SoftwareX 34 (2026) 102702

Contents lists available at ScienceDirect

SoftwareX

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

annotaid: A browser-based tool for LLM-assisted qualitative coding of open text

Philipp Sprengholz

University of Bamberg, Institute of Psychology, Markusstraße 8a, 96047 Bamberg, Germany

ARTICLE INFO

Keywords:
Qualitative coding
Content analysis
Large language models
Local inference

ABSTRACT

annotaid is a lightweight, browser-based tool that enables social science researchers to classify large collections of open text using locally-running large language models (LLMs). Operating entirely within the user's browser and communicating with a locally-running inference server, *annotaid* processes no data externally, making it suitable for research involving sensitive or confidential text such as interview transcripts or survey responses. The tool has native support for LM Studio and Ollama, and is compatible with any other inference backend that adheres to the OpenAI API protocol. Researchers define a coding scheme through a natural-language system prompt, optionally test it on individual items, and then apply it to an entire dataset provided as a CSV file. Results are returned as an enriched CSV with the model's output appended as a new column. The tool requires no programming knowledge and no cloud subscription, lowering the barrier to LLM-assisted qualitative coding for researchers without computational backgrounds.

1. Code metadata

Field		Value
C1	Current code version	1.0
C2	Permanent link to code/ repository	https://github.com/sprengholz/annotaid
C3	Permanent link to reproducible capsule	https://github.com/sprengholz/annotaid
C4	Legal code license	MIT
C5	Code versioning system used	Git
C6	Software code languages, tools, and services used	HTML5, CSS3, JavaScript (ES6+)
C7	Compilation requirements, operating environments & dependencies	Inference backend that adheres to the OpenAI API protocol (e.g., LM Studio, Ollama)
C8	Link to developer documentation/ manual	https://github.com/sprengholz/annotaid/blob/main/DEVELOPER.md
C9	Support email for questions	philipp.sprengholz@uni-bamberg.de

2. Motivation and significance

Coding open text, i.e., assigning categories or labels to free-text responses such as interview answers, social media posts, or open survey items, is a central task in qualitative and mixed-methods social science research [1,2]. The systematic application of a predefined coding scheme requires researchers to read, interpret, and categorize each item. This process is time-consuming, resource-intensive, and susceptible to intercoder variability, particularly as dataset sizes increase [1].

The emergence of large language models (LLMs) has opened new possibilities for automating or semi-automating this process. Recent studies demonstrate that instruction-tuned LLMs can perform zero-shot text classification with accuracy matching or exceeding trained human coders on a range of social science tasks [3–6]. Because researchers can adapt these models to arbitrary coding schemes by simply formulating instructions in natural language, they represent a promising alternative to traditional annotation workflows that do not require task-specific training data.

However, the practical adoption of LLM-assisted annotation in empirical social science research is constrained by several structural barriers. Cloud services like ChatGPT (OpenAI), Gemini (Google) or Claude (Anthropic) raise significant data privacy concerns when handling sensitive research data that may be subject to institutional ethics approvals, data protection legislation (e.g., the EU General Data Protection Regulation), or confidentiality agreements. Thus, using cloud services for content analysis has been flagged as a concern in the methodological literature [7]. Running LLMs locally can solve this problem, but existing approaches to local LLM annotation (whether custom Python scripts or packages such as GABRIEL [8]) require programming proficiency that many social science researchers do not

E-mail address: philipp.sprengholz@uni-bamberg.de.

https://doi.org/10.1016/j.softx.2026.102702
Received 26 February 2026; Received in revised form 19 April 2026; Accepted 28 April 2026
Available online 4 May 2026

possess. AI-assisted coding environments such as Claude (Anthropic) or Cursor (Anysphere) can lower this barrier by generating annotation pipelines on demand, but presuppose that the researcher can evaluate, execute, and adapt code.

annotaid addresses these barriers by combining the accessibility of a web interface with the privacy guarantees of fully local model inference. The tool targets social scientists, survey researchers, and content analysts who need to code moderate to large volumes of open text (i.e., hundreds to thousands of items) without transmitting data to third-party servers and without writing code.

3. Software description

3.1. Architecture

annotaid is implemented as a single self-contained HTML file that runs in any modern web browser. It communicates with a locally-running inference server; the tool has no server-side component, no database, and no external service dependencies beyond optional web fonts loaded at startup.

3.2. Configuration

3.2.1. Starting inference backend

Before *annotaid* can be used, a local inference service needs to be started. Users need to set up LM Studio [9], Ollama [10], or any other service compatible with the OpenAI API, then select and load an LLM. For reproducibility, users are advised to set the temperature parameter for loaded models to 0, which minimizes variance in the model's output by reducing randomness in token sampling (note that fully deterministic output cannot be guaranteed in all cases, as results may still vary slightly across different hardware or operating systems even at temperature 0). Where the selected model supports an extended thinking or reasoning mode, this should be disabled, as internal chain-of-thought output may interfere with the expected coding format. Cross-Origin Resource Sharing (CORS) must to be enabled for the local inference service.

3.2.2. Starting annotaid

annotaid can be accessed in two ways depending on the user's browser. The GitHub-hosted version at https://sprengholz.github.io/annotaid works directly in Firefox without any additional setup. However, a browser security policy introduced in Chrome 142+ blocks mixed-content requests (i.e., HTTP calls to a local inference server initiated from the GitHub page served over HTTPS) which prevents the hosted version from functioning in current versions of Chrome, Edge, and Safari. Users of these browsers need to serve *annotaid* locally; full setup instructions are provided at https://github.com/sprengholz/annotaid.

3.2.3. Linking annotaid with the inference backend

Users can select their preferred local inference backend and configure the connection in the *annotaid* settings dialog. LM Studio and Ollama are preconfigured, a third option allows users to specify a custom server address manually, making *annotaid* compatible with any inference server that adheres to the OpenAI API protocol. Once a backend has been selected and configured, all models available on the running inference server are automatically listed in *annotaid*.

3.3. Coding workflow

After setting up the connection to the inference backend and selecting a model, users can define a coding instruction, validate it on a small sample, and then apply it at scale.

3.3.1. Defining the coding scheme

The user formulates a coding instruction as a natural-language system prompt in the context panel at the top of the interface. This prompt describes the categories to be assigned, the decision rules to apply, and any output format constraints—functionally equivalent to a codebook entry in traditional content analysis [1]. The panel includes a rough token counter to help researchers monitor prompt length relative to the model's context window. While the tool is primarily designed for zero-shot classification (where the prompt describes the coding scheme without providing labeled examples), few-shot prompting is equally supported; researchers can include example texts with their correct labels in the context panel to guide the model's behavior.

3.3.2. Single-item testing

Before batch processing, the *Test* tab allows the user to submit individual text items interactively and inspect the model's response (see Fig. 1, left panel). This supports iterative refinement of the coding scheme (e.g., adjusting category definitions or output format instructions) without committing to a full processing run. This step is analogous to coder training in traditional content analysis.

3.3.3. Batch processing

In the *Batch* tab, the user uploads a CSV file, selects the column containing the text to be coded, and specifies a name for the output column (see Fig. 1, right panel). A preview applies the current prompt to five randomly sampled rows so the user can verify output quality before initiating the full run. The full batch then processes all rows sequentially or in parallel. Parallelism is configurable from 1 to 16 concurrent requests (in the settings dialog), enabling faster processing on hardware with sufficient compute resources. Interrupted runs can be resumed from the last completed row, avoiding redundant reprocessing.

3.3.4. Output

The enriched dataset is downloaded as a new CSV file that retains all original columns plus the coded output column. This format is directly importable into standard statistical analysis software including SPSS, Stata, R, and Excel, minimizing friction in the transition from annotation to analysis.

3.4. Implementation notes

All logic is handled in vanilla JavaScript within a single HTML file. This prioritizes ease of distribution and use; researchers can download, share, or bookmark *annotaid* with no build step or dependency installation. The trade-off is that the codebase is less modular than a multi-file project; to mitigate this, the code is organized into clearly delimited, commented sections. Developer documentation describing the internal structure is available at the code repository.

Token management is handled conservatively: the context prompt and each input text are passed as separate messages, allowing the inference server to apply its own context window limits. The token counter in the context panel provides a rough estimate based on character count to help users avoid exceeding model limits, though exact tokenization varies by model.

Error handling covers the most common failure modes: unreachable inference server, malformed API responses, and interrupted batch runs. Connection errors are surfaced with descriptive messages in the UI, and interrupted batches can be resumed from the last successfully processed row.

Regarding output validation, *annotaid* currently passes the model's raw text output directly to the output CSV without downstream type checking. Researchers are therefore advised to specify the expected output format precisely in the context prompt and to inspect the output column for unexpected values before analysis. Structured output constraints (e.g., enforcing valid JSON or a fixed label set via the inference server's grammar-based sampling) are a planned feature for future versions.

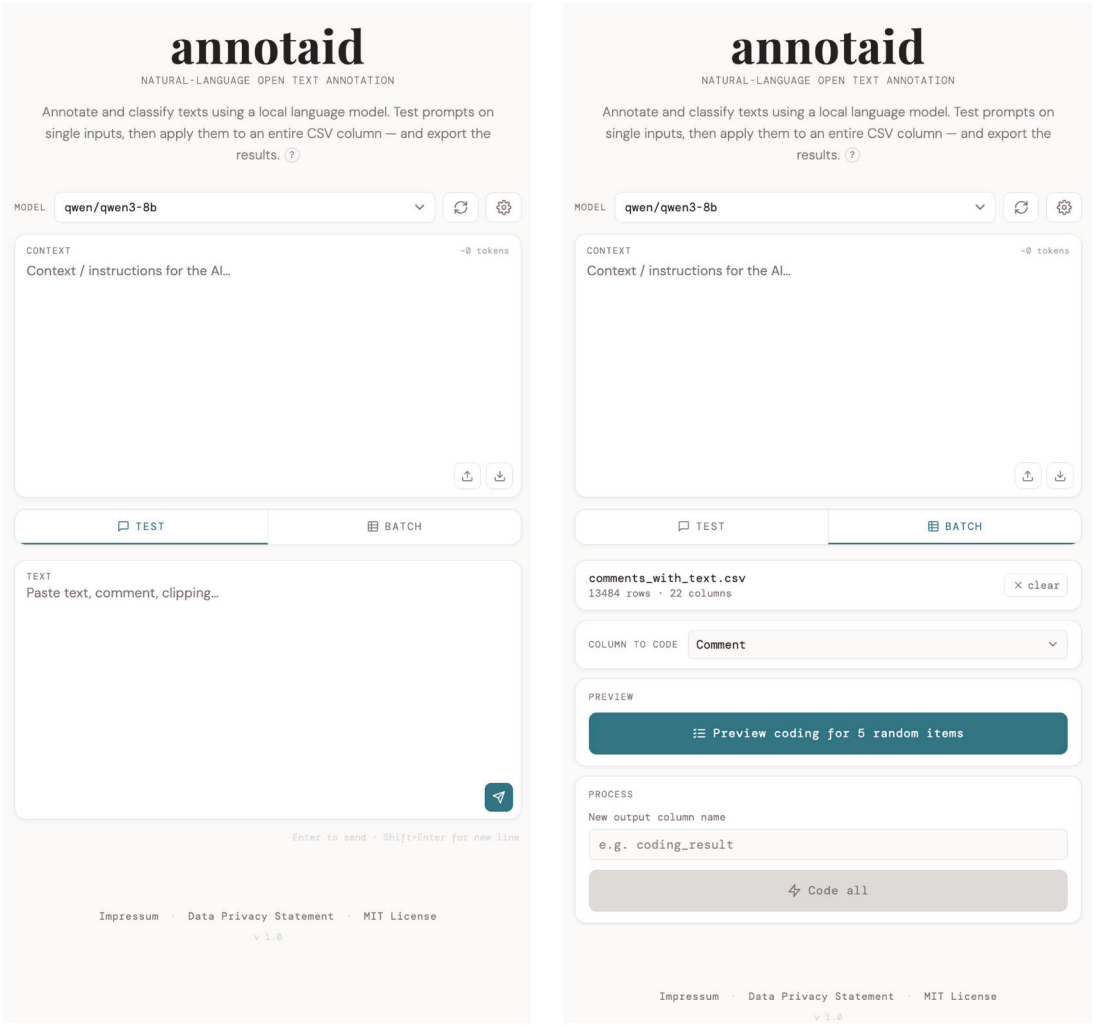

Fig. 1. User interface.

4. Illustrative example

To illustrate the capabilities of *annotaid*, we draw on a study of public comments submitted to a federal rulemaking docket concerning the proposed rescission of the EPA's endangerment finding (the foundational regulatory determination that greenhouse gases threaten public health and welfare). Rulemaking dockets of this kind can receive tens of thousands of public comments within a short submission window, making manual coding at scale practically infeasible. The dataset used here comprised 13,484 prefiltered comments from the public docket (EPA-HQ-OAR-2025–0194), stored as a CSV file with one comment per row. Examples:

```
[Comment 00467] I have a 2014 VW Jetta and just had to pay 3000 to get it fixed while I'm a full time college student working two jobs. My nox catalyst went out, and if my nox was allowed to be deleted I would have not gone without a car for over a week, and it wouldn't have been 3000 to fix something that just causes more problems.
[Comment 10346] I want the EPA to monitor and regulate anything that impacts my and my families life. This includes impacts based upon science that effects climate change. Climate change definitely impacts all our health and well being. As a tax payer and citizen of the US I want my health and well being protected and opposed EPA-HQ-OAR-2025–0194–0093.
[Comment 01143] Thank you for the opportunity to provide Business Council for Sustainable Energy's comments on this proposal.
```

The first research question concerned whether commenters opposed or supported the rescission. After iterative testing and refinement, the following context prompt was used:

```
You are a precise text classification tool. You output only one word, nothing else.
Your task is to classify the stance of public comments regarding the rescission of the EPA endangerment finding. The EPA endangerment finding (2009) is a formal determination that greenhouse gases threaten public health and welfare, and is the legal foundation for federal greenhouse gas regulations. The current administration is proposing to rescind (remove/revoke) this finding.
Output "oppose" if the commenter opposes the rescission and wants to keep the endangerment finding in place. This includes comments that: argue climate change is real and human-caused; cite health or environmental harms from greenhouse gases; defend the scientific consensus; warn of consequences of deregulation; express concern for future generations, vulnerable communities, or ecosystems; argue the EPA
```

```
has a legal duty to regulate greenhouse gases; or criticize the rescission as politically motivated.
  Output "support" if the commenter supports the rescission and wants to remove the endangerment finding. This includes comments that: deny or downplay climate change or the role of CO2; argue the finding was regulatory overreach or unlawful; cite economic burdens, energy costs, or job losses caused by related regulations; argue CO2 is natural or beneficial to the climate; claim the underlying science is flawed, manipulated, or uncertain; argue the EPA lacks authority to regulate greenhouse gases; or express general support for deregulation or the current administration's energy agenda.
  Output "na" if the comment is empty, incoherent, off-topic, or the stance cannot be clearly determined. This includes: comments consisting only of a name or address; references to attached documents with no stated position; generic patriotic or political statements unrelated to the finding; comments in a language that cannot be understood; or comments that mention the topic but take no discernible side.
  You MUST output exactly one of these three strings: "oppose", "support", "na". No explanation, no punctuation, no capitalization variants. When in doubt, output "na".
```

The full CSV file was batch processed at a concurreny of 1 using the model Qwen 3 8b (Alibaba; 4 bit MLX version, loaded in ML Studio, ~ 4.31 GB). Completion took about 10 h on an Apple MacBook Air (M2 processor, 16 GB RAM). Results showed that about 80% of participants opposed the rescission of the endangerment finding. Comparing the LLM results with a human coder on a subset of 200 randomly selected codings revealed almost perfect intercoder reliability (98% agreement, Cohen's $\kappa = 0.95$).

In the next step, comments were re-coded by argument type. Applied separately to the subsets of supporting and opposing comments, new coding schemes aimed to identify the presence of different argument frames. While most supporters of the rescission made economic arguments, those who opposed it often referenced scientific evidence and the potential consequences for human health and the environment. The two-pass approach (stance and argument coding) illustrates how *annotaid* supports sequential, multi-dimensional coding workflows.

5. Impact and conclusion

annotaid lowers the practical threshold for LLM-assisted text annotation in the social sciences by removing two major barriers to adoption: programming requirements and data privacy concerns. The tool is particularly suited to research contexts where (a) the dataset is too large for manual coding alone but too sensitive for cloud processing, (b) a reproducible and auditable coding procedure is required, and (c) the research team has limited or no programming capacity. The natural-language system prompt used in *annotaid* serves as a transparent and inspectable record of the coding logic, equivalent to a published codebook, that can be included in supplementary materials to support replication [1].

As open-weight models continue to improve in instruction-following ability and task-specific accuracy [4], the quality of local coding is expected to converge toward that of larger proprietary models. Recent benchmarking work demonstrates that fine-tuned open-weight models can already match or exceed zero-shot GPT-3.5 performance on common social science annotation tasks [4]. Tools like *annotaid* are therefore positioned to become increasingly viable components of annotation workflows in empirical research, especially as model quantization techniques make capable models accessible on consumer hardware.

Compared to more flexible approaches such as scripted pipelines or computational notebooks, *annotaid* is intentionally constrained in scope: it is optimized for single-column, prompt-based batch classification and does not support more complex workflows such as multi-column input, conditional coding logic, or programmatic post-processing. These constraints are deliberate, they are what make the tool immediately usable without programming knowledge. However, researchers with more complex annotation needs or greater computational literacy may find scripted approaches more suitable. The context prompt serves as the primary means of controlling model behavior. As such, the reproducibility of results depends critically on careful prompt documentation alongside the model name, version, temperature setting and hardware configuration, all of which should be reported in the methods section of any publication using *annotaid*-generated annotations.

Importantly, *annotaid* complements rather than replaces human coding in qualitative and mixed-methods research. As recommended in the emerging methodological literature on LLM annotation, users are encouraged to validate the model output against a human-coded subsample, particularly for novel or ambiguous coding schemes [5,11].

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the author used Claude Sonnet 4.6 (Anthropic) in order to refine the manuscript text, including section structure and phrasing. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the published article.

CRediT authorship contribution statement

Philipp Sprengholz: Writing – review & editing, Writing – original draft, Software, Methodology, 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.

References

[1] Krippendorff K. Content analysis: an introduction to its methodology. 2455 teller road. Thousand Oaks California: SAGE Publications, Inc.; 2019. https://doi.org/10.4135/9781071878781. 91320.

[2] Neuendorf KA. The content analysis guidebook. 2455 teller road. Thousand Oaks California: SAGE Publications, Inc; 2017. https://doi.org/10.4135/9781071802878. 91320.

[3] Gilardi F, Alizadeh M, Kubli M. ChatGPT outperforms crowd workers for text-annotation tasks. Proc Natl Acad Sci USA 2023;120:e2305016120. https://doi.org/10.1073/pnas.2305016120.

[4] Alizadeh M, Kubli M, Samei Z, Dehghani S, Zahedivafa M, Bermeo JD, et al. Open-source LLMs for text annotation: a practical guide for model setting and fine-tuning. J Comput Soc Sc 2025;8:17. https://doi.org/10.1007/s42001-024-00345-9.

[5] Abdurahman S, Salkhordeh Ziabari A, Moore AK, Bartels DM, Dehghani M. A primer for evaluating large language models in social-science research. Adv Methods Pract Psychol Sci 2025;8:25152459251325174. https://doi.org/10.1177/25152459251325174.

[6] Törnberg P. Large language models outperform expert coders and supervised classifiers at annotating political social Media messages. Soc Sci Comput Rev 2025;43:1181–95. https://doi.org/10.1177/08944393241286471.

[7] Ollion É, Shen R, Macanovic A, Chatelain A. The dangers of using proprietary LLMs for research. Nat Mach Intell 2024;6:4–5. https://doi.org/10.1038/s42256-023-00783-6.

[8] Asirvatham H, Mokski E, Shleifer A. GPT as a measurement tool. Cambridge, MA: National Bureau of Economic Research; 2026. https://doi.org/10.3386/w34834.

[9] LM Studio. Local AI, on your computer. https://lmstudio.ai; 2025 (accessed February 24, 2026).

[10] Ollama. Start building with open models. https://ollama.com; 2025 (accessed February 24, 2026).

[11] Bail CA. Can Generative AI improve social science? Proc Natl Acad Sci USA 2024;121:e2314021121. https://doi.org/10.1073/pnas.2314021121.