MASTER IN COMPUTER SCIENCE

Project Report

Explaining Public Budgets with Fuzzy Driven Summaries

Seminar in Urban Computing

submitted by

Bohdan Potuzhnyi

Vlada Svirsh

Semester A2025

UNIVERSITÄT BERN

UNIVERSITÉ DE NEUCHÂTEL

UNI FR
UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

Contents

1 Introduction

1.1 Background

Swiss municipalities such as Zürich publish detailed budget tables through open data portals, yet this openness does not automatically translate into understanding. In practice, budget PDFs and API payloads often assume technical and accounting backgrounds, which limits who can interpret the figures and participate in public-finance discussions. Prior to work on transparency similarly argues that publishing data is not enough: citizens benefit from representations that connect spending to meaningful civic questions and provide context for change over time (Bertot et al., 2010; Cucciniello et al., 2017).

> "Numbers have an important story to tell. They rely on you to give them a clear and convincing voice." — Stephen Few.

1.2 Problem

Preliminary discussions about Zürich's open budget portal indicated four recurring frictions.

1. The material is perceived as overly technical: readers encounter tables and codes rather than explanations.
2. Accessibility remains low because answering even simple questions requires API literacy or data-processing skills.
3. These barriers reduce engagement, since many residents feel disconnected from financial decisions.
4. The absence of interpretive layers makes it difficult to identify what matters most in public spending, especially when priorities shift gradually across years.

1.3 Goals

The project explores whether summaries driven by fuzzy logic can serve as a lightweight translation layer from numeric budgets to citizen-facing statements. Concretely, we aim to:

1. Implement systems that map departmental shares and trends to controlled linguistic labels.
2. Provide deterministic and reproducible responses that remain traceable to the original data.
3. Evaluate whether the resulting messages are interpretable and useful for typical citizen questions.

1.4 Presented values and report structure

This report presents a reproducible prototype for explaining Zürich's public budget in plain language using fuzzy-driven summaries. The main contribution is a Python implementation that loads budget data from the official API or from cached CSV exports and generates traceable, structured summaries (see Section 5; code in the repository (Potuzhnyi & Svirsh, n.d.)). In addition, the report documents the fuzzy labeling and robust trend estimation approach used to translate numeric shares into terms such as "low/high" and "rising/stable/falling" (Section 5). It reports two small evaluation rounds that assess interpretability, usability, and repeatability of the outputs (Section 6).

2 Related Work and Theory

This section summarizes the ideas that shaped our approach: why open budget data is often still hard to understand, and how fuzzy linguistic summaries can help. We first look at work on transparency and citizen-friendly budget communication, then introduce "computing with words" and fuzzy summarization, and finally explain why we use robust trend estimation for short municipal time series. These concepts directly inform the prototype design (Section 5) and the evaluation criteria (Section 6).

2.1 Budget transparency & citizen-centered public finance communication

Research on transparency repeatedly shows that publishing more data is not the same as creating understanding. If data is released in a technical or overwhelming form, it can create "transparency illusions", overload readers, or even reduce trust (Bannister & Connolly, 2011; Grimmelikhuijsen & Meijer, 2014; Heald, 2006). In the open budget domain, this leads to the idea of translation layers: interfaces that convert administrative structures into the kinds of questions citizens actually ask in consultations (Kim et al., 2016; Tygel et al., 2015). This is why our summaries focus on familiar topics (e.g., education, transport) and on changes over time, instead of presenting raw ledger tables.

2.2 Computing with words and perceptions

Zadeh's "computing with words" argues that people often reason with qualitative terms like "high", "low", or "stable", not with exact numbers (Zadeh, 1996, 2002). Later work makes a similar point: systems should adapt to human language and perception rather than forcing non-experts to adopt technical parameters and accounting vocabulary (Mendel et al., 2010). We apply this idea by mapping numeric budget shares and trends to the kinds of phrases that naturally appear in civic discussions (e.g., "healthcare is rising" or "education is high but stable").

2.3 Fuzzy linguistic summarization

Fuzzy linguistic summarization is a way to express statements like "spending is high" while still being explicit about uncertainty, by attaching a membership degree (μ) and confidence cues (Kacprzyk & Zadrożny, 2005; Yager, 1982). For representing labels in a controlled and interpretable way, the 2-tuple model is a common approach (Herrera & Martínez, 2000). In our prototype, we use trapezoidal membership functions and calibrate them from Zürich's observed data: percentiles (10/30/50/70/90) for spending shares, and the median absolute deviation (MAD) for trend slopes. This keeps labels anchored in what is typical for Zürich and allows us to say when a label is only a weak match (e.g., low μ for very small departments).

2.4 Trend estimation in time series

A simple least-squares slope can change a lot with outliers, which matters for municipal budgets where single-year events can distort trends. The Theil–Sen estimator is more robust because it uses the median of pairwise slopes (Sen, 1968; Theil, 1950), which works well for short windows like 5–6 years. In the prototype, we report trends both as percentage points per year and as relative change compared to the mean share, so that trends are comparable across departments with very different baseline sizes.

2.5 Summary of gaps

Overall, prior work covers:

1. The limits of "raw transparency".
2. Citizen-oriented budget interfaces.
3. Fuzzy linguistic summaries.
4. Robust trend estimation.

However, we did not find publicly documented Zürich prototypes that combine the official RPK-API budget feeds, Theil–Sen trend estimation, and fuzzy linguistic summaries in a scriptable Q&A-style workflow (*RPK-API Documentation*, 2024). This project addresses this gap by implementing such an artifact and evaluating it.

3 Research Questions and Method

3.1 Research questions

The project focuses on whether Zürich's open budget data can be explained in plain language without losing traceability to the underlying numbers. The following research questions guided the work:

- RQ1: Can fuzzy-driven summaries make Zürich's open budget data easier to understand for non-experts?
- RQ2: How do we keep the summaries stable, repeatable, and hard to misread (especially across different time windows)?

Interpretability is evaluated pragmatically. In this project, an output is considered "interpretable" if users can correctly explain what the statement refers to (topic + timeframe + direction of change) and if they rate the answer as clear enough to be useful in a citizen-facing setting.

3.2 Methodological approach

To structure the project work, a Design Science Research (DSR) approach was applied (Peffers et al., 2007). The project was organized in iterative steps, moving from problem framing to a working artifact and a small-scale evaluation.

1. Problem identification. Initial observations and feedback from the seminar context indicated that budget portals are technically rich but hard to use for non-experts. In particular, citizens often ask questions in everyday terms ("education", "more lately", "what changed") that do not match accounting structures.
2. Objectives of a solution. The goal was to create a prototype that returns short, understandable statements while remaining verifiable. The artifact therefore had to:
   - Map citizen phrasing to a small set of supported query parameters.

- Summarize budget shares and trends in linguistic terms.
- Remain stable enough that minor changes in the time window do not produce contradictory narratives without warning.

3. Design and development. A modular pipeline was implemented in Python with four stages: data retrieval, aggregation, linguistic summarization, and response generation. To keep results reproducible during demonstrations, the system supports cached CSV exports in addition to live API access. A deterministic NLU layer was added to map free-form questions to three query slots: timeline, field, and requested level of detail.
4. Demonstration. The artifact is demonstrated as a question–answer flow comparable to the motivating citizen dialogue. Users can ask about a department (e.g., education) or the whole city budget, optionally restricting the time window (e.g., “since 2021”), and receive a structured JSON response that includes a short natural-language statement plus the supporting evidence.
5. Evaluation. Evaluation was aligned with the two research questions:
   - For RQ1 (Interpretability), six peers with STEM backgrounds executed two scripted citizen tasks (a topic-specific question and a citywide follow-up) while saying a result aloud. We recorded whether they retrieved the correct summary, how many reformulations were needed, and Likert ratings for clarity, explainability, and usability.
   - For RQ2 (Robustness & reproducibility), we applied determinism-by-design checks: each scripted request was replayed twice on the cached dataset to verify identical JSON responses. We also inspected rolling four-year windows (and small shifts of the start year) to confirm that label changes occur only when the underlying numbers shift meaningfully, not due to small numerical noise. Finally, we ran an automated regression of 70 requests (`python_code/nlu/nlu_test_set.json`) to confirm that the deterministic parser maps everyday questions to the intended slots consistently.
6. Communication. The outcomes are documented in the final report and supported by a runnable CLI/JSON demo that mirrors the use-case dialogue.

3.3 Ethical and societal considerations

A citizen-facing transparency tool can easily create a false impression of certainty. For this reason, the artifact is designed to show the basis of each statement rather than only a “nice sentence”. Each response includes the selected topic, the time window, and the numeric trend unit. In addition, fuzzy membership values are reported to indicate how strongly a department fits a label such as “high” or “rising”.

The system also avoids causal claims. Budget shifts can result from political decisions, accounting changes, exceptional events, or administrative reclassifications. Since such causes are not contained in the dataset used here, the artifact does not generate explanations of “why” spending changed. Instead, it limits itself to describing observed patterns and pointing back to the original data source for verification. Cached datasets are stored in a simple format so that results can be reproduced and inspected.

4 Use Case, Data, and Requirements

4.1 Use-case scenarios

We designed the prototype around the kinds of questions people actually ask when they open a public budget portal. In practice, these questions come in many patterns, yet we define and focus on the next three:

1. Citizen quick question (topic-focused). A user wants a simple explanation for one area, like education or transport. The system should return a short statement that includes the current spending level and the direction of change over the selected years (e.g., "Since 2019, education spending has stayed high and mostly stable…").
2. Citizen follow-up (citywide overview). After the topic answer, users often ask a broader question like "So what changed the most overall?". Here the system should keep the same timeline and return the largest increases and decreases across departments, so people can compare shifts in priorities.
3. Fact-check / shareable answer (repeatable output). For students, journalists, or anyone who wants to verify results, the same request should always produce the same answer and expose the evidence behind it. The system therefore returns a deterministic JSON payload that includes the selected topic, the time window, and the numeric trend information.

4.2 Data sources

The prototype is built on Zürich's public RPK budget API. We use two endpoint families:

- `departemente` for department metadata (names and identifiers),
- `sachkonto2stellig` for aggregated budget amounts by two-digit account categories.

To focus on decisions that are politically "final" in the data, we filter to approved budgets (`betragsTyp = GEMEINDERAT_BESCHLUSS`). The script queries departments across years with light throttling to avoid stressing the public endpoint and uses the public API key described in `python_code/README.md`.

- Scope of the dataset. We use the years 2019–2024 to include both pre- and post-pandemic budget adjustments. Amounts are aggregated to department totals per year, and we derive each department's share of total spending, so results are comparable across years.
- Reproducibility. Because public endpoints can change or fail or connection can be lost (either from client or server side), the prototype supports cached CSV exports. This makes demos and evaluations repeatable and allows inspection of the exact data that produced a summary.
- Limitations. The summaries focus on expenditure priorities rather than net budgets. Revenues are excluded when `spending_only=True` to avoid negative shares. Also, because we aggregate at a department level, the system cannot explain shifts inside a department (e.g., which programs changed).

4.3 Requirements

From the use cases above, we created a small set of requirements. They are split into functional requirements (what the system must do) and non-functional requirements (how it should behave).

- Functional:
  - F1 (Topic + timeline queries): The system must answer questions about a department or topic with an optional time window. This includes simple mappings (e.g., “education” → Schul- und Sportdepartement) and basic fallback matching for near-misses.
  - F2 (Traceable output): Return JSON containing the natural-language summary, share %, slope, time window, and optional context about other departments.
  - F3 (Citywide follow-up): Provide both topic-specific and “citywide” summaries to handle generalization-level follow-up questions.
- Non-functional:
  - NF1 (Reproducibility): The system must run on cached CSV data and produce deterministic outputs so results can be repeated and checked.
  - NF2 (Robustness): Trend estimation should be stable for short series and not flip due to outliers; the system should also fail gracefully if a query does not match any department cleanly.
  - NF3 (Interpretability): Messages must stay short and readable, and they must expose uncertainty cues (membership μ) so users can see whether a label is a strong or weak match.

5 Artifact Design and Implementation

5.1 System overview

The prototype follows a simple pipeline: load data → compute shares and trends → assign fuzzy labels → generate a response. A small CLI script orchestrates the process. It either loads a cached dataset (`python_code/zrh_budget_by_dept_year.csv`) or fetches fresh data from the API, computes department shares, trend slopes, and linguistic labels, and then outputs either batch summaries or a JSON answer to a single request. Figure 1 shows the end-to-end flow. Internally, the pipeline is: data source (API/CSV) → trend module → fuzzy labeling → response templates, exposed through the deterministic NLU parser and a query function.

5.2 Iterative development

The artifact was developed in two iterations:

- Iteration 1 (proof of concept v0): a minimal JSON request/response prototype that executed a small set of scripted queries and returned fuzzy driven linguistic statements done to validate that such an idea is feasible and that it can provide consistent results.
- Iteration 2 (proof of concept v1): the current version with a Streamlit UI, CLI interface, simple NLU parser, query service, and a reusable summarizer module.

5.3 Data processing

Data loading is handled by `load_or_fetch`. If cached aggregates exist, the system uses them to keep demos and evaluations deterministic. Otherwise, it fetches per-department totals from the API, converts amount fields (`betrag`) to numeric CHF, and sums two-digit account rows (`sachkonto`) to department

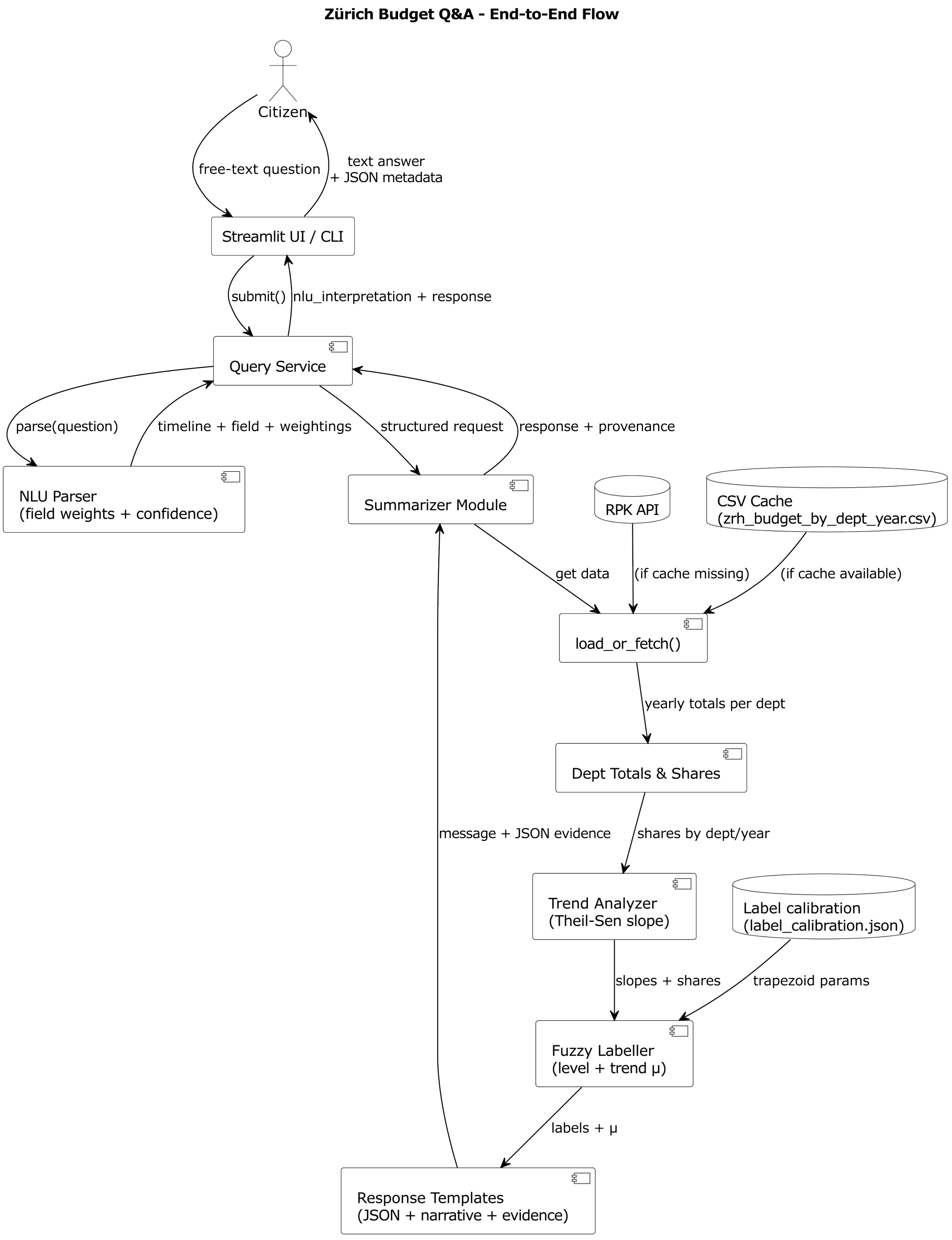
Zürich Budget Q&A - End-to-End Flow
Citizen
free-text question
text answer + JSON metadata
Streamlit UI / CLI
submit()
nlu_interpretation + response
Query Service
parse(question)
timeline + field + weightings
structured request
response + provenance
NLU Parser (field weights + confidence)
Summarizer Module
RPK API
CSV Cache (zrh_budget_by_dept_year.csv)
get data
(if cache missing)
(if cache available)
load_or_fetch()
yearly totals per dept
Dept Totals & Shares
message + JSON evidence
shares by dept/year
Trend Analyzer (Theil-Sen slope)
Label calibration (label_calibration.json)
slopes + shares
trapezoid params
Fuzzy Labeller (level + trend μ)
labels + μ
Response Templates (JSON + narrative + evidence)

Figure 1: End-to-end flow of the Zürich budget Q&A prototype

totals per year (`python_code/summarizer/zurich_budget_linguistic_summaries.py`).

To keep the story consistent ("public spending"), we optionally filter out revenue-related negatives (`spending_only=True`). After aggregation, yearly totals are merged back in, so each department-year row also contains its share of total city spending. These shares are the input for the fuzzy "low/medium/high" labels.

5.4 Trend computation

Trends are computed with a Theil–Sen slope estimator (`theil_sen_slope`). Instead of fitting one least-squares line (which can flip with outliers), it takes the median of all pairwise year-to-year slopes.

We report trends in two forms:

- percentage points per year (`slope_pp_per_year`) for direct interpretability
- relative slope compared to the mean share (`slope_pct_of_mean`) so departments of different size can be compared more fairly.

These values are then used for the fuzzy trend labels.

5.5 Fuzzy model

The fuzzy membership functions are calibrated from Zürich's 2019–2024 distribution and stored in `python_code/summarizer/label_calibration.json` so the configuration is reproducible.

- Spending level (LOW / MEDIUM / HIGH). We build trapezoids from the empirical share percentiles 10/30/50/70/90. This keeps the label boundaries tied to what is "small" or "large" in Zürich's actual budget distribution rather than arbitrary thresholds.
- Trend (FALLING / STABLE / RISING). We use the median absolute deviation (MAD) of relative slopes to set what counts as "stable" in practice. Roughly, "stable" covers a band around the median ($\approx \pm 1 \cdot$ MAD), while "rising" and "falling" extend outward.

For each department, the system outputs a label and membership strength: (`level_label`, `level_mu`, `trend_label`, `trend_mu`). We include μ values because they serve as a transparency cue: a label can be a strong fit (high μ) or only a weak fit (low μ).

Membership functions are calibrated from the empirical distribution of Zürich's 2019–2024 budget shares and persisted in `python_code/summarizer/label_calibration.json` for reproducibility. Level trapezoids use the 10/30/50/70/90 percentiles to define LOW, MEDIUM, and HIGH categories, keeping their overlap proportional to how departments cluster in the data. Trend trapezoids rely on the median absolute deviation (MAD) of the relative slopes (Theil–Sen slope divided by mean share) so that "stable" spans approximately $\pm 1 \cdot$ MAD around the citywide median and "rising/falling" occupy progressively wider shoulders. Each department thus receives (`level_label`, `level_mu`, `trend_label`, `trend_mu`), and the final message concatenates the dominant linguistic values together with supporting statistics to maintain transparency.

Table 1: Level member. trap. (share in %).

Level label	a	b	c	d
LOW	0.00	0.00	2.75	6.37
MEDIUM	2.75	6.37	8.06	11.05
HIGH	8.06	11.05	27.33	100.00

Table 2: Trend member. trap. (relative slope).

Trend label	a	b	c	d
FALLING	-100.00	-4.24	-2.47	-0.71
STABLE	-4.24	-2.47	1.05	2.82
RISING	-0.71	1.05	2.82	100.00

The exact trapezoid boundaries used in the demo are shown in Table~1 and Table~2. These values are also stored in `python_code/summarizer/label_calibration.json`, so the reported configuration matches the executable artifact.

5.6 Response generation

When the CLI is called with `--request`, the program parses the JSON payload, filters the dataset by the requested time window, resolves the department name (including simple English aliases), and returns one of two response types:

1. Topic answer (one department)
2. City-wide answer (largest increases/decreases across departments)

The output mirrors the slide dialogue shown in `presentation/SUC_Final.pdf` and `presentation/SUC_MidTerm.pdf`: a short message plus a structured JSON payload with the evidence.

```
{
  "message": "Since 2019, Schul- und Sportdepartement is stable and
      currently high (29.6% in 2024, +0.30 pp/yr). Meanwhile, the biggest
       increases are in Departement der Industriellen Betriebe, Schul-
      und Sportdepartement.",
  "request": {"timeline": {"since": 2019}, "field": "education", "
      generalization_level": 1},
  "department": "Schul- und Sportdepartement",
  "summary": {
    "level": "high",
    "level_mu": 0.97,
    "trend": "stable",
    "trend_mu": 1.00,
    "share_last_pct": 29.57,
    "slope_pp_per_year": 0.30,
    "slope_pct_of_mean": 1.06,
    "years": {"start": 2019, "end": 2024}
  }
}
```

A small “generalization level” controls whether the system appends a broader context (e.g., top movers citywide). This makes topic questions and follow-up questions work as a simple multi-turn flow, without requiring a full chatbot stack.

For citywide questions, the JSON explicitly exposes ranked movers:

```
{
  "message": "Between 2019 and 2024, the biggest increases are in
      Departement der Industriellen Betriebe (+0.82 pp/yr), Schul- und
      Sportdepartement (+0.30 pp/yr). Decreases are led by
      Sozialdepartement (-1.21 pp/yr), Tiefbau- und
      Entsorgungsdepartement (-0.15 pp/yr).",
  "request": {"timeline": {"since": 2019}, "field": "all", "
     generalization_level": 1},
  "top_increases": [
    {"departement": "Departement der Industriellen Betriebe", "
        slope_pp_per_year": 0.823, "last_share": 12.12},
    {"departement": "Schul- und Sportdepartement", "slope_pp_per_year":
        0.300, "last_share": 29.57}
  ],
  "top_decreases": [
    {"departement": "Sozialdepartement", "slope_pp_per_year": -1.215, "
        last_share": 17.99},
    {"departement": "Tiefbau- und Entsorgungsdepartement", "
        slope_pp_per_year": -0.145, "last_share": 11.90}
  ]
}
```

Because the payload is deterministic (normally the information about the closed year financial statements is not going to change), a third party can rerun the same request and verify both the text and the structured evidence.

5.7 Reproducibility and deployment

Running `python3 summarizer/zurich_budget_linguistic_summaries.py` produces both aggregated and summarized CSV outputs, so intermediate values can be inspected. Dependencies are intentionally small (`pandas`, `numpy`, and optionally `requests`), which keeps the tool lightweight for offline demos. Cached CSVs also act as fixtures for tests and notebooks, and the repository README documents setup and example commands.

For demonstrations, we provide a thin query service and a Streamlit front-end (`python_code/streamlit_app.py`). The UI shows the parsed request (timeline/field/detail level) as part of the JSON response (`request` field), so the interpretation does not need to be duplicated separately. When the request originates from the NLU layer, it also includes confidence metadata such as `field_confidence` and `field_candidates`.

NLU regression set To make the "question understanding" layer reproducible, the repository includes a small test set at `python_code/nlu/nlu_test_set.json`. It contains 70 example utterances with expected slot outputs (timeline, field, detail level), including short follow-ups and compact phrasing:

```
{
  "schema_version": "0.2.7",
  "items": [
    {"id": 1, "utterance": "So what are they spending more on now?", "
        expected": {"timeline": "all", "field": "all", "
        generalization_level": 1}},
    {"id": 2, "utterance": "Whats increasing the most since 2019?", "
        expected": {"timeline": {"since": 2019}, "field": "all", "
        generalization_level": 1}},
    {"id": 69, "utterance": "Detailed: fin dep changes since 2020.", "
        expected": {"timeline": {"since": 2020}, "field": "finance", "
        generalization_level": 2}},
    {"id": 70, "utterance": "In the last 4 years, how did finance change?
        ", "expected": {"timeline": {"since": 2021}, "field": "finance",
        "generalization_level": 1}}
    ]
}
```

The CLI tester (`python_code/nlu/run_nlu_tests.py`) runs the full set and reports exact-match accuracy, so future changes to the parser can be checked for regressions when new paraphrases or languages are added.

6 Evaluation and Results

6.1 Evaluation goals

The evaluation checks whether the artifact fulfils three requirements:

1. Interpretability (RQ1). Can users read a fuzzy-driven summary and correctly restate what it says (topic, timeframe, and direction of change), without going back to raw tables?
2. Robustness and reproducibility (RQ2). Do repeated requests return the same result, and does the interpretation layer (timeline/field parsing + wording constraints) reduce the chance of misreading by always surfacing timeframe and numeric units?

6.2 Evaluation design

We ran two small evaluation rounds with the same group of six peers (STEM backgrounds), using the cached 2019–2024 dataset to keep all runs deterministic.

Round 1 (JSON / CLI). Participants interacted with the JSON request/response version (no UI). Each participant executed two scripted tasks:

- Task A – Citizen quick question: “Find out what happened with education in 2019–2024.” Success required restating the direction of change (stable) and level label (high) for education; additional attempts were allowed if the participant reformulated the query.
- Task B – Citywide follow-up: “Find which departments have the biggest changes in these years.” Participants reused the same timeframe but asked for `field=all`, validating how the system phrases increases/decreases citywide. Success required from participants naming the biggest increases (Departement der Industriellen Betriebe (+0.82 pp/yr), Schul- und Sportdepartement (+0.30 pp/yr)) and decreases (Sozialdepartement (-1.21 pp/yr), Tiefbau- und Entsorgungsdepartement (-0.15 pp/yr))

After the tasks, participants rated clarity, explainability, and usability on a 1–5 Likert scale, selected a preference between the prototype and the Zürich open-data API baseline, and left optional notes.

Round 2 (Streamlit UI). The same tasks were repeated in the Streamlit/CLI interface to check whether the same interaction works in a lightweight UI and whether removing raw JSON improves perceived usability.

Repeatability checks (RQ2). For each task, we sent the same structured request twice and compared the full JSON responses to confirm deterministic outputs.

Intent regression (RQ2). We ran the NLU regression set (`python_code/nlu/nlu_test_set.json`, 70 requests) through `python_code/nlu/run_nlu_tests.py` to confirm that everyday questions map consistently to the intended slots (timeline, field, generalization level), including compact phrasing and German/English mixing.

6.3 Metrics

We report the following indicators:

- Task success rate: share of participants who produced a correct restatement per task, and how often it succeeded on the first attempt.
- Likert averages: mean ratings for clarity, explainability, and usability, plus short qualitative notes.
- Preference split: number of participants preferring the prototype vs. the baseline API.
- Repeatability diff: identical vs. non-identical JSON responses for repeated requests (determinism check).
- NLU exact-match accuracy: percentage of utterances whose parsed slots match the expected intent.

6.4 Results

Running the artifact on the cached 2019–2024 data produced the latest-year summaries in Table~3. Education (Schul- und Sportdepartement) accounts for ~29.6% of spending in 2024 and is labeled HIGH & STABLE in this window. Some departments show clearer directional change (e.g., positive or negative slope in percentage points per year), which results in RISING or FALLING labels with strong membership.

Table 3: Department summaries on the cached 2019–2024 dataset.

Department	Share 2024 (%)	Trend (pp/yr)	Labels
Schul- und Sportdepartement	29.57	+0.30	HIGH & STABLE
Sozialdepartement	17.99	-1.21	MEDIUM & FALLING
Departement der Industriellen Betriebe	12.12	+0.82	MEDIUM & RISING
Tiefbau- und Entsorgungsdepartement	11.90	-0.15	MEDIUM & FALLING

Citizen task walkthrough (RQ1)

All participants completed both tasks successfully (Table~4). For Task A, five participants obtained the correct interpretation on the first request; one required a reformulation. For Task B, two participants reformulated once, mainly to discover or confirm the `field=all` behavior.

Across the API round, the average ratings were 4.0 (clarity), 3.8 (explainability), and 4.17 (usability). Five of six participants preferred the fuzzy-summary workflow over using the API directly. One participant noted they would still use the API when they "need consolidated data".

Table 4: Evaluation Round 1 (API). Requests per task, task success, ratings (1–5), and preference.

Part.	A reqs	A ok	B reqs	B ok	Clarity	Expl.	Usab.	Pref.
P1	1	1	1	1	4	4	4	System
P2	1	1	2	1	4	3	4	API
P3	1	1	1	1	4	4	5	System
P4	1	1	1	1	4	4	4	System
P5	1	1	2	1	4	4	4	System
P6	1	1	1	1	4	4	4	System

Because all participants had technical backgrounds (CS studies or regular use of developer tooling), these scores should be treated as indicative rather than representative of a broad citizen audience (generally people don't know how to work with APIs).

Follow-up evaluation (Round 2 UI)

In the Streamlit/CLI round (Table~5), all six participants solved both tasks on the first request, which suggests that the same underlying logic transfers well to a lightweight UI and that removing raw JSON friction improves interaction.

Table 5: Evaluation Round 2 (Streamlit/CLI). Requests per task, task success, ratings (1–5), and preference.

Part.	A reqs	A ok	B reqs	B ok	Clarity	Expl.	Usab.	Pref.
P1	1	1	1	1	5	4	5	System
P2	1	1	1	1	5	3	5	System
P3	1	1	1	1	4	4	5	System
P4	1	1	1	1	4	4	4	System
P5	1	1	1	1	5	4	5	System
P6	1	1	1	1	4	4	4	System

Average ratings increased to 4.5 (clarity), 3.8 (explainability), and 4.67 (usability), and all six participants preferred the prototype in this format. Qualitative notes are clustered around three themes:

1. Lower interaction friction. Participants explicitly compared the UI to issuing raw requests: "Chat-like messaging is definitely better than an API." (P5)
2. Fast insight without heavy machinery. One participant valued that the prototype stays lightweight: "I like that there's no LLM or other crap involved." (P2)
3. Clear potential, but the scope is still narrow. Participants saw value but asked for more use cases: "The use cases need to be extended." (P3)

Robustness and reproducibility (RQ2)

Determinism-by-design. Replaying the Task A and Task B requests produced identical JSON responses down to floating-point values (diff = 0), confirming that the pipeline behaves deterministically on the cached dataset.

Intent regression. Running `python_code/nlu/run_nlu_tests.py` over the 70-utterance set resulted in 70/70 exact matches (100%). The set includes compact phrasing, multilingual requests, and relative references (e.g., "In the last 4 years…"), which indicates that the slot canonicalization is stable for the phrasing variety represented in the test set.

6.5 Threats to validity

This evaluation is small (n = 6) and biased toward technically fluent participants, so the Likert ratings likely overestimate usability for a general citizen population. The scope is also limited to Zürich and to expenditures only (revenues excluded). The aggregation level is department totals, which hides intra-department changes. Finally, the membership functions are calibrated for 2019–2024; if future budget distributions shift strongly, the calibration should be re-run to avoid label drift.

7 Discussion: Lessons Learned

7.1 What worked well

The fuzzy linguistic summaries did what we hoped: they turn large budget tables into short, readable statements while still keeping the link to the underlying numbers. In both evaluation rounds, participants could restate the main message (level + trend over a stated timeframe), which supports the interpretability goal (RQ1). The interface design also held up across formats: the same JSON-based request/response logic works in a CLI setting, and the Streamlit UI mainly reduced interaction friction rather than changing the underlying model.

7.2 Failure modes and edge cases

Some outputs remain harder to read when the label fit is weak. This happens most often for small departments or for cases where the shares sit near label boundaries, so the dominant label can be a "weak" match. Even when μ makes this visible, a few readers still need more guidance to interpret what "weakly rising" or "weakly falling" means in practice.

Window sensitivity is another edge case. With short municipal series (5–6 years), small shifts of the start year can move a department across the "stable" threshold, especially when the true slope is close to zero. The Theil–Sen estimator reduces sensitivity to outliers, but it does not eliminate sensitivity to short time windows. This supports the idea that a citizen-facing tool should either encourage longer windows or explicitly warn when the trend classification is borderline.

Finally, data quality issues (missing values, zeros, or schema quirks from the API) require careful handling. Even small preprocessing decisions can change the derived shares and therefore the linguistic labels, so these steps must stay transparent and reproducible.

7.3 Limitations

The prototype is intentionally descriptive: it does not generate causal explanations ("why did healthcare rise?"), because the budget data alone does not contain policy context. The current scope is also Zürich-only and aggregated at the department level, which hides shifts inside departments (program-level changes).

On the interaction side, the NLU remains template- and slot-based. It works well for the supported question patterns, but coverage is limited (especially for multilingual phrasing and richer paraphrases). The evaluation is also small and uses a technically fluent peer group, so the results should be treated as indicative rather than representative for a broad citizen audience. Finally, the calibration is tied to the 2019–2024 distribution; if future budgets shift substantially, the membership functions should be recalibrated to avoid label drift.

7.4 Lessons learned

- The evaluations reinforced that interpretability is not only about the summarization method but also about interaction friction. The same logic scored higher once the Streamlit UI removed the need to work with raw JSON.
- We underestimated how much iteration is needed even for a “simple” computing-with-words system: getting membership functions, templates, and provenance to work together in a consistent way took most of the engineering effort.
- Adding lightweight transparency cues (timeframe, slope units, μ) matters. Participants repeatedly pointed to these details as the reason the sentences felt trustworthy rather than “hand-wavy”.

7.5 Implications for citizen-centered design

Even a lightweight NLP layer can bridge the gap between open budget data and the kinds of questions residents ask, as long as the system stays explicit about provenance and confidence. In a practical deployment, the same approach could be embedded into civic explainers or chatbot-style interfaces. However, topic mappings and label wording should be co-designed with residents so that “high”, “low”, and “rising” match how people actually talk about priorities.

8 Conclusion and Outlook

8.1 Conclusion

This project demonstrated that fuzzy linguistic summaries, powered by robust trend estimation and transparent membership scores, can translate Zürich’s open budgets into interpretable statements while remaining traceable to the underlying numbers. Determinism-by-design checks confirmed that the system reproducibly returns the same JSON payload when the same request is issued twice on the cached dataset.

The artifact addresses RQ1 by showing that participants can restate the topic, timeframe, and direction of change after reading the summaries. It addresses RQ2 by demonstrating repeatable outputs and a deterministic interpretation layer that surfaces the selected timeframe and trend units in every response, reducing the risk of misreading.

8.2 Outlook

Next steps include: 1. Scheduling periodic re-calibration runs as new budgets are released. 2. Extending the deterministic NLU coverage to handle broader phrasing and multilingual input. 3. Expanding the data granularity to finer-grained account lines and additional municipalities. 4. Linking external policy artifacts (e.g., council minutes or policy documents) so the system can reference evidence-backed context alongside the descriptive fuzzy summaries.

The immediate roadmap focuses on: 1. Extending response templates to support more linguistic variations without losing clarity. 2. Broadening the request vocabulary and topic mappings in the NLU layer. 3.

Running a larger follow-up evaluation beyond the initial six STEM peers. 4. Presenting the prototype to Stadt Zürich to explore collaboration and validate requirements for a minimum viable product.

8.3 Use of AI tools

AI-based tools were used to improve the grammar and wording of this report. The system design, implementation, and evaluation were carried out by the authors.

References

Bannister, F., & Connolly, R. (2011). The trouble with transparency: A critical review of openness in e-government. *Policy & Internet*, *3*(1), 158–187. https://doi.org/10.2202/1944-2866.1076

Bertot, J. C., Jaeger, P. T., & Grimes, J. M. (2010). Using ICTs to create a culture of transparency in e-government. *Government Information Quarterly*, *27*(3), 264–271. https://doi.org/10.1016/j.giq.2010.03.001

Cucciniello, M., Porumbescu, G. A., & Grimmelikhuijsen, S. G. (2017). 25 years of transparency research: Evidence and future directions. *Public Administration Review*, *77*(1), 32–44. https://doi.org/10.1111/puar.12685

Grimmelikhuijsen, S. G., & Meijer, A. J. (2014). Effects of transparency on the perceived trustworthiness of a government organization: Evidence from an online experiment. *Journal of Public Administration Research and Theory*, *24*(1), 137–157. https://doi.org/10.1093/jopart/mus048

Heald, D. (2006). Varieties of transparency. In C. Hood & D. Heald (Eds.), *Transparency: The key to better governance?* (pp. 25–43). Oxford University Press.

Herrera, F., & Martínez, L. (2000). A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, *8*(6), 746–752. https://doi.org/10.1109/91.890332

Kacprzyk, J., & Zadrożny, S. (2005). Linguistic database summaries and their protoforms: Towards natural language based knowledge discovery tools. *Information Sciences*, *173*(4), 281–304. https://doi.org/10.1016/j.ins.2005.03.002

Kim, N. W., Jung, J., Ko, E.-Y., Han, S., Lee, C. W., Kim, J., & Kim, J. (2016). BudgetMap: Engaging taxpayers in the issue-driven classification of a government budget. *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*, 1028–1039. https://doi.org/10.1145/2818048.2820004

Mendel, J. M., Zadeh, L. A., Trillas, E., Yager, R. R., Lawry, J., Hagras, H., & Guadarrama, S. (2010). What computing with words means to me. *IEEE Computational Intelligence Magazine*, *5*(1), 20–26. https://doi.org/10.1109/MCI.2009.934561

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*(3), 45–77. https://doi.org/10.2753/MIS0742-1222240302

Potuzhnyi, B., & Svirsh, V. (n.d.). *ZuriBudgetFuzzy_SUC: Explaining public budgets with fuzzy driven summaries (code repository)*. GitHub repository. https://github.com/bohdanpotuzhnyi/ZuriBudgetFuzzy_SUC

RPK-API documentation. (2024). Stadt Zürich. https://opendatazurich.github.io/rpk-api/docs/

Sen, P. K. (1968). Estimates of the regression coefficient based on Kendall’s tau. *Journal of the American Statistical Association*, *63*(324), 1379–1389. https://doi.org/10.1080/01621459.1968.10480934

Theil, H. (1950). A rank-invariant method of linear and polynomial regression analysis. *Proceedings of the Koninklijke Nederlandse Akademie van Wetenschappen*, *53*, 386–392.

Tygel, A., Attard, J., Orlandi, F., Campos, M. L. M., & Auer, S. (2015). *“How much?” Is not enough: An analysis of open budget initiatives*. arXiv preprint. https://arxiv.org/abs/1504.01563

Yager, R. R. (1982). A new approach to the summarization of data. *Information Sciences*, *28*(1), 69–86. https://doi.org/10.1016/0020-0255(82)90033-0

Zadeh, L. A. (1996). Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, *4*(2), 103–111. https://doi.org/10.1109/91.493904

Zadeh, L. A. (2002). From computing with numbers to computing with words—from manipulation of measurements to manipulation of perceptions. *International Journal of Applied Mathematics and Computer Science*, *12*(3), 307–324. https://eudml.org/doc/207589