d_{in} summarizer attributes, d_{out} qualifier attributes, and let each attribute have up to m linguistic terms. Base fuzzification is $O(N(d_{\text{in}} + d_{\text{out}})m)$. The number of candidate summarizer patterns is $|\mathcal{P}| = \prod_{k=1}^{d_{\text{in}}} (m_k + 1)$ (Eq. 12), and similarly $|\mathcal{R}|$ for qualifiers. Once fuzzy sufficient statistics are cached, scoring all instantiated statements is $O(|\mathcal{Q}||\mathcal{R}||\mathcal{P}|)$. Redundancy reduction operates on a DAG with $O(|\mathcal{Q}||\mathcal{R}||\mathcal{P}|)$ nodes in the worst case, and graph construction/export can dominate runtime in higher-dimensional settings.

b) From base predicates to statement-level memberships: The library distinguishes *base* memberships (predicate-by-attribute) from *statement* memberships (protoform instantiations). For each dataset category, base summarizer memberships are stored as a tensor $\mu_{\text{base}}^P \in \mathbb{R}^{d_{\text{in}} \times m_P \times N}$, where m_P is the maximum number of fuzzy predicates defined for any summarizer attribute. An analogous tensor $\mu_{\text{base}}^R \in \mathbb{R}^{d_{\text{out}} \times m_R \times N}$ is computed for qualifier attributes.

Candidate summarizer statements P_p (and qualifier statements R_r) are represented by index matrices that select, for each attribute, which fuzzy predicate participates in the statement (or a sentinel indicating “attribute not used”). Given these indices, statement memberships for all N records are assembled by slicing the base tensor and applying a vectorized t -norm across attributes. `FuzzyLinguistics` uses the minimum t -norm, so for record i :

$$\mu_{P_p}(i) = \bigwedge_{k \in \mathcal{K}_p} \mu_{P_{k,j(k)}}(x_{i,k}), \quad \mu_{R_r}(i) = \bigwedge_{\ell \in \mathcal{L}_r} \mu_{R_{\ell,j(\ell)}}(y_{i,\ell}), \quad (9)$$

where $\mathcal{K}_p$ (resp. $\mathcal{L}_r$) is the set of attributes included in the statement. Operational relevance weights are propagated in the same way (by taking the minimum over included predicates). Statement length is tracked as L_p and L_r (defined in Section III-C) for later quality scoring.

c) Truth computation as reusable sufficient statistics: For each candidate protoform instantiation (q, r, p) , the core truth computation follows the standard ratio-based semantics: compute a fuzzy count of records satisfying both qualifier and summarizer, normalize by an appropriate fuzzy cardinality, then apply a quantifier membership function. Denote

$$A_{r,p}(i) = \mu_{R_r}(i) \wedge \mu_{P_p}(i), \quad A_{r,p} = \sum_{i=1}^N A_{r,p}(i), \quad B_p = \sum_{i=1}^N \mu_{P_p}(i). \quad (10)$$

The implementation forms $A_{r,p}$ for *all* (r, p) combinations in a single vectorized reduction (stacking μ_{R_r} and μ_{P_p} , applying `np.min` across the stack, then summing with `np.sum`). This corresponds directly to the fuzzy AND operator $\wedge$ (minimum) used throughout this paper. These aggregated statistics are cached per dataset category and reused when evaluating multiple quantifiers. For qualifier-only ratios, the engine additionally forms $B_r = \sum_{i=1}^N \mu_{R_r}(i)$.

A normalized ratio is then computed according to the statement family (conditional vs. unconditional). For conditional summaries, the engine forms the standard proportional ratio $c_{R|P}(r, p) = A_{r,p}/B_p$ (with NaN when $B_p = 0$). For unconditional summarizer-only or qualifier-only summaries, it uses normalized fuzzy support ratios $c_P(p) = B_p/N$ and $c_R(r) = B_r/N$, respectively. The truth degree is obtained by quantifier fuzzification of the appropriate ratio:

$$T = \mu_{Q_q}(c), \quad (11)$$

where c is the normalized (fuzzy) support ratio.

In addition to truth, the engine computes auxiliary measures used downstream for ranking and simplification. In the current implementation, *focus* is computed as a truth-weighted normalized support term, using B_p/N for conditional summaries and the appropriate unconditional support ratio for summarizer-only or qualifier-only summaries (see Section III-C), complexity is defined as in Eq. 7, with $\text{Sim} = 1 - C$, and *operational relevance* is propagated conservatively as the minimum of the quantifier, qualifier, and summarizer relevance weights. All of these quantities are stored in dense tensors indexed by dataset, quantifier, qualifier-statement, and summarizer-statement, enabling subsequent stages (ranking, redundancy reduction, and differential summaries) to operate purely on array slices rather than recomputing record-level memberships.

D. Statement Generation and Scoring

After configuration (Section IV-B) and vectorized membership evaluation (Section IV-C), `FuzzyLinguistics` constructs a finite set of candidate protoform instantiations and assigns each candidate a small set of interpretable quality measures used for ranking and (later) redundancy reduction.

a) *Enumerating candidate statements*: Let d_{in} be the number of summarizer (input) attributes and d_{out} the number of qualifier (output) attributes. For each summarizer attribute k , a statement may either (i) select exactly one fuzzy predicate among the m_k predicates defined for that attribute, or (ii) exclude the attribute entirely. We encode this with an integer index $p_k \in \{-1, 0, \dots, m_k - 1\}$, where -1 denotes “unused.” The complete summarizer statement set is thus the Cartesian product

$$\mathcal{P} = \bigtimes_{k=1}^{d_{\text{in}}} \{-1, 0, \dots, m_k - 1\}, \quad |\mathcal{P}| = \prod_{k=1}^{d_{\text{in}}} (m_k + 1), \quad (12)$$

and analogously

$$\mathcal{R} = \bigtimes_{\ell=1}^{d_{\text{out}}} \{-1, 0, \dots, m_{\ell}^{(\text{out})} - 1\}, \quad |\mathcal{R}| = \prod_{\ell=1}^{d_{\text{out}}} (m_{\ell}^{(\text{out})} + 1), \quad (13)$$

for qualifier statements, where $m_{\ell}^{(\text{out})}$ is the number of fuzzy predicates defined for qualifier attribute ℓ and $r_{\ell} = -1$ denotes “unused.” Quantifiers are enumerated directly as $q \in \{0, \dots, |\mathcal{Q}| - 1\}$. In the implementation, these products are materialized deterministically, yielding dense integer matrices representing summarizer statement indices and qualifier statement indices that serve as the canonical statement encodings.

b) *Statement rendering*: Each candidate linguistic summary is identified by the triple (q, r, p) and additionally by a *node code* formed by concatenating (i) the quantifier index and (ii) the per-dimension predicate indices for the qualifier and summarizer selections. This code is later used to define parent/child relations in the redundancy graph (a parent contains -1 where children commit to a specific predicate).

For presentation, candidates are rendered into deterministic template-based English. Depending on whether the summarizer/qualifier parts are empty (all indices are -1), the library emits one of the following forms:

$$\text{“}\langle Q \rangle \text{ of the } \langle \text{category} \rangle \text{ are } \langle P \rangle\text{.”} \quad (14)$$

$$\text{“}\langle Q \rangle \text{ of the } \langle \text{category} \rangle \text{ are } \langle R \rangle\text{.”} \quad (15)$$

$$\text{“Of the } \langle \text{category} \rangle \text{ with } \langle P \rangle, \langle Q \rangle \text{ are } \langle R \rangle\text{.”} \quad (16)$$

Any optional compression or merging of statements is performed later, as part of redundancy reduction and linguistic post-processing (Section IV-E).

c) *Quality measures per candidate*: For each dataset category and each valid (q, r, p) combination, the engine computes truth $T \in [0, 1]$ (quantifier applied to the instantiated template ratio), focus $F \in [0, 1]$ (truth-weighted normalized support), and simplicity $\text{Sim} = 1 - C$ (Eq. 7). Operational relevance is a conservative weight $O(q, r, p) = \min(w_Q(q), w_R(r), w_P(p))$.

d) *Aggregated score and ranking*: To rank candidates, `FuzzyLinguistics` forms an aggregated score $V(q, r, p)$ that combines correctness (truth) with coverage and conciseness:

$$G(q, r, p) = 1 + w_F F(q, r, p) + w_C \text{Sim}(q, r, p), \quad (17)$$

$$V(q, r, p) = \frac{O(q, r, p) T(q, r, p) G(q, r, p)}{1 + w_F + w_C}.$$

where w_F and w_C are user-configurable weights. Candidates are sorted in descending V to produce the initial ranked linguistic summary. For transparency and downstream use, the library exports ranked summaries with columns $\{\text{Rank}, V(S), O(S), T(S), F(S), C(S), \text{Sim}(S)\}$ as plain text and CSV, and can optionally convert CSV outputs into IEEE-friendly L^AT_EX tables for direct inclusion in manuscripts.

E. Redundancy Reduction and Linguistic Post-Processing

Enumerating all protoform instantiations yields $|\mathcal{Q}| \times |\mathcal{R}| \times |\mathcal{P}|$ candidate statements (with $|\mathcal{R}|$ reduced when qualifiers are disabled). Even after ranking by $V(S)$, many top-scoring statements are near-duplicates: they differ only by adding one extra predicate (becoming more specific) without materially changing what a reader learns. `FuzzyLinguistics` therefore applies a multi-stage simplification pipeline [24] that removes structural redundancy and optionally compresses the final text.

The library constructs a specialization directed acyclic graph (DAG) whose nodes are statements (plus a single root placeholder) and whose edges encode *specialization*. Each statement is represented by its node code (quantifier index q and per-dimension predicate indices for R and P , using -1

to denote “unused”). A node connects to children formed by selecting exactly one unused component (−1) and replacing it with a concrete predicate index; intuitively, children are strictly more specific than their parent because they add one additional linguistic constraint. The DAG (and the report decisions) can be exported so users can inspect what was pruned and why.

a) Stage 1: thresholding: Here we write $S \equiv (q, r, p)$ to denote a specific instantiated statement triple. Nodes are first filtered by a user threshold on the aggregated statement value $V(S)$ (the same score used for ranking in Section IV-D). For stability across datasets and clearer visualization, $V(S)$ is linearly rescaled to $[0, 1]$ within each dataset category before applying the threshold τ .

b) Stage 2: structural pruning: Given the active nodes, the simplifier selects which statements should be reported. By default, reporting favors succinctness: if a parent statement is active, its more specific descendants can be suppressed; similarly, if a node’s immediate children are all active, the parent can be reported to avoid listing a family of near-duplicates.

c) Stage 3: grouping and compaction: Remaining reportable statements that differ only in a single component are grouped (e.g., same quantifier and qualifier but different summarizers, or vice versa) and rendered as a single compound statement using logical “or” forms; when appropriate, the renderer can use shorter “except ...” phrasing instead of enumerating many alternatives.

d) Stage 4: textual compression (optional): Finally, an optional post-processing step merges compatible sentences that share the same subject/quantifier scaffolding, combining only the differing predicate phrases. This step does not recompute fuzzy semantics; it rewrites the already-selected statements into more compact English while preserving the meaning implied by the retained protoform instances.

e) Outputs and reproducibility: The pipeline exports: (i) the full statement hierarchy graph, (ii) intermediate pruning-stage graphs, and (iii) the final simplified textual summary. These artifacts make the simplification process auditable and easy to reproduce: given the same configuration, scoring weights, and threshold, the same nodes are selected and the same final compressed text is produced.

F. Differential Linguistic Summarization

Beyond summarizing a single dataset, our package supports *differential* (comparative) linguistic summarization between two datasets or two model outputs under a shared linguistic vocabulary [25]. This is useful whenever a practitioner wants statements of the form “many more” / “many less” / “about the same” describing how two populations differ with respect to the same fuzzy predicates.

a) Setup and assumptions: Differential summarization operates on pairs of dataset categories that share the same category name but have different model names (e.g., model M_1 vs. model M_2). The same summarizers, qualifiers, and quantifiers are applied to both datasets so that comparisons

are linguistically aligned. The current implementation supports exactly two models at a time.

b) Comparative truth semantics: For each candidate statement (q, r, p) , the engine computes fuzzy “support” statistics for each model $M \in \{1, 2\}$:

$$\begin{aligned} A_M(r, p) &= \sum_{i=1}^{N_M} (\mu_{R_r}^{(M)}(i) \wedge \mu_{P_p}^{(M)}(i)), \\ B_M(p) &= \sum_{i=1}^{N_M} \mu_{P_p}^{(M)}(i). \end{aligned} \quad (18)$$

where A_M is the fuzzy cardinality of records satisfying both qualifier and summarizer, and B_M is the fuzzy support of the summarizer condition. The differential signal is then the signed difference

$$a(r, p) = A_1(r, p) - A_2(r, p). \quad (19)$$

To obtain a bounded comparison ratio, the denominator is chosen as a max-support normalization:

$$\begin{aligned} b(p) &= \max(B_1(p), B_2(p)), \\ c(r, p) &= \begin{cases} \frac{a(r, p)}{b(p)}, & b(p) > 0, \\ \text{NaN}, & A_1(r, p) = A_2(r, p) = 0, \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \quad (20)$$

The final case is a numerical safeguard to keep $c(r, p)$ well-defined when $b(p) = 0$ in degenerate support regimes.

For unconditioned statements (the “empty” summarizer case), the implementation uses a record-count normalization to keep c well-scaled even when no P condition is present. A configurable language mode parameter additionally controls whether the normalization is computed per-category or aggregated across categories (useful when comparing multiple categories jointly). Unlike the single-dataset proportional case (where $c \in [0, 1]$), the comparative ratio is signed and lies in $[-1, 1]$, so comparative quantifiers are defined over $[-1, 1]$.

Finally, comparative truth is computed by fuzzifying this signed ratio with a set of *comparative quantifiers* defined over $[-1, 1]$ (e.g., *Many less*, *Some less*, *The Same Amount*, *Some more*, *Many more*):

$$T_{q, r, p} = \mu_{Q_q}(c(r, p)). \quad (21)$$

A special “None” quantifier is supported to capture cases where $c(r, p)$ is undefined because both models have zero support for the statement (i.e., no evidence in either dataset).

c) Focus and ranking under comparison: To avoid emphasizing differences supported by negligible mass, we define comparative focus as

$$F_{q, r, p} = T_{q, r, p} \times \frac{B_1(p) + B_2(p)}{N_1 + N_2}. \quad (22)$$

All other quality measures and aggregation into $V(S)$ (operational relevance, complexity/simplicity, and ranking) are computed exactly as in the single-dataset case (Section IV-D).

d) *Comparative phrasing*: Differential statements use the same deterministic templates as single-model summaries, augmented with a model-order suffix (e.g., “from M_1 than from M_2 ”) so that directional quantifiers such as “more” / “less” are unambiguous. For neutral quantifiers (e.g., “the same amount”), the phrasing is adjusted to indicate symmetry (“from M_1 and from M_2 ”). This makes the output directly readable while preserving a one-to-one correspondence between text and the underlying computed fuzzy comparison.

V. EXPERIMENTAL EVALUATION

A. Datasets and Experimental Setup

We evaluate `FuzzyLinguistics` using three compact, human-relatable examples designed to illustrate the library’s core workflow: (i) single-dataset summarization, (ii) scaling to a higher-dimensional feature space with mixed predicates, and (iii) differential (two-population) summarization. All experiments use trapezoidal membership functions for predicates and quantifiers and the same conjunction operator as defined in Section III-B, and the same ranking weights $w_F=0.35$ and $w_C=0.15$. Unless stated otherwise, we simplify the ranked list using the redundancy-reduction pipeline with a reporting threshold $\tau=0.65$ applied to the per-category scaled aggregated statement value $V(S)$. All configuration files and runnable scripts used to reproduce Experiments 1–3 are provided in the `FuzzyLinguistics` repository [26].

a) *Experiment 1 (Weather $\rightarrow$ outdoor comfort)*: A small table of $N=24$ “days” with two summarizers (cloudiness, rain intensity) and a qualifier (outdoor comfort). This example demonstrates conditional summaries (“Of the days with . . . , most are . . .”) and highlights how the simplification pipeline compresses a combinatorial candidate space into a short report.

b) *Experiment 2 (Music tracks: mood/energy + genre affinities $\rightarrow$ enjoyment)*: A playlist-like dataset of $N=30$ tracks with six summarizers: two continuous descriptors (energy, mood) and four soft genre affinities (pop/rock/hiphop/classical), plus one qualifier (enjoyment). This example stresses higher-dimensional statement generation (many possible protoform instantiations) while keeping the semantics intuitive.

c) *Experiment 3 (Cats vs. Dogs: traits $\rightarrow$ ease of care)*: Two populations (Cats and Dogs), each with $N=25$ items, summarized with two traits (activity level, sociability) and one qualifier (ease of care). This demonstrates differential linguistic summarization with comparative quantifiers (e.g., “some more”, “about the same”).

B. Results: Fidelity and Conciseness

Across all experiments, the top-ranked summaries are high-truth statements derived deterministically from the configured fuzzy partitions and quantifiers. The simplification pipeline substantially reduces redundancy while preserving the underlying protoform semantics.

TABLE I
SUMMARY OF THE THREE MINIMAL EXPERIMENTS. “PROTOF.” COUNTS FULL (q, τ, p) INSTANTIATIONS; “NODES” INCLUDES ROOT/PARTIAL NODES FOR REDUNDANCY REDUCTION. $> \tau$ INDICATES NODES ABOVE THE REPORTING THRESHOLD (APPLIED TO PER-CATEGORY SCALED AGGREGATED VALUE) BEFORE STRUCTURAL PRUNING.

Exp.	N	d_{in}	Protof.	Nodes	$> \tau$	Final
Weather	24	2	64	257	33	9
Music	30	6	16.4k	65.5k	515	15
Cats/Dogs	25/25	2	64	385	38	5

a) *Experiment 1 (Weather)*: Starting from a 257-node redundancy graph (64 concrete protoform instantiations plus partial nodes), thresholding yields 33 candidates above $\tau=0.65$ and 9 final simplified statements. Example simplified outputs include: (i) “Of the weather days with clear cloudiness, most are high outdoor comfort.” (ii) “Of the weather days with heavy rain intensity, almost all are low outdoor comfort.” and (iii) “Of the weather days with clear cloudiness and no rain, almost all are high outdoor comfort.”

b) *Experiment 2 (Music)*: This experiment demonstrates scaling behavior: the system evaluates 16,384 concrete protoform instantiations (and a 65,537-node redundancy graph), then simplifies to 15 reportable statements at $\tau=0.65$. Representative outputs include: (i) “Of the playlist tracks with neutral mood almost all are medium enjoyment.” (ii) “Of the playlist tracks with strong pop affinity and low rock affinity some are high enjoyment.” and (iii) “Of the playlist tracks with strong hiphop affinity almost all are medium enjoyment.” The largest runtime costs arise from graph construction/export and simplification on the larger node set; this highlights where engineering optimizations (or stronger early filters) matter most at higher dimensionalities.

c) *Experiment 3 (Cats vs Dogs)*: Differential summarization produces 5 final comparative statements after thresholding (38 above τ) and structural pruning. Example outputs include: (i) “High ease of care is somewhat more prevalent among cats than among dogs.” (ii) “Moderate activity level is somewhat more prevalent among cats than among dogs.” and (iii) “Among pets with mixed sociability, medium ease of care is somewhat more prevalent among cats than among dogs.”

Across all experiments, simplification reduced the candidate set by orders of magnitude (e.g., 65.5k nodes to 15 statements in Music).

Complexity and scaling. For fixed fuzzy partitions and quantifier sets, the dominant cost in statement scoring is evaluating μ_{R_r} and μ_{P_p} and reducing across samples, which scales as $O(N|\mathcal{R}||\mathcal{P}|)$ for exhaustive enumeration. The redundancy graph introduces additional overhead that grows with the number of nodes and edges; in practice, graph construction and pruning dominate only after the candidate space becomes large (e.g., Experiment 2). This motivates (i) stronger early filtering (e.g., thresholding during enumeration) and (ii) caching or batching strategies for higher-dimensional settings.

VI. CONCLUSION AND FUTURE WORK

This paper introduced `FuzzyLinguistics`, an open-source Python framework for generating, simplifying, and comparing fuzzy linguistic summaries of numerical data. The library implements protoform-based linguistic summarization with vectorized truth computation, exports transparent quality measures, and applies a graph-based redundancy-reduction pipeline that compresses large candidate spaces into concise, human-readable reports. Across three minimal, relatable experiments, we showed that the system produces deterministic summaries and that simplification is essential for turning exhaustive protoform enumeration into practical outputs.

Future work will focus on tighter integration with Large Language Models (LLMs) in two complementary directions: using LLMs as a controlled post-processor to refine phrasing while preserving computed semantics, and leveraging LLMs to assist in automatic membership-function generation (e.g., proposing initial partitions from domain text or data descriptions that can be validated and frozen for reproducibility). In addition to trapezoidal membership functions, we would like to also support other common membership function definitions such as Gaussian. Another planned addition is user-defined protoform templates for customizing statement generation. We also plan to generalize differential summarization beyond two models/datasets to support multi-population comparisons and aggregated comparative narratives. Finally, we will evaluate `FuzzyLinguistics` as a tool for LLM-based reasoning: measuring whether supplying FLS outputs (truth-ranked and simplified) improves factuality, robustness, and task performance when LLMs analyze large or domain-specific datasets.

REFERENCES

- [1] B. J. Alvey, D. T. Anderson, and J. M. Keller, “Explainable ai via linguistic summarization of black box computer vision models,” in *2023 IEEE Conference on Artificial Intelligence (CAI)*. IEEE, 2023.
- [2] J. Warner, J. Sexauer, W. V. den Broeck, B. P. Kinoshita, J. Balinski, scikit fuzzy, C. Clauss, twmeggs, alexsavio, A. Unnikrishnan, M. Miretti, G. Castelh o, F. A. Pontes, T. Uelwer, phme283, pd2f, laurazh, F. Batista, moetayuko, alexbuy, W. Song, T. Ni, T. G. Badger, R. A. M. P rez, N. Muoh, J. F. Power, H. Mishra, G. O. Trullols, A. H rteborn, and T. Germer, “Jdwarner/scikit-fuzzy: Scikit-fuzzy 0.5.0,” Aug. 2024. [Online]. Available: 10.5281/zenodo.13372212
- [3] S. Spolaor, C. Fuchs, P. Cazzaniga, U. Kaymak, D. Besozzi, and M. S. Nobile, “Simpful: A user-friendly python library for fuzzy logic,” *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1687–1698, 2020. [Online]. Available: 10.2991/ijcis.d.201012.002
- [4] L. Ferranti and J. Boutellier, “Fuzzylogic.jl: A flexible library for efficient and productive fuzzy inference,” in *2023 IEEE International Conference on Fuzzy Systems (FUZZ)*, 2023, pp. 1–5.
- [5] J. Hostetter, “fuzzy-ml: Fuzzy theory for machine learning in pytorch,” GitHub repository, 2025, accessed: 2026-01-27. [Online]. Available: <https://github.com/johnHostetter/fuzzy-ml>
- [6] A. Gatt and E. Krahmer, “Survey of the state of the art in natural language generation: Core tasks, applications and evaluation,” *Journal of Artificial Intelligence Research*, vol. 61, pp. 65–170, 2018. [Online]. Available: <https://www.jair.org/index.php/jair/article/view/11173>
- [7] W. Li, W. Wu, M. Chen, J. Liu, X. Xiao, and H. Wu, “Faithfulness in natural language generation: A systematic survey of analysis, evaluation and optimization methods,” arXiv preprint arXiv:2203.05227, 2022. [Online]. Available: <https://arxiv.org/abs/2203.05227>
- [8] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, and T. Liu, “A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions,” *ACM Transactions on Information Systems*, vol. 43, no. 2, pp. 1–55, Jan. 2025. [Online]. Available: 10.1145/3703155
- [9] L. A. Zadeh, “Fuzzy sets,” *Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.
- [10] L. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning—i,” *Information Sciences*, vol. 8, no. 3, pp. 199–249, 1975. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0020025575900365>
- [11] L. A. Zadeh, “A computational approach to fuzzy quantifiers in natural languages,” *Computers & Mathematics with Applications*, vol. 9, no. 1, pp. 149–184, 1983.
- [12] R. R. Yager, “A new approach to the summarization of data,” *Information Sciences*, vol. 28, no. 1, pp. 69–86, 1982. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0020025582900330>
- [13] R. R. Yager, K. M. Ford, and A. J. Ca as, “An approach to the linguistic summarization of data,” in *Uncertainty in Knowledge Bases*, ser. Lecture Notes in Computer Science. Springer, 1991, vol. 521, pp. 456–468.
- [14] J. Kacprzyk and S. Zadrozny, “Linguistic database summaries and their protoforms: Towards natural language based knowledge discovery tools,” *Information Sciences*, vol. 173, no. 4, pp. 281–316, 2005.
- [15] L. A. Zadeh, “Fuzzy logic = computing with words,” *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, pp. 103–111, 1996.
- [16] A. P. Niewiadomski, J. Ochelska, and P. S. Szczepaniak, “Interval-valued linguistic summaries of databases,” *Control and Cybernetics*, vol. 35, pp. 415–443, 2006. [Online]. Available: <https://api.semanticscholar.org/CorpusID:6501613>
- [17] A. Jain and J. M. Keller, “On the computation of semantically ordered truth values of linguistic protoform summaries,” in *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015, pp. 1–8.
- [18] A. Jain, T. Jiang, and J. M. Keller, “Impact of the shape of membership functions on the truth values of linguistic protoform summaries,” in *Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer, 2016, pp. 204–213.
- [19] J. Kacprzyk and R. R. Yager, “Linguistic summaries of data using fuzzy logic,” *International Journal of General Systems*, vol. 30, pp. 133–154, 2001.
- [20] M.-J. Lesot, G. Moyse, and B. Bouchon-Meunier, “Interpretability of fuzzy linguistic summaries,” *Fuzzy Sets and Systems*, vol. 292, pp. 307–317, 2016, special Issue in Honor of Francesc Esteva on the Occasion of his 70th Birthday. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0165011414004667>
- [21] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R o, M. Wiebe, P. Peterson, P. G rard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, “Array programming with numpy,” *Nature*, vol. 585, no. 7825, pp. 357–362, 2020. [Online]. Available: 10.1038/s41586-020-2649-2
- [22] W. McKinney, “Data structures for statistical computing in python,” in *SciPy*, 2010. [Online]. Available: <https://api.semanticscholar.org/CorpusID:13156234>
- [23] S. Colvin, E. Jolibois, H. Ramezani, A. Garcia Badaracco, T. Dorsey, D. Montague, S. Matveenko, M. Trylesinski, S. Runkle, D. Hewitt, A. Hall, and V. Plot, “Pydantic,” 2025. [Online]. Available: 10.5281/zenodo.15174950
- [24] B. Alvey, D. T. Anderson, and J. M. Keller, “Minimizing protoform redundancy to enhance linguistic summaries in object detection,” in *2025 IEEE International Conference on Fuzzy Systems (FUZZ)*. IEEE, 2025.
- [25] B. J. Alvey, D. T. Anderson, and J. M. Keller, “Linguistic comparisons of black box models,” in *2024 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2024.
- [26] B. Alvey, “Fuzzylinguistics,” <https://github.com/lyvelyte/fuzzylinguistics>, 2025, gitHub repository, version v0.2.1.