CaBSALLM: Efficient Context-Aware Batch Annotation of Conversational Streams with Large Language Models

Mohammadsadegh Abolhasani[1], **Reza Mousavi**[2], **Paul Jen-Hwa Hu**[1]
[1] University of Utah, [2] University of Virginia
Correspondence: sadegh.abolhasani@utah.edu

Abstract

Analyses of parasocial cues in live-stream chats require accurate, efficient, and scalable annotations. However, manual annotations are tedious and large language models (LLMs) often err in applying subjective, discourse-dependent labels. We propose Context-aware Batching for Stream Annotation with LLMs (CaBSALLM), an efficient pipeline that incorporates lightweight conversational context and a novel dynamic batching method to increase throughput and scalability. Compared with state-of-the-art pipelines, this generalizable pipeline is significantly more time- and cost-efficient, while achieving comparable or better predictive performance and agreement. All codes and data are available at github.com/abolhasani/ACL_PSR_CUE

1 Introduction

Large-scale annotations of subjective, discourse-dependent social interactions remain a critical bottleneck in computational social science (Zimba and Gasparyan, 2021). Human expert annotations are inefficient, and crowdwork is costly and difficult to standardize for nuanced constructs (Mikulová et al., 2022; Horych et al., 2025). Researchers increasingly turn to LLMs for automated annotations, given their human-level performance in some settings (Tseng et al., 2025). When annotating millions of social media chats, posts, and comments under tight constraints, existing LLM pipelines face a persistent quality–deployability trade-off, as high-performing pipelines typically involve prohibitive cost and latency (Zhou et al., 2024), while efficiency-oriented ones tend to lose the contextual nuance needed for subtle interpretations (Lu et al., 2025; Yang et al., 2024). As a result, researchers are often forced to choose between unaffordable accuracy and scalable, shallow annotation.

This study addresses this important challenge by proposing Context-aware Batching for Stream Annotation with LLMs (CaBSALLM), which incorporates a lightweight, context-aware dynamic batching mechanism that adjusts the batch size in real time based on throughput behavior. In this effort, the pipeline preserves the interpretive depth essential to complex social annotations while substantially reducing runtime and inference costs.

Parasocial interactions (PSIs) exemplify an important application for this broader methodological challenge. PSIs refer to illusory social experiences in which a person feels a reciprocal connection to a public figure in media despite the fundamentally one-sided nature of the relationship (Horton and Wohl, 1956). These interactions are fostered and intensified by widespread social media and live streams (Schramm et al., 2024), shaping individuals' media involvement and determining important persuasion outcomes such as impulse buying (Tukachinsky et al., 2020). However, PSIs are difficult to annotate accurately at scale because their cues are subtle, contextual, and psychologically layered. Overall, PSIs not only represent an important and substantive application area, but also constitute a demanding setting that requires an effective, efficient strategy for scalable LLM-enabled annotations. This study develops CaBSALLM, an efficient annotation pipeline for analyzing PSIs in live streams, and uses this setting as a motivating and high-difficulty testbed.

2 Related Work

LLMs as annotators While crowdsourcing has been widely used to annotate subjective social constructs (e.g., toxicity, stance), it becomes cost-prohibitive and unscaleable for voluminous chats in live streams (Mikulová et al., 2022). Recent work has pivoted to LLMs as surrogate annotators, achieving performance parity with human crowdsourced annotators in domains such as propaganda detection and discourse analysis (Sahitaj

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 615–636

et al., 2025; Petukhova and Kochmar, 2025). Social science constructs like PSI are inherently ambiguous and discourse-dependent; naive prompting often yields high disagreement rates or superficially plausible but semantically incorrect labels (Gu et al., 2025). Reliability-centric frameworks are designed to address the limitations in accuracy and agreement, including "explain-then-annotate" (AnnoLLM) (He et al., 2024), multi-model voting committees (LiaHR) (Chochlakis et al., 2025), and event extraction (Lu et al., 2025). While these frameworks improve reliability, they typically require multiple inference passes per item, which exacerbates time latency, increases costs, and thus makes them impractical for (nearly) real-time, scalable deployment for high-volume annotations.

Context-Awareness and Discourse Modeling Detecting PSI cues requires resolving the "interactional scaffolding" in live-streams (e.g., addressivity, shared rituals, and emotional cues), which tend to be dissolved when chat messages or comments are processed in isolation (McLaughlin and Wohn, 2021). For instance, a single word like "classic" could be a generic term, or refer to a specific inside joke (community ritual) or a sarcastic critique (negative dimension), depending on the preceding event or the streamer's reaction. Prior work attempts to capture such dependencies via Retrieval-Augmented Generation (RAG) or by ingesting entire conversation histories (Hagström et al., 2025; Cao et al., 2025). But full-history ingestion is computationally prohibitive, and standard RAG tends to introduce retrieval noise that distracts analyses of subtle interpersonal signals. An optimal middle ground is needed to capture the *local* conversational window to resolve immediate temporal dependencies (e.g., reactions to recent events), while also providing a *global* entity profile to provide premises for static community references. We respond to the required middle ground and operationalize context that disambiguates "borderline cases" (defined formally in Section 3.2), where isolated inferences typically hallucinate.

Inference Efficiency and Batch Prompting To address the economic constraints for large-scale deployment, batch prompting amortizes fixed instruction overhead across multiple samples by grouping multiple inputs into a single prompt (Cheng et al., 2023; Zhang et al., 2024; Korikov et al., 2025). While effective for straightforward classification, batching is associated with significant instability

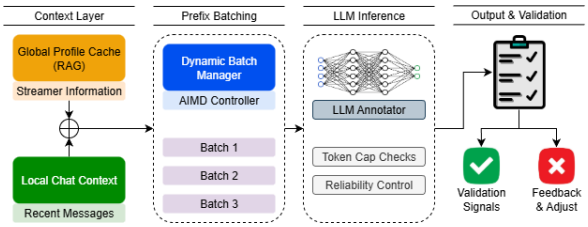

Figure 1: Overview of the proposed annotation pipeline.

in complex extraction tasks; for example, large batches increase the risk of "lost-in-the-middle" errors, context window overflows, and schema validation failures (Du et al., 2025). Most existing frameworks rely on static batch sizes or offline optimization (Fernandez et al., 2025), which are brittle in live-stream environments where message length and throughput vary widely. There is a lack of *adaptive* frameworks capable of modulating batch sizes at inference time to balance "token-feasibility" against the risk of model failure.

The Gap Prevalent pipelines force a binary choice between reliability-centric frameworks (e.g., AnnoLLM) that are robust but too slow for real-time streams, and high-efficiency batching that lacks the discourse-awareness required for subjective PSI annotations. This study bridges this divide and proposes a LLM-enabled pipeline that integrates a two-layer context (Global+Local) to resolve social ambiguity, driven by a dynamic batching controller based on the Additive-Increase/Multiplicative-Decrease (AIMD) algorithm that optimizes throughput and cost in real-time without sacrificing the semantic precision necessary for computational social science.

3 Method

PSIs are scaffolded by addressivity and chat interactivity, featuring recurrent ritualized participation with emotional support and financial investment through platform affordances (Hamilton et al., 2014; Jodén and Strandell, 2022; Kowert and Daniel, 2021). Building on extant literature, we consider nine aspects to operationalize PSI cues in live-stream chats: direct address, attachment and affection, self-disclosure, reply seeking, community rituals, monetary actions, backseat guidance, emote usage, and message valence (dimension).

3.1 Task and Label Schema

The proposed pipeline supports efficient and accurate annotations of chat messages to identify PSI cues (e.g., *DirectAddress*, *SelfDisclose*) and (*Di-*

Table 1: Concise overview of the PSI annotation schema. Examples are illustrative rather than exhaustive.

Aspect	Cue	Definition	Example
`DirectAddress`	Direct address	Message explicitly targets the streamer as the addressee through name, @mention, or clear second-person reference.	"Emiru, you are late today"; "you need to see this"
`Attachment`	Attachment / affection	Message expresses warmth, care, longing, loyalty, pride, or relational closeness toward the streamer.	"love u"; "you look gorgeous today"
`SelfDisclose`	Self-disclosure / identification	Viewer shares personal information, experience, or similarity in a way that invites relational connection.	"I had a rough day, thanks for being here"; "I'm also a grad student"
`ReplySeeking`	Reply seeking	Message seeks acknowledgment, recognition, or a direct response from the streamer.	"notice me"; "did you see my message?"
`CommRitual`	Community ritual	Message invokes shared rituals, inside jokes, coordinated norms, or co-created chat practices.	"chat do the thing"; "only real ones remember"
`Monetary`	Monetary support	Message signals subscription, donation, gifted support, renewal, streaks, or similar financial contribution.	"gifted 5 subs"; "resubscribed for 24 months"
`Backseat`	Backseat guidance	Message renders advice, direction, coaching, or suggestions about gameplay or content	"go left"; "use ult now"
`Emotes`	Emote / emoji usage	Message contains Twitch-style emotes, emoji, or affective symbolic tokens.	"KEKW"; "<3"
`Dimension`	Valence / stance	Overall stance expressed toward the streamer or community, coded as positive (`P`) or negative (`N`).	`P`: "love this stream"; `N`: "you're an idiot "

mension $\in \{\texttt{P}, \texttt{N}\}$), using a nine-aspect schema. Each message is labeled independently at the output level (i.e., one label object per input message), allowing for interpretation that conditions on contextual signals. To make the annotation task explicit, we operationalize PSI expressions in livestream chats using a nine-field schema that combines relational, interactional, and surface-form cues. This schema indicates whether a message directly addresses the streamer, expresses attachment, discloses personal information in a relational manner, seeks acknowledgment, invokes community rituals, signals monetary support, provides backseat guidance, contains emotes/emojis, or conveys a positive versus negative stance. Table 1 presents the schema with concise definitions and representative examples. Figure 1 depicts the overall process.

3.2 Baselines and Diagnostic Active Learning

The proposed pipeline is composed of three stages: (1) A few-shot baseline prompt with minimal instructions but without discourse context, (2) an enhanced-prompt baseline integrates PSI domain knowledge to improve construct alignment, and (3) a diagnostic semi-active learning workflow to identify failure modes using a pool-based committee to aggregate multiple stochastic LLM annotations per item, and routes disagreements (e.g., vote entropy) or high-uncertainty "borderline items" for human reviews. Because label variations can be viewed as either signal or noise, disagreements are informative (Gruber et al., 2025). An analysis of the borderline instances reveals that dominant disagreements are discourse-dependent rather than lexical, with a notable concentration in addressee- and intent-sensitive labels (e.g., *DirectAddress*, *Attachment*, *ReplySeeking*). Hence, It is valuable to incorporate short-range conversational context where label uncertainty is high, particularly a batching design that *combines* multiple messages with local context to improve both predictive performance and throughput. This analysis inspires the proposed context-aware pipeline as a targeted remedy for construct-boundary ambiguities in annotations. Appendix A describes the semi-active learning diagnostics.

3.3 Two-Layer Context Augmented with Sequential Batching

With a stream of chat messages as input, the proposed pipeline views annotation as a discourse-aware interpretation by supplying two complementary context layers:

Global streamer context (G_s). A compact, structured profile (e.g., nicknames, friends) is built for each streamer s, which uses an entity-centric, cache-backed open-web RAG profiling (web-search retrieval and structured profile generation), caches it, and then reuses it across batches.

Local batch context (L_i). A contiguous window of K input messages $B_i = \{m_i, \ldots, m_{i+K-1}\}$ is processed at stream position i to construct a local context summary that captures the topic/game/event to which the chat relates, then identi-

fies salient entities, and interaction patterns. Contiguity is enforced to keep discourse coherence.

LLM annotation. Labels are conditioned on both contexts and the combined message list:

$$\hat{Y}_i = f_\theta(G_s, L_i, B_i),$$

where f_θ outputs a label object per message in B_i. For practical reasons, strict output validation is applied to parseability and schema to support scalable deployment with automatic retry and resumability.

3.4 Adaptive Prefix Batching (dynamic K)

Effective grasping of local temporal context in chat messages requires batching and group processing. But large batches create output pressure and increase failure risk. A controller selects K adaptively for each step using (i) a hard token-feasibility cap and (ii) an online AIMD controller guided by observable signals (e.g., parsing success).

Token-feasibility cap. Let C be the model context window, $\alpha \in (0, 1)$ indicate a safety discount rate, M_{out} the reserved max output tokens, and B_{safe} an additional buffer. The input budget is

$$C_{\text{in}} = \alpha C - M_{\text{out}} - B_{\text{safe}}.$$

Let $H(G_s, L_i)$ indicate fixed overhead tokens (instructions, schema, context), and let $T_{\text{msg}}(B_i)$ denote the tokenized packed messages. The feasibility constraint is

$$H(G_s, L_i) + T_{\text{msg}}(B_i) \leq C_{\text{in}},$$

and $K_{\text{token}}(i)$ is the maximum K that satisfies it.

Feasibility-Guided AIMD updates. Because newly arrived streams lack ground-truth labels, the controller uses only observable proxies per attempt: a hard failure flag F (e.g., parse/schema failure), and an operational pressure flag T (e.g., output nearing M_{out}). When outputs are available, a deterministic validator yields a per-item validity rate $Y \in [0, 1]$. Next, these outputs are integrated into a bounded loss g_t, tracked with an EWMA $\bar{g}_t$:

$$g_t = \text{loss}(F_t, T_t, Y_t), \qquad \bar{g}_t = \beta \bar{g}_{t-1} + (1-\beta) g_t.$$

Next, batch size updates with additive-increase/multiplicative-decrease (AIMD) and a cooldown to prevent oscillations:

$$K \leftarrow \begin{cases} \lfloor \gamma K \rfloor & \text{if } F_t = 1 \text{ or } T_t = 1 \\ K & \text{if in cooldown} \\ K + \Delta & \text{if } \bar{g}_t \leq \tau \\ \lfloor \gamma K \rfloor & \text{otherwise} \end{cases}$$

Table 2: Benchmark landscape used in the main evaluation. The appendix reports the full variant inventory and per-method results.

Pipeline Family	Representative Methods
Prompting-only	SimplePrompt: ZeroShot, FewShot, CoT, ToT
Domain-grounded single-item	DomainPrompt; DomainPrompt+CCoT
Fixed-K batching ablations	CaBSALLM-FixedK=5; CaBSALLM-FixedK=70
Reliability-oriented pipelines	LiaHR Adaptation; AnnoLLM Adaptation; EE Adaptation
Holdout fine-tuning comparator	DomainPrompt-FT
Proposed pipeline	CaBSALLM

$$K \leftarrow \min\{K, K_{\text{token}}(i)\}.$$

With Δ (increase step) and γ (decrease factor) as hyperparameters. If an attempt fails, the *same* prefix window is retried, starting at i with a reduced K; with the pointer advancing only after a successful, validated output. Appendix B presents implementation details.

3.5 Prompting and Execution

To perform the focal annotation task, we employ a zero-shot, schema-guided, batch-prompting strategy, combined with explicit decision rules and contrastive behavioral demonstrations embedded with layered context. Please see Appendix C for details.

4 Results

We use real-world data to evaluate the proposed pipeline relative to 20 benchmarks, see Appendix D for details. To clarify the evaluation landscape, the study compares CaBSALLM with multiple benchmark families shown in Table 2. Table 3 presents aggregate comparative results of the six best-performing pipelines: two variants derived from the proposed pipeline using fixed low- or high-batch strategy, a fine-tuned LLM trained on borderline data and tested on a disjoint holdout set (DomainPrompt-FT), and three adapted reliability-oriented pipelines (LiaHR, AnnoLLM, and Event Extraction). No single metric should be interpreted in isolation. Higher predictive performance values are preferable, whereas lower cost and latency reflect greater efficiency. MCN-A complements predictive metrics by indicating whether a pipeline exhibits systematic disagreement with human annotations under McNemar's exact test. Two pipelines with similar F1 scores may meaningfully differ in consistently over- or under-calling PSI cues. Thus,

Table 3: Aggregate quality and efficiency over 9 labels. *mUSD/msg* denotes milli-USD per message. MCN-A indicates the number of labels without systematic disagreement against the human annotator in the McNemar test.

Method	Model	Mean-F1	M-Rec.	M-Prec.	M-Acc.	Tok_{in}/msg	Tok_{out}/msg	Sec/msg	Cost (mUSD/msg)	MCN-A
CaBSALLM	GPT-5.1	0.866	0.851	**0.884**	**0.949**	193	117	1.29	1.27	6
CaBSALLM-FixedK=5	GPT-5.1	0.861	0.848	0.878	0.947	558	127	1.78	1.450	**7**
CaBSALLM-FixedK=70	GPT-5.1	0.848	0.856	0.841	0.934	**80**	85	**0.80**	**0.916**	4
AnnoLLM Adaptation	GPT-5.1	0.867	0.857	0.881	0.948	9394	666	11.96	8.23	3
LiaHR Adaptation	GPT-5.1	0.850	0.856	0.859	0.942	4497	152.5	3.11	2.91	2
EE Adaptation	GPT-5.1	0.840	0.827	0.861	0.937	9882	231	5.36	5.17	1
DomainPrompt-FT	GPT-4.1 FT	**0.871**	**0.883**	0.865	0.945	2000	**56**	1.59	6.05	1

comparative results should be interpreted from the cost-quality-agreement trade-off in which an optimal pipeline preserves predictive performance while lowering operational burden and systematic disagreement. Regarding annotation quality across aggregate metrics, CaBSALLM remains within a narrow margin of the best Mean-F1 value and slightly exceeds the best-performing benchmark in accuracy, suggesting its ability to achieve high annotation quality while preserving efficiency gains. Appendix E provides illustrated cases in which local context corrects labels previously misclassified. Deployment-ready PSI annotations require a pipeline capable of satisfying five criteria holistically: prediction performance; cost efficiency and time efficiency (Aguda et al., 2024); real-time operability (Zhan et al., 2025); and consistency with human judgments (Lee et al., 2025). In a holistic evaluation, CaBSALLM outperforms strong benchmarks. Compared with DomainPrompt-FT, it reduces per-message latency by 18.9% and inference cost by 79% (using 90.4% fewer input tokens), while achieving comparable F1 (0.57% lower) and notably higher agreement according to a McNemar test (6 versus 1; see Appendix F for details). The fixed-context results reveal why static choices are brittle. FixedK=5 reflects tight control (aggressive) and increases input tokens by 208% and latency by 37% for each additional label agreement, while exhibiting a marginal reduction in prediction performance. In contrast, FixedK=70 is loose control and improves efficiency by reducing time per message by 38% and cost by 27%, but at the expense of performance (a 2% drop in F1) and agreement. Beyond its higher per-message cost and latency, the fine-tuned benchmark also creates training overhead and requires substantial ground-truth data (Pangakis and Wolken, 2024); however, it does not improve overall efficacy relative to the proposed pipeline.

Compared with reliability-oriented benchmarks, CaBSALLM achieves an aggregate accuracy comparable to that of the best-performing benchmark, with an accuracy 0.11% higher than that benchmark. The proposed pipeline also yields substantial efficiency and agreement improvements, lowering per-message cost by 56%, latency by 58.5%, input tokens by 95%, and output tokens by 23% than LiaHR. Paired directional-disagreement analysis shows that the proposed pipeline passes twice as many fields as the best-performing benchmark (6 versus 3), implying fewer systematic disagreements with human annotations. The holistic evaluation reveals the superiority of CaBSALLM over prevalent pipelines for real-world deployment, with substantially lower cost and latency, notable improvements in agreement, and comparable effectiveness for identifying PSI cues. Ablation analysis suggests that a substantial portion of the efficiency improvement is stemming from the AIMD-based controller rather than from prompt design or context alone. Appendix G presents benchmark details, per-label breakdowns, and additional analyses.

5 Conclusion

We propose an efficient pipeline for annotating live-stream chats for PSI cues. This pipeline combines domain knowledge with lightweight conversational context and dynamic batching to increase throughput under stringent operational constraints. Compared with prevalent benchmarks, it achieves comparable performance, substantially reduces per-item cost and latency, and improves paired agreement with human annotations. Evaluation results reveal the value of inference-time context and batching for efficient, scalable annotations by LLMs, which is central to the cost-quality frontier for computational social science research that requires efficacious, efficient, and scalable deployments subject to resource constraints.

Limitations

This study serves as a point of departure for annotating live chats for computational social science research, and it has several limitations. For example, the evaluation is limited in scope, using a single

platform and N=2,520 Twitch user chat messages with a fixed 9-label schema, which likely confines the generalizability of the reported results across platforms (e.g., TikTok Live), user communities, languages, or cultures. Its approach binarizes PSI cues and simplifies the dimension to a dichotomy, {P, N}, which facilitates deployment but might collapse meaningful nuance (e.g., mixed stance). In a related sense, the evaluation inherits the limitations of subjective human labels. For example, some cues (e.g., ATTACHMENT, SELFDISCLOSE) and stance are arguably ambiguous and intrinsically subjective.

Also, LLM variability may create difficulties for autonomous streamer context gathering, due to web searching capability constraints. In addition, dynamic batching is tuned to observable proxies (schema/parse validity, truncation/pressure, latency), which may not be perfect correlates of semantic label quality and likely vary across model versions, providers, and load conditions. In a related sense, the adaptive prefix batching controller is conservative by design, prioritizing constraint satisfaction (context feasibility, schema validity, and truncation avoidance) and relying on noisy proxy signals. As a result, both quality and efficiency might shift under different API policies or context-window constraints. This limitation calls for more robust controls that incorporate uncertainty, stability across repeated decodes, and explicit budget-aware optimization.

Finally, the reported results are conditional on specific API-based LLMs and inference settings. Reexaminations that use local/open-weight models (including task-specific fine-tuning) are essential and can further scrutinize the observed cost-quality trade-off, especially for deployments featuring computational resource availability and budget constraints that shift the preferred point on the frontier. Overall, the proposed pipeline is designed for accurate, efficient, and high-volume annotations in resource-constrained settings. As constraints relax, preferences may shift toward pipelines that can better align with available computational capacity and budget. Adapting this pipeline for open-weight models would further democratize research by reducing costs and enhancing privacy.

Ethical considerations

This project is exempt from IRB review. The dataset contains no personally identifiable information at individual levels and is gathered exclusively through official platform APIs in compliance with applicable terms of service. The study is conducted for academic research purposes only, without any funding or support from commercial firms or research institutions.

References

Toyin D. Aguda, Suchetha Siddagangappa, Elena Kochkina, Simerjot Kaur, Dongsheng Wang, and Charese Smiley. 2024. Large language models as financial data annotators: A study on effectiveness and efficiency. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10124–10145, Torino, Italia. ELRA and ICCL.

Albert H. Bowker. 1948. A Test for Symmetry in Contingency Tables. *Journal of the American Statistical Association*, 43(244):572–574. Publisher: Taylor & Francis.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Bowen Cao, Deng Cai, and Wai Lam. 2025. InfiniteICL: Breaking the limit of context window size via long short-term memory transformation. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 11402–11415, Vienna, Austria. Association for Computational Linguistics.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. 2023. Batch Prompting: Efficient Inference with Large Language Model APIs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 792–810, Singapore. Association for Computational Linguistics.

Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. 2023. Contrastive chain-of-thought prompting. *Preprint*, arXiv:2311.09277.

Dah-Ming Chiu and Raj Jain. 1989. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1):1–14.

Georgios Chochlakis, Peter Wu, Tikka Arjun Singh Bedi, Marcus Ma, Kristina Lerman, and Shrikanth Narayanan. 2025. Humans Hallucinate Too: Language Models Identify and Correct Subjective Annotation Errors With Label-in-a-Haystack Prompts. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 19648–19667, Suzhou, China. Association for Computational Linguistics.

Yufeng Du, Minyang Tian, Srikanth Ronanki, Subendhu Rongali, Sravan Babu Bodapati, Aram Galstyan, Azton Wells, Roy Schwartz, Eliu A Huerta, and Hao Peng. 2025. Context length alone hurts LLM performance despite perfect retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23281–23298, Suzhou, China. Association for Computational Linguistics.

Jared Fernandez, Clara Na, Vashisth Tiwari, Yonatan Bisk, Sasha Luccioni, and Emma Strubell. 2025. Energy considerations of large language model inference and efficiency optimizations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 32556–32569, Vienna, Austria. Association for Computational Linguistics.

Cornelia Gruber, Helen Alber, Bernd Bischl, Göran Kauermann, Barbara Plank, and Matthias Aßenmacher. 2025. Revisiting Active Learning under (Human) Label Variation. In *Proceedings of the The 4th Workshop on Perspectivist Approaches to NLP*, pages 75–86, Suzhou, China. Association for Computational Linguistics.

Feng Gu, Zongxia Li, Carlos Rafael Colon, Benjamin Evans, Ishani Mondal, and Jordan Lee Boyd-Graber. 2025. Large Language Models Are Effective Human Annotation Assistants, But Not Good Independent Annotators. *arXiv preprint*. ArXiv:2503.06778 [cs] version: 2.

Lovisa Hagström, Sara Vera Marjanovic, Haeun Yu, Arnav Arora, Christina Lioma, Maria Maistro, Pepa Atanasova, and Isabelle Augenstein. 2025. A reality check on context utilisation for retrieval-augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19691–19730, Vienna, Austria. Association for Computational Linguistics.

William A. Hamilton, Oliver Garretson, and Andruid Kerne. 2014. Streaming on twitch: fostering participatory communities of play within live mixed media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1315–1324, New York, NY, USA. Association for Computing Machinery.

Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. 2024. AnnoLLM: Making Large Language Models to Be Better Crowdsourced Annotators. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 165–190, Mexico City, Mexico. Association for Computational Linguistics.

D. Horton and R. R. Wohl. 1956. Mass communication and para-social interaction; observations on intimacy at a distance. *Psychiatry*, 19(3):215–229.

Tomáš Horych, Christoph Mandl, Terry Ruas, Andre Greiner-Petter, Bela Gipp, Akiko Aizawa, and Timo Spinde. 2025. The Promises and Pitfalls of LLM Annotations in Dataset Labeling: a Case Study on Media Bias Detection. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 1370–1386, Albuquerque, New Mexico. Association for Computational Linguistics.

Henrik Jodén and Jacob Strandell. 2022. Building viewer engagement through interaction rituals on Twitch.tv. *Information, Communication & Society*, 25(13):1969–1986. Publisher: Routledge _eprint: https://doi.org/10.1080/1369118X.2021.1913211.

Imran Khan. 2025. You Don't Need Prompt Engineering Anymore: The Prompting Inversion. *arXiv preprint*. ArXiv:2510.22251 [cs].

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners. *Preprint*, arXiv:2205.11916.

Anton Korikov, Pan Du, Scott Sanner, and Navid Rekabsaz. 2025. Batched Self-Consistency Improves LLM Relevance Assessment and Ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 32675–32691, Suzhou, China. Association for Computational Linguistics.

Rachel Kowert and Emory Daniel. 2021. The one-and-a-half sided parasocial relationship: The curious case of live streaming. *Computers in Human Behavior Reports*, 4:100150.

Noah Lee, Jiwoo Hong, and James Thorne. 2025. Evaluating the consistency of LLM evaluators. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10650–10659, Abu Dhabi, UAE. Association for Computational Linguistics.

Wenxuan Liu, Zixuan Li, Long Bai, Yuxin Zuo, Daozhu Xu, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2025. Towards Event Extraction with Massive Types: LLM-based Collaborative Annotation and Partitioning Extraction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 34365–34387, Suzhou, China. Association for Computational Linguistics.

Junyu Lu, Kai Ma, Kaichun Wang, Kelaiti Xiao, Roy Ka-Wei Lee, Bo Xu, Liang Yang, and Hongfei Lin. 2025. Is LLM an Overconfident Judge? Unveiling the Capabilities of LLMs in Detecting Offensive Language with Annotation Disagreement. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 5609–5626, Vienna, Austria. Association for Computational Linguistics.

Caitlin McLaughlin and Donghee Yvette Wohn. 2021. Predictors of parasocial interaction and relationships in live streaming. *Convergence*, 27(6):1714–1734.

Quinn McNemar. 1947. Note on the Sampling Error of the Difference Between Correlated Proportions or Percentages. *Psychometrika*, 12(2):153–157.

Marie Mikulová, Milan Straka, Jan Štěpánek, Barbora Štěpánková, and Jan Hajic. 2022. Quality and efficiency of manual annotation: Pre-annotation bias. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2909–2918, Marseille, France. European Language Resources Association.

Keith Ord. 2004. Charles Holt's report on exponentially weighted moving averages: an introduction and appreciation. *International Journal of Forecasting*, 20(1):1–3.

Nicholas Pangakis and Sam Wolken. 2024. Knowledge distillation in automated annotation: Supervised text classification with LLM-generated training labels. In *Proceedings of the Sixth Workshop on Natural Language Processing and Computational Social Science (NLP+CSS 2024)*, pages 113–131, Mexico City, Mexico. Association for Computational Linguistics.

Kseniia Petukhova and Ekaterina Kochmar. 2025. A Fully Automated Pipeline for Conversational Discourse Annotation: Tree Scheme Generation and Labeling with Large Language Models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 15829–15852, Vienna, Austria. Association for Computational Linguistics.

Ariana Sahitaj, Premtim Sahitaj, Veronika Solopova, Jiaao Li, Sebastian Möller, and Vera Schmitt. 2025. Hybrid Annotation for Propaganda Detection: Integrating LLM Pre-Annotations with Human Intelligence. In *Proceedings of the Fourth Workshop on NLP for Positive Impact (NLP4PI)*, pages 215–228, Vienna, Austria. Association for Computational Linguistics.

Holger Schramm, Nicole Liebers, Laurenz Biniak, and Franca Dettmar. 2024. Research trends on parasocial interactions and relationships with media characters. A review of 281 English and German-language studies from 2016 to 2020. *Frontiers in Psychology*, 15. Publisher: Frontiers.

Yu-Min Tseng, Wei-Lin Chen, Chung-Chi Chen, and Hsin-Hsi Chen. 2025. Evaluating Large Language Models as Expert Annotators. *arXiv preprint*. ArXiv:2508.07827 [cs] version: 1.

Riva Tukachinsky, Nathan Walter, and Camille J Saucier. 2020. Antecedents and effects of parasocial relationships: A meta-analysis. *Journal of Communication*, 70(6):868–894.

Guoqing Wang, Zeyu Sun, Zhihao Gong, Sixiang Ye, Yizhou Chen, Yifan Zhao, Qingyuan Liang, and Dan Hao. 2024. Do Advanced Language Models Eliminate the Need for Prompt Engineering in Software Engineering? *arXiv preprint*. ArXiv:2411.02093 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Li Yang, Qifan Wang, Jianfeng Chi, Jiahao Liu, Jingang Wang, Fuli Feng, Zenglin Xu, Yi Fang, Lifu Huang, and Dongfang Liu. 2024. EAVE: Efficient product attribute value extraction via lightweight sparse-layer interaction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1491–1505, Miami, Florida, USA. Association for Computational Linguistics.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Xianyang Zhan, Agam Goyal, Yilun Chen, Eshwar Chandrasekharan, and Koustuv Saha. 2025. SLM-mod: Small language models surpass LLMs at content moderation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8774–8790, Albuquerque, New Mexico. Association for Computational Linguistics.

Kaiyi Zhang, Ang Lv, Yuhan Chen, Hansen Ha, Tao Xu, and Rui Yan. 2024. Batch-ICL: Effective, Efficient, and Order-Agnostic In-Context Learning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10728–10739, Bangkok, Thailand. Association for Computational Linguistics.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A Survey on Efficient Inference for Large Language Models. *arXiv preprint*. ArXiv:2404.14294 [cs] version: 2.

Olena Zimba and Armen Yuri Gasparyan. 2021. Social media platforms: a primer for researchers. *Reumatologia*, 59(2):68–72.

Appendices

A Semi-active Learning Diagnostics

This appendix elaborates the diagnostic semi-active learning procedure used to identify systematic disagreement regions and thereby enables the context-aware design. We used GPT-4.1 for this task because of the intent to optimize temperature and fine-tune the model for subsequent evaluations, which more up-to-date models, such as GPT-5.1, were not available at the time of our experiments.

A.1 Pool-based Query-by-committee

With an input of an unlabeled pool $\mathcal{U}$ of messages in a stream, this procedure samples a working set and obtains a *committee* of S stochastic LLM annotations for each message (e.g., via nonzero temperature). Each stochastic run returns a full multi-label object over fields (aspects) $\mathcal{F} = \{\textit{DirectAddress}, \textit{Attachment}, \ldots, \textit{Dimension}\}$, and a short free-text `Notes` field that can be used for auditing but excluded from uncertainty scoring.

A.2 Uncertainty via Normalized Vote Entropy

For message x and field $f \in \mathcal{F} \setminus \{\texttt{Notes}\}$, let $\{\hat{y}_f^{(s)}(x)\}_{s=1}^{S}$ be committee votes and let $p_f(v)$ denote the vote share for label value v. Define normalized vote entropy Let $V_f(x) = \{v : p_f(v) > 0\}$.

$$H_f(x) = \begin{cases} 0, & \text{if } |V_f(x)| \le 1, \\ \widetilde{H}_f(x), & \text{otherwise,} \end{cases} \tag{1}$$

$$\widetilde{H}_f(x) = \frac{-\sum_{v \in V_f(x)} p_f(v) \log\big(p_f(v) + \epsilon\big)}{\log |V_f(x)| + \epsilon} \tag{2}$$

A small ϵ is included for numerical stability. Consistent with the described implementation, a message-level uncertainty score is computed as the maximum entropy over a core subset of fields:

$$U(x) = \max_{f \in F_{\text{core}}} H_f(x),$$
$$F_{\text{core}} = \{\textit{DirectAddress}, \textit{Attachment}, \textit{SelfDisclose}, \textit{ReplySeeking}, \textit{Dimension}\}. \tag{3}$$

so $U(x) \in [0, 1]$.

A.3 Majority Label and Acquisition Score

Let $\hat{y}^{\text{maj}}(x)$ be the per-field majority vote, and let $m_f(x)$ denote the majority fraction for field f. To prioritize cases that are *reliability-critical* even when the committee agrees, the procedure includes bounded penalty checks $\Pi(x) \in [0, 1]$ as task-specific sanity rules to deal with invalid categorical values, missing or empty notes (penalized), or metadata/label contradictions. The acquisition score is calculated as

$$A(x) \;=\; \min\{1,\; U(x) + \lambda\Pi(x)\}, \qquad (4)$$

with a penalty weight $\lambda \geq 0$.

A.4 Review Set Selection Within Each Batch

Within each processed batch of size K, the procedure selects a human-review subset under a fixed review budget $\rho \in (0, 1)$:

$$B_{\text{review}} \;=\; B_{\text{must}} \;\cup\; \text{Top}_{\lceil \rho K \rceil}\big(\{A(x)\}_{x \in B}\big), \qquad (5)$$

where B_{must} is an override set induced obtained from hard safety rules; for example, if the committee is insufficiently decisive on *Dimension*, operationalized as $m_{\textit{Dimension}}(x)$ below a threshold, or if insufficient committee samples are successfully parsed for x, item review is enforced. All items not in B_{review} are auto-accepted using $\hat{y}^{\text{maj}}(x)$. The implementation further enforces a minimum auto-accept fraction by trimming the lowest-$A(x)$ non-mandatory items from B_{review} when necessary. The resulting borderline set then is processed for (i) qualitative error analysis and (ii) optional targeted fine-tuning on boundary cases.

Table 4 reports per-label uncertainty statistics over the borderline pool (n=405) after deduplication. For each message x and label field f, we use the committee outputs and mark f as uncertain if there is any non-unanimity (vote entropy $H_f(x) > 0$ or max-vote $\text{MV}_f(x) < 1$); thus $n_{\text{unc}}(f)$ counts how many messages are uncertain on f (messages may be uncertain for multiple fields). We also report the mean H_f and mean MV_f averaged across all n messages. Because these diagnostic analyses capture disagreement-based uncertainty, low uncertainty (high MV_f) does not imply a label is annotated easily and correctly, because systematic confident errors may not be routed into the uncertain set. Moreover, the review set is manually reviewed for model diagnostics.

Table 4: Semi-active learning uncertainty summary for constructing the borderline set (n=405). Columns: n=items, n_{unc}=uncertain items, unc%=uncertainty rate, H=mean entropy, MV=mean max-vote.

Label	n	n_{unc}	unc%	H	MV
Attachment	405	148	36.5%	0.332	0.872
DirectAddress	405	105	25.9%	0.226	0.917
Dimension	405	99	24.4%	0.221	0.915
ReplySeeking	405	80	19.8%	0.176	0.933
SelfDisclose	405	69	17.0%	0.152	0.943
Backseat	405	46	11.4%	0.093	0.968
Emotes	405	37	9.1%	0.075	0.974
CommRitual	405	25	6.2%	0.052	0.982
Monetary	405	9	2.2%	0.017	0.995

A.5 Use Case and Fine-Tuning

Beyond diagnostics, we use the selected borderline messages (items routed to human review due to high $A(x)$ or hard overrides) as a focused supervision set for targeted adaptation. We match the reviewed items with ground-truth labels and notes, then fine-tune the same base model used to generate the committee on this high-disagreement subset, which is viewed as boundary-case training data.

Next, we evaluate the fine-tuned model on (i) the remaining messages not used for fine-tuning (disjoint holdout) and (ii) two additional benchmark settings reported in Appendix G. Tables 9–11 present the improvements gained through fine-tuning. This design minimizes train-test leakage by ensuring that all evaluation items are excluded from the fine-tuning pool, such that measured gains revealt generalization beyond the audited borderline set.

B Adaptive Prefix Batching for Sequential LLM Annotation Under Usability and Throughput Constraints

B.1 Setting

We consider a sequential stream of messages, with a hard operational contiguity requirement to preserve local discourse context (e.g., coreference, addressee, topic continuity). Accordingly, each API call processes a contiguous prefix window of size K starting at the current pointer position. The decision variable is the scalar batch size K, not an item-wise selection vector.

Token amortization is the objective of batching. That is, a fixed per-call prompt overhead (instructions, schema, shared context) is shared across items to reduce the number of input tokens per item and thereby increase efficiency, as inference cost/latency are strongly driven by token usage.

B.2 Token Feasibility and Amortization

Let: C (context window), $\alpha \in (0,1)$ (safety discount), M_{out} (reserved max output tokens), B_{safe} (reserved output buffer), H (fixed per-call overhead tokens), s_i (token length of message i after formatting), and h (per-item wrapper overhead tokens).

Define the available input budget:

$$C_{\text{in}} \;=\; \alpha C \;-\; M_{\text{out}} \;-\; B_{\text{safe}}. \tag{6}$$

Define the input tokens for a batch of size K starting at pointer i_t:

$$T_{\text{in}}(K) \;=\; H \;+\; \sum_{j=0}^{K-1} \big(s_{i_t+j} + h\big). \tag{7}$$

Hard feasibility constraint that must always hold:

$$T_{\text{in}}(K) \;\le\; C_{\text{in}}. \tag{8}$$

Amortization identity (explicit batching gain):

$$\frac{T_{\text{in}}(K)}{K} \;=\; \frac{H}{K} \;+\; \frac{1}{K} \sum_{j=0}^{K-1} \big(s_{i_t+j} + h\big), \tag{9}$$

so an increase in K strictly decreases the fixed-overhead term H/K, which facilitates large K subject to operational safety.

B.3 Pipeline-observable "Quality" and "Feasibility" Without Ground Truth

New streams lack ground-truth labels, so the controller relies only on pipeline-observable proxies.

Usability (primary). Let F_t be a hard failure indicator for attempt t:

$$F_t \;=\; \begin{cases} 1 & \text{if the response is unusable (e.g., JSON/schema failure, truncation to invalid structure, provider error if implemented)} \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

This is the primary reliability signal.

Internal validity / coherence (secondary, only when usable). If $F_t = 0$, we obtain per-item outputs $y_{t,1}, \ldots, y_{t,K_t}$. Let $V(\cdot)$ be a deterministic validator (schema checks + construct-logic constraints). Define:

$$Y_t = \frac{1}{K_t} \sum_{j=0}^{K_t-1} V(y_{t,j}) \in [0,1], \quad \text{only observed when } F_t = 0. \tag{11}$$

This avoids assigning arbitrary "coherence" values when the batch is unusable.

Output-pressure / truncation risk (binary). Let P_t denote the observed batch output pressure at attempt t (an alternative for latency L_t which would be another optimization goal), defined as the ratio of realized output tokens to the reserved output budget M_{out}:

$$P_t \;=\; \frac{\text{Tok}_t^{\text{out}}}{M_{\text{out}}} \tag{12}$$

Let $\pi_{\text{out}} \in (0,1)$ be a hyperparameter pressure threshold and let $\text{Trunc}_t \in \{0,1\}$ indicate whether the provider truncated the response due to output-length limits. We define a binary operational-risk event:

$$T_t \;=\; \mathbb{I}[P_t \ge \pi_{\text{out}} \;\vee\; \text{Trunc}_t = 1] \;\in\; \{0,1\}. \tag{13}$$

Guidance for choosing π_{out}:

$$\pi_{\text{out}} \;=\; \min\{\pi_{\max},\; p_{95}(P \mid K = K_0)\}\,, \tag{14}$$

where $p_{95}(\cdot)$ is computed from the cold-start phase under a small fixed K_0, and $\pi_{\max} \in (0,1)$ is a conservative ceiling (e.g., 0.7–0.9) that preserves headroom against truncation.

B.4 Bounded, Normalized Health Loss (Single Scalar)

Define a bounded health loss:

$$g_t = \frac{w_F F_t + w_T T_t + w_V (1 - Y_t)(1 - F_t)}{w_F + w_T + w_V} \in [0,1]. \tag{15}$$

Key clarifications:

- *Coherence optional* When $w_V = 0$, the coherence term vanishes and g_t is dependent on hard failures and timeouts only. In our experiments, the controller operates in a hard-signal mode, so Y_t is computed as a diagnostic indicator.
- *No double counting and no selection bias* The coherence term is multiplied by $(1 - F_t)$, so coherence is only "taxed" when observable (usable batch).

B.5 EWMA Health Monitor (Updated on Every Attempt)

Maintain an EWMA (Ord, 2004) of health:

$$\bar{g}_t \;=\; \beta \bar{g}_{t-1} + (1 - \beta) g_t, \quad \beta \in (0,1). \tag{16}$$

EWMA gets updated in every attempt, including retries (reattempts).

B.6 Batch-size Controller: AIMD With Hard Overrides and Cooldown

Let K_t be the batch size selected for the next attempt at pointer i_t. Define: Δ (additive increase step), γ (multiplicative decrease factor), n_{cool} (cooldown length), $K_{\min}$ (minimum batch size), and τ (target EWMA threshold). Let c_t be a cooldown counter. We perform forward greedy search to compute $K_{\text{token}}(i_t)$.

Hard feasibility cap:

$$K_{\text{token}}(i_t) = \max\{K \le K_{MAX} : \ T_{\text{in}}(K) \le C_{\text{in}}\}. \tag{17}$$

Update rule (AIMD (Chiu and Jain, 1989), with hard overrides and cooldown):

$$\begin{aligned}
\textbf{if } (F_t{=}1 \vee T_t{=}1) &: K \leftarrow \max\{K_{\min}, \lfloor \gamma K \rfloor\} \\
&\quad c \leftarrow n_{\text{cool}} \\
\textbf{else if } (c > 0) &: c \leftarrow c - 1 \\
\textbf{else if } (\bar{g}_t \le \tau) &: K \leftarrow K + \Delta \\
\textbf{else } (\bar{g}_t > \tau) &: K \leftarrow \max\{K_{\min}, \lfloor \gamma K \rfloor\} \\
&\quad c \leftarrow n_{\text{cool}} \\
\textbf{finally} &: K \leftarrow \min\{K, \ K_{\text{token}}(i_t)\}.
\end{aligned} \tag{18}$$

B.7 Cold Start

We process early data with a small, fixed K_0 for a short warmup of B_0 batches to estimate robust message-length statistics and baseline latency. A typical length is estimated by the median of observed s_i from warmup:

$$s \ = \ \text{median}\{s_i : \ i \text{ appeared in warmup}\}. \tag{19}$$

A conservative fraction of the available budget is applied for initialization, starting ρ_0 at 20-50% capacity, which provides headroom for long outliers and formatting expansions.

$$K_{\text{init}} = \min\left\{K_{\text{token}}(i_0), \ \left\lfloor \rho_0 \frac{C_{\text{in}} - H}{s + h} \right\rfloor\right\}, \tag{20}$$

$\rho_0 \in [0.2, 0.5]$.

B.8 Failure Handling and Retry Semantics

At pointer i_t, we attempt to annotate a contiguous block starting at i_t with size K. If $F = 0$, accept outputs and advance pointer by K: $i_{t+1} = i_t + K$. If $F = 1$, do not advance pointer; compute the hard-event backoff in Eq. (18) and retry the same starting position with the smaller K. Retry up to $r_{\max} \in \{1, 2\}$ times to avoid aggressive and expensive retry loops. That is, we seek to shrink the batch. If still failing at $K_{\min}$, fall back to $K = 1$ with the same controller state or skip and log, depending on strictness. After each attempt (including a retry), compute g_t using Eq. (15) and update $\bar{g}_t$ in Eq. (16).

B.9 Implementation Summary

The pseudo code 1 details the overall algorithm for dynamic K.

Algorithm 1 Adaptive Prefix Batching controller at pointer i (assumes K initialized from cold start or previous iteration)

Compute $K_{\text{token}}(i)$ via Eq. (17)
$K \leftarrow \min\{K, \ K_{\text{token}}(i)\}$
for $a = 1, \ldots, r_{\max} + 1$ **do**
  Execute batch on messages $m_i, \ldots, m_{i+K-1}$
  Observe F_t, $\text{Tok}_t^{\text{out}}$, and Trunc_t
  Compute $P_t \leftarrow \text{Tok}_t^{\text{out}} / M_{\text{out}}$ via Eq. (12)
  If $F_t = 0$, compute Y via Eq. (11) (else Y unobserved)
  $T_t \leftarrow \mathbb{I}[P_t \ge \pi_{\text{out}} \ \vee \ \text{Trunc}_t = 1]$ via Eq. (13)
  Compute g via Eq. (15); update $\bar{g}$ via Eq. (16)
  if $F = 0$ **then**
    Accept; $i \leftarrow i + K$; **break**
  else
    Apply hard-event backoff/cooldown update and feasibility cap (Eq. (18))
    Retry same pointer i with reduced K
  end if
end for

B.10 Hyperparameters

The hyperparameters and implementation instantiation used for dynamic K are presented in Table 5. We chose the hyperparameters conservatively through a series of empirical analyses to minimize fluctuations and errors, as well as reducing truncation failures and schema violations. Important hyperparameters were tuned for each use case to achieve the best results and optimize performance.

Table 5: Implemented Controller Hyperparameters

Hyperparameter	Value
C	400,000
α	0.10
M_{out}	4,000
B_{safe}	1,500
$K_{\max}$	60
K_0	20
B_0	10
ρ_0	0.30
β	0.90
Δ	1
γ	0.70
n_{cool}	3
$K_{\min}$	1
τ	0.10
$r_{\max}$	2
(w_F, w_T, w_V)	(10, 3, 0)
π_{out}	0.90

B.11 Remarks on design choices

The described method makes no claim of global optimality. Rather, it the design of a low-overhead

adaptive controller for black-box, stochastic LLM APIs. The health loss is normalized and bounded; timeouts are binary; and coherence is optional and only evaluated when observable. This controller responds immediately to hard failures or timeouts, using cooldown to attenuate oscillations. All metrics are pipeline-observable; that is, no human review and no ground truth are required.

C Proposed Method Prompt

We provide the prompt used for the zero-shot, schema-guided, context-augmented annotation condition. The prompt specifies the output schema, decision rules, two-layer context usage, and contrastive demonstrations for obtaining PSI labels. The complete prompt text, along with the streamer context gathering prompt, is provided here: Prompt.

All code (including benchmarks) and data are publicly available on GitHub (the link has been provided in the abstract).

D Data

We collected data from the official Twitch API and selected 2,520 real-time user chat messages sample from a de-identified representative dataset that contains more than 20 million chat messages. We used this sample for the intended proof of concept, annotating messages for downstream PSI identification. Table 6 presents some descriptive statistics of the sample.

E Context Gains

To further illustrate the aggregate results, Table 7 provides illustrative cases where adding context helps resolve ambiguities using the same prompt. It is noted that the prompt fails to disambiguate without context. As shown, the context-enabled variant (in our proposed pipeline) produces correct annotations, but the no-context benchmark (DomainPrompt) systematically err. This pattern reaffirms that lightweight retrieval/context can offer informative, diagnostic cues and thereby improves labeling accuracy measured by inferred intent (e.g., *CommRitual*, *ReplySeeking*) or stance (*Dimension*) rather than purely surface form.

F McNemar and Bowker Tests

For each binary label, we applied McNemar's (McNemar, 1947) exact test on paired outcomes (model vs. human annotation), where the 2×2 contingency table is $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and b, c are discordant counts. Interpreting this test in our setting: if $p < 0.05$, there is evidence that the model's (marginal) positive rate differs from human annotated labels suggesting a systematic tendency to over- or under-call a focal label; if $p \geq 0.05$, there is no evidence of a marginal-rate difference, depending on the direction of the test), though disagreements may still exist yet appear more balanced rather than directionally biased. The observed p-values thus can complement per-label F1/Recall/Accuracy by indicating whether a method's errors are either *asymmetric* (and hence potentially correctable via calibration or thresholding) or broadly symmetric noises.

This adaptation is intuitive because the same items are labeled by both systems in the form of paired observations. That is, tests operate directly on *within-item* disagreements instead of treating model and human annotation outcomes as independent samples. Such tests are common in psychology and social sciences research.

For paired nominal outcomes that have more than two categories, Bowker's test of symmetry (Bowker, 1948), a multi-class analogue of McNemar), must be applied to support multilabel spaces beyond a binary one.

Table 6: Compact one-column summary of annotations and distribution.

Label Statistics, N = 2520			
Label	**Y**	**Not-Y**	**%Y**
DirectAddress	1821	699	72.3%
Attachment	879	1641	34.9%
SelfDisclose	346	2174	13.7%
ReplySeeking	404	2116	16.0%
CommRitual	221	2299	8.8%
Monetary	227	2293	9.0%
Backseat	377	2143	15.0%
Emotes	656	1864	26.0%
Dimension (P/N), N = 2520			
P (Positive Stance)	2329		92.4%
N (Negative Stance)	191		7.6%
Messages per Streamer, N = 2520			
Streamer	**Count**		**% of Total**
Emiru	847		33.6%
Ironmouse	695		27.6%
Hasanabi	355		14.1%
Pokimane	284		11.3%
Valkyrae	197		7.8%
Denims	88		3.5%
Fuslie	54		2.1%

G Detailed Evaluation Results

This appendix provides detailed evaluation results, according to the quality and deployment-relevant

Table 7: Examples where context enables correct labeling. In all cases, proposed pipeline matches the human annotation but the benchmark without context errs.

Streamer	raw_msg	Label	Correct Annotation	CaBSALLM	DomainPrompt
ironmouse	Take care Mousie and everyone ♡	CommRitual	Y	Y	_
emiru	I'm mad that emiru has to carry those apes	Dimension	P	P	N
ironmouse	How are you lately?	DirectAddress	Y	Y	_
emiru	LOL do you even give a shit about the awards	ReplySeeking	Y	Y	_
pokimane	gonna sub with prime in 2 days	SelfDisclose	Y	Y	_

criteria. In addition to common performance measures (e.g., accuracy recall, precision, F-1), we also examined efficiency (tokens, time, cost) and agreement (the paired agreement tests in Appendix F). Because the overarching goal is efficiency gain, we compared the proposed efficiency-centric pipeline and benchmarks in terms of time and cost efficiency as well as annotation agreement, beyond annotation quality.

G.1 Benchmark Families and Rationale

For comprehensiveness and robustness, we evaluated the proposed pipeline in comparison with 20 benchmarks pertaining to (i) baselines for LLM annotation reliability, (ii) advanced, sophisticated prompting, (iii) partial proposed pipeline derivations (including fixed batching schedules and model variants), and (iv) prevalent models and their splits for robustness analysis (FULL vs. HOLDOUT; GPT-5.1 vs. GPT-4.1 family). Where feasible, prompts were kept as identical or highly similar across pipelines, using the same schema, label definitions, and decoding rules, and only altered when required by the benchmark design (e.g., adding rationale fields for explain-then-annotate, or committee/adjudication wiring for collaborative baselines).

Literature-based baselines We considered three important approaches for improving LLM annotation reliability at inference time from the extant literature:

- **LiaHR (label rectification).** We implemented Label-in-a-Haystack Rectification, in line with (Chochlakis et al., 2025). Starting from candidate labels produced by our DOMAINPROMPT baseline, the queried item and its candidate label-set were incorporated as a randomly placed "haystack" demonstration in a disjoint demo pool. Next, the model was asked to relabel the item from scratch, with disagreements being treated as rectification signals under the LiaHR policy (with originals preserved for the subsequent auditing). Demonstration pools were disjoint from evaluation splits to avoid leakage.

- **Event Extraction-style collaboration (committee + selective adjudication).** We implemented a multi-annotator voting-and-adjudication baseline that mimics collaborative dataset construction in event extraction (Liu et al., 2025). A heterogeneous committee produced independent schema-constrained label sets, and the majority vote yielded provisional labels for disagreement analysis. High-disagreement items were then selectively escalated to a stronger adjudicator model for final labels. This baseline operationalizes reliability through model diversity and targeted escalation. We adopted different models in the OpenAI family to adapt this benchmark instead of using multiple LLMs.

- **AnnoLLM-style prompting (explain–then–annotate).** We applied the two-stage explain–then–annotate procedure (He et al., 2024) to multi-label Twitch schema. Stage 1 generates short rationales for active labels, based on gold demonstrations (cached for reproducibility). Stage 2 incorporates rationalized demonstrations in the prompt and annotates each item with strict JSON-only decoding, in alignment with AnnoLLM's principle of using self-generated explanations to strengthen few-shot prompting while keeping outputs structured.

Prompting paradigms (complexity sweep) Recent research shows diminishing returns of complex prompting strategies for state-of-the-art LLMs, which motivaties the use of complementary levers beyond prompt engineering (Wang et al., 2024; Khan, 2025). We investigated this empirically and compared pipelines that rely on prompt-only variants using minimal context to isolate prompting effects: zero-shot prompting (Kojima et al., 2023), few-shot prompting (Brown et al., 2020), chain-of-

Table 8: Benchmark families used to contextualize the proposed pipeline.

Family	Representative methods	What it isolates	Why included
Prompting-only	SimplePrompt: ZeroShot, FewShot, CoT, ToT	Generic prompting complexity without PSI-specific grounding or discourse-aware batching	Provides a lower-complexity baseline and tests whether prompt engineering alone is sufficient.
Domain-grounded single-item	DomainPrompt; DomainPrompt+CCoT	PSI-specific schema grounding and demonstrations without adaptive batching	Separates domain grounding effects from batching/controller effects.
Fixed-K batching ablations	CaBSALLM-FixedK=5; CaBSALLM-FixedK=70	Static batching under conservative versus liberal operating points	Shows whether a fixed batch schedule can replace adaptive prefix control.
Existing reliability-oriented pipelines	LiaHR Adaptation; AnnoLLM Adaptation; EE Adaptation	Reliability-focused multi-step or collaborative inference pipelines	Compares against stronger reliability-based literature-inspired pipelines.
Holdout fine-tuning comparator	DomainPrompt-FT	Boundary-case supervision through targeted fine-tuning	Compares fine-tuning on borderline data against inference-time context and batching.
Proposed method	CaBSALLM	Two-layer context + adaptive prefix batching	The intended deployment-oriented operating point.

thought prompting (Wei et al., 2023), contrastive chain-of-thought prompting (Chia et al., 2023), and tree-of-thought prompting (Yao et al., 2023). These benchmarks represent "SimplePrompt" variants to contextualize the incremental value of domain grounding and context-aware batching.

G.2 Complete benchmark list

To complement the benchmark-family summary reported in the main text, Table 8 provides the full benchmark inventory used throughout the evaluation. The table expands each family into its concrete implementations, including the proposed pipeline and ablations, domain-grounded prompting variants, prompt-only complexity sweeps, literature-inspired reliability-oriented adaptations, and the holdout/fine-tuning comparator. This organization is intended to make clear that the evaluation is not a narrow one-to-one comparison, but a structured landscape spanning alternative sources of performance gains: prompt design, context grounding, batching policy, collaborative reliability mechanisms, model choice, and targeted adaptation. Unless otherwise noted, results are reported on the FULL setting; HOLDOUT and fine-tuned variants are marked explicitly.

G.3 The appendix tables

Table 9 presents efficiency comparison results (FULL and HOLDOUT), in terms of aggregated runtime, token, and spend logs. Aggregate quality metrics averaged over the nine labels are reported in Table 10. Fine-grained reliability and per-label behavior are shown in two complementary ways: Table 11 stacks per-label metrics (F1, recall, precision, accuracy) alongside paired-test p-values on HOLDOUT; Tables 12 and 13 provide the corresponding FULL per-label matrices, including paired-test diagnostics. Appendix F provides interpretations of McNemar/Bowker tests and their role as reliability diagnostics. Tables 9–11 also include the fine-tuned model produced through the semi-active learning protocol in Appendix A.

G.4 Holistic Evaluation

In line with the holistic evaluation described in the main text, we considered large-scale annotations an important deployment issue and emphasize that methods should be assessed in a five-way trade-off holistically: (i) aggregate and per-label effectiveness, (ii) cost efficiency, (iii) time efficiency (Aguda et al., 2024), (iv) real-time deployment operability (Zhan et al., 2025), and (v) consistency with human annotations (Lee et al., 2025). Pipelines that marginally improves the average F1 score but significantly increases inference cost or exhibits stronger directional disagreement may be less suitable for large-scale social-science research than those that preserve comparable quality while improving throughput and agreement. Accordingly, tables in this appendix intend to convey such trade-offs more explicitly. All efficiency metrics (mUSD/msg, Tokin/msg, Tokout/msg, Sec/msg) are computed from the same execution code that issues the API calls, timing end-to-end wall-clock runtime per request (including batching and any retries), and token costs are priced using OpenAI's official rates, distinguishing input, cached input, and output tokens.

Table 9: Efficiency per message on FULL (N=2520) and HOLDOUT (N=2115). Values are computed from total runtime, tokens, and spend logs. (Best in class marked in bold)

Set	Method	Model	Tok$_{in}$/msg	Tok$_{out}$/msg	Sec/msg	Cost (mUSD/msg)
FULL	CaBSALLM	5.1	193	117	1.29	1.274
FULL	LiaHR Adaptation	5.1	4,497.0	152.5	3.11	2.91
FULL	AnnoLLM Adaptation	5.1	9394	666	11.96	8.23
FULL	EE Adaptation	MIX	9,881.9	231.4	5.36	5.17
FULL	CaBSALLM (CCoT)	5.1	208	112	1.156	1.222
FULL	CaBSALLM-FixedK=70	5.1	**80**	85	**0.802**	0.916
FULL	CaBSALLM-FixedK=5	5.1	558	127	1.776	1.450
FULL	CaBSALLM-5-mini	5-mini	217	118	1.473	**0.265**
FULL	DomainPrompt (CCoT)	5.1	2226	133	2.33	1.67
FULL	DomainPrompt	5.1	1903	150	3.01	1.88
FULL	SimplePrompt ToT	5.1	849	183	2.50	2.89
FULL	SimplePrompt CoT	5.1	764	141	2.10	2.36
FULL	SimplePrompt FewShot	5.1	595	155	2.89	2.30
FULL	SimplePrompt ZeroShot	5.1	360	155	2.23	2.00
FULL	DomainPrompt (CCoT)-4.1	4.1	2227	43	0.99	1.54
FULL	DomainPrompt-4.1	4.1	2001	**41**	1.15	2.57
FULL	SimplePrompt FewShot-4.1	4.1	639	69	1.23	1.83
HOLDOUT	DomainPrompt	5.1	1999	56	1.97	0.90
HOLDOUT	DomainPrompt-4.1	4.1	2000	**41**	1.14	2.56
HOLDOUT	DomainPrompt-FT	4.1-FT	2000	56	1.59	6.05

Table 10: Aggregated mean metrics (averaged over the nine labels) for FULL (N=2520) and HOLDOUT (N=2115) evaluations. (Best in class marked in bold)

Set	Method	Model	Mean-F1	Mean-Recall	Mean-Precision	Mean-Accuracy
FULL	CaBSALLM	5.1	0.866	0.851	0.884	**0.949**
FULL	LiaHR Adaptation	5.1	0.850	0.856	0.859	0.942
FULL	AnnoLLM Adaptation	5.1	0.867	0.857	0.881	0.948
FULL	EE Adaptation	MIX	0.840	0.827	0.861	0.937
FULL	CaBSALLM (CCoT)	5.1	0.863	0.851	0.879	0.948
FULL	CaBSALLM-FixedK=70	5.1	0.848	0.856	0.841	0.937
FULL	CaBSALLM-FixedK=5	5.1	0.861	0.848	0.878	0.947
FULL	CaBSALLM-5-mini	5-mini	0.680	0.684	0.695	0.866
FULL	DomainPrompt (CCoT)	5.1	0.848	0.822	0.885	0.944
FULL	DomainPrompt	5.1	0.849	0.826	0.882	0.943
FULL	SimplePrompt ToT	5.1	0.744	0.763	0.778	0.904
FULL	SimplePrompt CoT	5.1	0.776	0.791	0.785	0.909
FULL	SimplePrompt FewShot	5.1	0.786	0.822	0.766	0.909
FULL	SimplePrompt ZeroShot	5.1	0.775	0.823	0.745	0.903
FULL	DomainPrompt (Pool AL)-4.1	4.1	0.825	0.776	**0.895**	0.930
FULL	DomainPrompt (CCoT)-4.1	4.1	0.834	0.812	0.879	0.934
FULL	DomainPrompt-4.1	4.1	0.847	0.841	0.859	0.935
FULL	SimplePrompt FewShot-4.1	4.1	0.796	0.796	0.804	0.910
HOLDOUT	DomainPrompt	5.1	0.837	0.796	0.886	0.933
HOLDOUT	DomainPrompt-4.1	4.1	0.853	0.845	0.865	0.937
HOLDOUT	DomainPrompt-FT	4.1-FT	**0.871**	**0.883**	0.865	0.945

Table 11: HOLDOUT (N=2115): Per-label metrics with explicit names. Each cell stacks F1, Recall (R), Precision (P), Accuracy (Acc), and McNemar exact p. Labels: DA=DirectAddress, Att=Attachment, SD=SelfDisclose, RS=ReplySeeking, CR=CommRitual, Mon=Monetary, Back=Backseat, Emo=Emotes, Dim=Dimension. (Best in labels and classes marked in bold)

Method	Model	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim
DomainPrompt Holdout	5.1	F1=0.919 R=0.894 **P=0.946** Acc=0.887 **p=5.5e-8**	F1=0.796 R=0.736 **P=0.866** Acc=0.874 p=6.8e-11	F1=0.838 R=0.847 P=0.829 Acc=0.957 **p=0.602**	F1=0.783 R=0.715 **P=0.865** Acc=0.936 p=4.0e-7	F1=0.579 R=0.502 P=0.682 Acc=0.929 p=1.2e-5	**F1=1.000** **R=1.000** **P=1.000** **Acc=1.000** **p=1**	F1=0.735 R=0.650 **P=0.846** **Acc=0.929** p=1.2e-9	F1=0.920 R=0.886 P=0.958 Acc=0.959 p=6.5e-6	F1=0.960 R=0.936 P=0.986 Acc=0.928 p=6.0e-17
DomainPrompt Holdout-4.1	4.1	F1=0.938 **R=0.980** P=0.899 Acc=0.906 p=9.8e-24	F1=0.806 R=0.772 P=0.843 Acc=0.876 p=2.5e-4	F1=0.833 R=0.904 P=0.772 Acc=0.952 p=2.1e-6	F1=0.851 R=0.859 P=0.844 Acc=0.952 **p=0.621**	F1=0.629 R=0.566 P=0.707 Acc=0.935 **p=5.8e-4**	F1=0.997 R=0.995 **P=1.000** **Acc=1.000** **p=1**	F1=0.737 R=0.709 P=0.767 Acc=0.923 **p=0.070**	F1=0.928 R=0.896 **P=0.962** Acc=0.963 p=2.0e-5	F1=0.959 R=0.928 **P=0.992** Acc=0.927 p=1.3e-26
DomainPrompt-FT Holdout	4.1-FT	**F1=0.954** R=0.978 P=0.930 **Acc=0.932** p=4.6e-11	F1=0.813 **R=0.780** P=0.849 **Acc=0.880** **p=4.1e-4**	**F1=0.877** **R=0.911** **P=0.845** **Acc=0.966** p=0.013	**F1=0.881** **R=0.947** P=0.824 **Acc=0.959** p=3.3e-8	**F1=0.663** **R=0.595** **P=0.748** **Acc=0.941** p=2.0e-4	**F1=1.000** **R=1.000** **P=1.000** **Acc=1.000** **p=1**	**F1=0.744** **R=0.856** P=0.657 Acc=0.911 p=8.9e-13	**F1=0.931** **R=0.914** P=0.948 **Acc=0.964** **p=0.029**	**F1=0.976** **R=0.967** P=0.985 **Acc=0.956** **p=2.6e-4**

Table 12: FULL (N=2520): Per-label matrices for F-1 and Accuracy. Labels: DA=DirectAddress, Att=Attachment, SD=SelfDisclose, RS=ReplySeeking, CR=CommRitual, Mon=Monetary, Back=Backseat, Emo=Emotes, Dim=Dimension. (Best in labels marked in bold)

(a) F1

Method	Model	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim
CaBSALLM	5.1	0.962	0.832	0.844	0.848	0.611	0.996	0.784	0.929	**0.980**
LiaHR Adaptation	5.1	0.954	0.847	0.820	0.874	0.488	0.991	0.777	**0.940**	0.953
AnnoLLM Adaptation	5.1	0.950	**0.853**	0.854	**0.878**	0.602	0.995	0.754	**0.940**	0.971
EE Adaptation	MIX	0.933	0.810	0.817	0.845	0.547	**0.998**	0.735	0.935	0.940
CaBSALLM (CCoT)	5.1	0.956	0.846	0.856	0.862	0.574	0.996	0.777	0.927	0.977
CaBSALLM-FixedK=70	5.1	**0.963**	0.803	0.810	0.835	0.590	0.991	**0.787**	0.898	0.971
CaBSALLM-FixedK=5	5.1	**0.963**	0.832	**0.872**	0.863	0.563	0.996	0.764	0.922	0.974
CaBSALLM-5-mini	5-mini	0.880	0.671	0.580	0.603	0.427	0.856	0.524	0.647	0.938
DomainPrompt (CCoT)	5.1	0.949	0.844	0.853	0.857	0.523	0.991	0.720	0.928	0.968
DomainPrompt	5.1	0.951	0.839	0.844	0.853	0.515	0.993	0.747	0.939	0.961
SimplePrompt ToT	5.1	0.928	0.634	0.643	0.824	0.246	0.940	0.619	0.920	0.940
SimplePrompt CoT	5.1	0.926	0.778	0.686	0.851	0.377	0.897	0.663	0.882	0.921
SimplePrompt FewShot	5.1	0.920	0.809	0.731	0.854	0.377	0.906	0.677	0.889	0.910
SimplePrompt ZeroShot	5.1	0.922	0.796	0.713	0.835	0.375	0.901	0.641	0.877	0.914
DomainPrompt (Pool AL)-4.1	4.1	0.949	0.684	0.845	0.827	0.595	0.996	0.730	0.840	0.958
DomainPrompt (CCoT)-4.1	4.1	0.943	0.814	0.784	0.827	0.525	**0.998**	0.743	0.927	0.948
DomainPrompt-4.1	4.1	0.934	0.816	0.804	0.836	**0.626**	**0.998**	0.731	0.927	0.955
SimplePrompt FewShot-4.1	4.1	0.926	0.772	0.793	0.849	0.487	0.910	0.652	0.883	0.897

(b) Accuracy

Method	Model	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim
CaBSALLM	5.1	0.945	0.883	0.958	0.953	0.941	0.999	**0.938**	0.963	**0.963**
LiaHR Adaptation	5.1	0.933	0.890	0.943	0.957	0.931	0.998	0.937	**0.968**	0.917
AnnoLLM Adaptation	5.1	0.928	**0.900**	0.959	**0.959**	0.935	0.999	0.936	**0.968**	0.949
EE Adaptation	MIX	0.901	0.875	0.945	0.947	0.933	**1.000**	0.927	0.967	0.895
CaBSALLM (CCoT)	5.1	0.936	0.892	0.960	0.956	0.933	**1.000**	0.936	0.963	0.957
CaBSALLM-FixedK=70	5.1	0.926	0.855	0.947	0.944	0.930	0.998	0.934	0.949	0.947
CaBSALLM-FixedK=5	5.1	**0.947**	0.882	**0.965**	0.956	0.933	0.999	0.933	0.952	0.955
CaBSALLM-5-mini	5-mini	0.826	0.735	0.890	0.866	0.920	0.970	0.879	0.823	0.889
DomainPrompt (CCoT)	5.1	0.927	0.894	0.959	0.955	0.931	0.998	0.928	0.963	0.941
DomainPrompt	5.1	0.929	0.891	0.956	0.954	0.931	0.999	0.933	**0.968**	0.930
SimplePrompt ToT	5.1	0.891	0.803	0.866	0.938	0.915	0.988	0.882	0.958	0.894
SimplePrompt CoT	5.1	0.889	0.858	0.885	0.948	0.913	0.979	0.906	0.942	0.863
SimplePrompt FewShot	5.1	0.879	0.860	0.910	0.948	0.910	0.981	0.904	0.941	0.847
SimplePrompt ZeroShot	5.1	0.882	0.840	0.898	0.941	0.904	0.980	0.895	0.935	0.852
DomainPrompt (Pool AL)-4.1	4.1	0.924	0.829	0.958	0.947	0.940	0.999	0.921	0.927	0.925
DomainPrompt (CCoT)-4.1	4.1	0.916	0.878	0.933	0.945	0.939	**1.000**	0.928	0.963	0.908
DomainPrompt-4.1	4.1	0.901	0.876	0.941	0.947	**0.942**	**1.000**	0.922	0.963	0.919
SimplePrompt FewShot-4.1	4.1	0.890	0.855	0.942	0.947	0.909	0.982	0.896	0.943	0.827

Table 13: FULL (N=2520): Per-label matrices for Precision, Recall, and McNemar exact p. Labels: DA=DirectAddress, Att=Attachment, SD=SelfDisclose, RS=ReplySeeking, CR=CommRitual, Mon=Monetary, Back=Backseat, Emo=Emotes, Dim=Dimension. (Best in labels marked in bold)

(a) Precision

Method	Shots	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim
CaBSALLM	5.1	0.961	0.827	0.860	**0.878**	0.717	0.991	0.814	0.924	0.981
LiaHR Adaptation	5.1	0.940	0.827	0.729	0.816	0.697	0.982	0.818	0.927	0.992
AnnoLLM Adaptation	5.1	0.950	0.873	**0.874**	0.838	0.649	0.991	**0.883**	0.921	0.984
EE Adaptation	MIX	0.910	0.620	0.862	0.802	0.671	0.995	0.800	0.961	0.992
CaBSALLM (CCoT)	5.1	**0.966**	0.837	0.851	0.869	0.653	0.991	0.808	0.950	0.979
CaBSALLM-FixedK=70	5.1	0.957	0.764	0.802	0.792	0.608	0.982	0.765	0.931	0.972
CaBSALLM-FixedK=5	5.1	0.962	0.826	0.868	0.868	0.657	0.991	0.818	0.934	0.978
CaBSALLM-5-mini	5-mini	0.880	0.591	0.610	0.574	0.577	0.751	0.636	0.672	0.968
DomainPrompt (CCoT)	5.1	0.948	0.869	0.842	**0.878**	0.669	0.983	0.857	0.927	0.988
DomainPrompt	5.1	0.945	0.864	0.830	0.867	0.664	0.991	0.851	0.934	0.990
SimplePrompt ToT	5.1	0.885	0.900	0.507	0.758	0.556	0.887	0.598	0.919	0.990
SimplePrompt CoT	5.1	0.896	0.856	0.548	0.782	0.512	0.814	0.718	0.946	0.992
SimplePrompt FewShot	5.1	0.876	0.770	0.618	0.773	0.476	0.828	0.681	0.876	**0.993**
SimplePrompt ZeroShot	5.1	0.877	0.717	0.581	0.757	0.435	0.819	0.657	0.868	**0.993**
DomainPrompt (Pool AL)-4.1	4.1	0.953	**0.957**	0.846	0.869	0.730	0.991	0.750	**0.972**	0.989
DomainPrompt (CCoT)-4.1	4.1	0.929	0.866	0.703	0.838	**0.825**	**1.000**	0.799	0.959	**0.993**
DomainPrompt-4.1	4.1	0.896	0.844	0.737	0.826	0.722	**1.000**	0.758	0.958	0.989
SimplePrompt FewShot-4.1	4.1	0.908	0.852	0.785	0.780	0.480	0.835	0.655	0.949	**0.993**

(b) Recall

Method	Model	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim
CaBSALLM	5.1	0.962	0.838	0.832	0.819	0.538	1.000	0.756	0.933	**0.979**
LiaHR Adaptation	5.1	0.970	0.867	**0.939**	0.943	0.375	**1.000**	0.740	0.952	0.918
AnnoLLM Adaptation	5.1	0.951	0.834	0.872	0.923	0.561	**1.000**	0.658	**0.957**	0.960
EE Adaptation	MIX	0.958	0.763	0.887	0.894	0.461	**1.000**	0.679	0.910	0.893
CaBSALLM (CCoT)	5.1	0.945	0.856	0.861	0.856	0.511	1.000	0.748	0.904	0.974
CaBSALLM-FixedK=70	5.1	0.940	0.846	0.818	0.884	**0.575**	1.000	**0.812**	0.867	0.970
CaBSALLM-FixedK=5	5.1	0.965	0.840	0.876	0.859	0.493	1.000	0.716	0.910	0.970
CaBSALLM-5-mini	5-mini	0.879	0.777	0.552	0.636	0.339	0.996	0.446	0.623	0.909
DomainPrompt (CCoT)	5.1	0.951	0.821	0.864	0.837	0.430	**1.000**	0.621	0.930	0.948
DomainPrompt	5.1	0.958	0.816	0.858	0.839	0.421	0.996	0.666	0.945	0.933
SimplePrompt ToT	5.1	**0.976**	0.489	0.882	0.901	0.158	**1.000**	0.642	0.921	0.895
SimplePrompt CoT	5.1	0.958	0.712	0.916	0.933	0.299	**1.000**	0.615	0.826	0.859
SimplePrompt FewShot	5.1	0.970	0.851	0.896	**0.953**	0.312	**1.000**	0.674	0.902	0.840
SimplePrompt ZeroShot	5.1	0.973	**0.895**	0.922	0.931	0.330	**1.000**	0.626	0.886	0.846
DomainPrompt (Pool AL)-4.1	4.1	0.940	0.532	0.844	0.790	0.502	**1.000**	0.711	0.739	0.930
DomainPrompt (CCoT)-4.1	4.1	0.957	0.768	0.887	0.817	0.385	0.996	0.695	0.896	0.907
DomainPrompt-4.1	4.1	**0.976**	0.790	0.884	0.847	0.552	0.996	0.706	0.898	0.924
SimplePrompt FewShot-4.1	4.1	0.945	0.706	0.801	0.931	0.493	**1.000**	0.650	0.825	0.818

(c) McNemar Exact P Values for each Lable (+p), with total number of agreements among all labels and ranking regarding total agreements between all benchmarks

Method	Model	DA	Att	SD	RS	CR	Mon	Back	Emo	Dim	Pass	Rank
CaBSALLM	5.1	**1.000**	**0.523**	0.285	0.016	7.9e-5	0.500	0.038	0.606	**0.757**	6	2
LiaHR Adaptation	5.1	5.81e-6	0.013	2.82e-18	9.6e-10	02.8e-15	0.125	0.005	0.057	1.06e-38	2	11
AnnoLLM Adaptation	5.1	0.824	0.014	0.167	6.6e-5	0.023	0.500	1.3e-14	0.001	2e-7	3	8
EE Adaptation	MIX	7.21e-10	1.33e-8	4.87e-7	7.7e-5	1.14e-7	**1.000**	3.36e-5	1e-4	1.21e-54	1	17
CaBSALLM (CCoT)	5.1	0.001954	0.275947	0.764353	0.633756	2.63652e-4	0.500000	0.033572	0.001261	0.289798	5	3
CaBSALLM-FixedK=70	5.1	0.022768	7.54e-7	0.603046	9.33e-5	0.407087	0.125000	0.086460	9.21e-5	0.665950	4	6
CaBSALLM-FixedK=5	5.1	0.665950	0.416629	0.832258	0.775003	2.81e-5	0.500000	3.42407e-4	0.110918	0.120328	**7**	**1**
CaBSALLM-5-mini	5-mini	0.961897	3.48e-27	0.054322	0.019207	1.05e-10	2.04e-21	8.75e-11	0.029462	6.67e-18	2	11
DomainPrompt (CCoT)	5.1	0.769	0.004	0.431	0.090	1.6e-9	0.125	3.7e-15	0.918	2.1e-15	5	3
DomainPrompt	5.1	0.061	0.004	0.294	0.267	7.3e-10	**1.000**	2.3e-10	0.434	7.9e-26	5	3
SimplePrompt ToT	5.1	2.7e-32	1.1e-82	4.8e-49	8.9e-10	8.2e-30	3.7e-9	0.118	**1.000**	1.5e-48	2	11
SimplePrompt CoT	5.1	4.3e-14	3.2e-15	7.7e-48	4.7e-12	3.9e-10	4.4e-16	5.3e-4	2.3e-12	7.6e-77	0	18
SimplePrompt FewShot	5.1	4.6e-31	1.2e-6	6.4e-27	1.8e-17	5.4e-7	1.4e-14	0.847	0.118	1.4e-92	2	11
SimplePrompt ZeroShot	5.1	1.1e-32	6.8e-29	2.5e-41	5.6e-15	8.2e-4	1.8e-15	0.295	0.347	9.8e-88	2	11
DomainPrompt (Pool AL)-4.1	4.1	0.071	5e-95	**1.000**	0.002	1.8e-8	0.500	0.177	1.7e-35	7.87e-27	4	6
DomainPrompt (CCoT)-4.1	4.1	2.5e-4	1.3e-8	1.2e-12	**0.444**	1.3e-23	**1.000**	3.3e-4	9.4e-6	5.8e-48	2	11
DomainPrompt-4.1	4.1	6.4e-27	0.002	1.4e-8	0.437	2.0e-5	**1.000**	0.074	2.5e-5	1.2e-29	3	8
SimplePrompt FewShot-4.1	4.1	9.9e-6	2.8e-15	0.618	7.1e-12	**0.742**	5.7e-14	**0.902**	2.6e-13	5.0e-106	3	8

Full Developer Prompt for Annotation in the Proposed Pipeline and Streamer Global Context Prompt

```
# ----------------------------
# Annotation Developer Prompt
# ----------------------------
DEVELOPER_PROMPT = r"""
You are an annotation model. Your task is to label each Twitch chat message with a set of
↪  binary/ternary fields using the provided schema.
Follow definitions and decision rules exactly. Return ONLY a JSON array (no wrapper object, no text).

OUTPUT FORMAT (STRICT)
Return ONLY a JSON array. Each element must include keys: Each array element corresponds to one input
↪  message and must contain all required keys:
DirectAddress, Attachment, SelfDisclose, ReplySeeking, CommRitual, Monetary, Backseat, Emotes,
↪  Dimension (strings).

Each label field must be one of:
"Y" = present
""  = not present / leave blank

For Dimension, use exactly one of:
"P" if positive/affiliative stance toward streamer/community
"N" if negative/hostile stance toward streamer/community
No neutral/mixed category. If mixed, label based on the dominant intent ("punchline").

WHAT YOU ARE ANNOTATING
Annotate message-level expressed cues in livestream chat from Twitch. Multi-label is allowed.
Do not infer hidden intent beyond the text and immediate conversational form.

TWO-LAYER CONTEXT YOU MUST USE
You will receive:
1) STREAMER_GENERAL_CONTEXT: background about the streamer (name, common nicknames, friends,
↪  recurring memes, controversies, locations, typical content, etc.).
   Use it only to resolve reference/ambiguity (e.g., who "you" refers to; whether a named person is
   ↪  the streamer vs. a friend).
2) LOCAL_BATCH_CONTEXT: what is being discussed in this particular batch (topic/game/event, current
↪  narrative, who is speaking to whom).
   Use it to disambiguate pronouns and whether a question is aimed at the streamer vs. the chat and
   ↪  the topic.

If the general + local contexts still do not make the addressee clear, prefer leaving categories blank
↪  (high precision).

Before labeling individual messages, quickly scan the entire INPUT_MESSAGES batch to infer
↪  LOCAL_BATCH_CONTEXT signals (topic, who "you" refers to, whether chatters are arguing, and
↪  whether questions target streamer vs chat), then annotate each message.

CATEGORY DECISION RULES
A) Dimension (valence)
Set Dimension independently from other categories.
P (Positive): praise/support, gratitude, affection, encouragement, admiration; playful teasing that
↪  is clearly supportive.
N (Negative): insults, hostility, shaming, threats, aggressive blame; hate-watch enjoyment of failure.
Mixed: choose dominant intent.

B) PSI/PSR-leaning categories (multi-label)
1) Direct Address / second-person to streamer -> DirectAddress
Mark "Y" only when the message targets the streamer as an addressee:
- explicit @streamer
- streamer name used as addressee ("Emiru, . . . ")
- second-person pronouns clearly referring to streamer not another person ("you/your/u") with
↪  directed statement/command
Avoid false positives:
- talking about streamer: "emiru is cracked"
- addressing chat: "you guys. . . "
- ambiguous "you" -> blank unless strong cue (name/@mention or clear local context).

Important Point! You should keep an eye on morphs, euphemisms, and varying forms of chatting most
↪  common for Gen-Z, like "ya", "yo", "u" etc. to better understand adressees.
```

```
2) Attachment / affection / relational warmth -> Attachment
Mark "Y" for warmth/care/pride/missing/love toward streamer or relational closeness markers.
Attachment-strengtheners (no extra labels; just annotate Attachment when applicable):
- longitudinal persistence beyond exposure: off-stream anticipation, absence felt, “all week,” “can’t
↪  wait,” “glad you’re back”
- investment/loyalty talk: “watch every stream,” “since 2018,” “never miss a VOD,” schedule organized
↪  around streams
- social realism / real-life tie framing: “you’re like family,” “real friend,” “like a brother/sister”
- anthropomorphism/personification: “you care about us,” “she understands me,” “he worries about chat”
Avoid false positives:
- “love this game” (not streamer-directed)
- generic hype alone (“Pog”, “Lets go”) unless clearly relational.

3) Identification / self-disclosure / homophily -> SelfDisclose
Mark "Y" when viewer shares personal info or similarity claims aimed at relational connection:
- “I’m also. . . ”, “same here. . . ”, “as a fellow. . . ”, "I also do that..."
- personal disclosure positioned as sharing-with-streamer (“I had a rough day, thanks for being here”)
- “I have an exam tomorrow, wish me luck” can count if framed as a bid for connection (even without
↪  explicit “you” if local context makes it streamer-directed).
Strengthener (no new label):
- knowledge/narrative recall is not SelfDisclose by itself; use it to support
↪  DirectAddress/Attachment/ReplySeeking if it functions that way.

4) Reciprocity bids / reply-seeking -> ReplySeeking
Mark "Y" when seeking acknowledgment/response/recognition:
- “notice me,” “read this,” “answer please,” “remember me,” “did you see my last message,” “thanks for
↪  replying last week,” "why you not seeing me"
- Or asking them to do something ordinary like drinking water or blink for safety of eyes.
Avoid false positives:
- generic info questions not aimed at streamer (“what game is this?”) unless streamer-directed by
↪  name/@mention or clear local context.

5) Community ritual talk & co-creation -> CommRitual
Mark "Y" for shared rituals/inside jokes/coordinated norms tied to community:
- “spam LUL,” “copy pasta time,” “chat do the thing,” “only real ones remember. . . ,” “as always. . . ,”
↪  "Let's go..."
Avoid false positives:
- emotes alone is not ritual talk without ritual framing.

6) Monetary support actions -> Monetary
Mark "Y" for subs/donos/gifts/streaks or accompanying text:
- “subbed X months,” “gifted subs,” “dono,” “renewed,” “superchat”
Hard rule: If the message tag is "USERNOTICE" instead of "PRIVATEMESSAGE" ALWAYS mark Monetary="Y".
Avoid false positives:
- unrelated money talk (“costs 60 bucks”).

ADDED CATEGORIES (behavioral / surface form)
7) Backseat guidance -> Backseat
Mark "Y" for gameplay/stream/content direction/coaching/strategy/idea/suggestion:
- “go left,” “use ult,” “buy armor,” “skip cutscene,” “watch out. . . ,” "You have to change it in the
↪  menu"
Or general questions asking regarding the topic:
- "Hassan are they capitalists?", "you should do a comparison on their technical data"
Backseat can co-occur with DirectAddress, but backseating is not parasocial by itself.

8) Emotes/emoji presence -> Emotes
Mark "Y" if any Twitch-style emote tokens or unicode emojis appear, including <3, and tokens like
↪  Kappa, pogchamp, KEKW, etc.

CONTRASTIVE DEMONSTRATIONS (BEHAVIORAL TARGETS)
Follow the demos for common pitfalls:
Demo A — DirectAddress vs about-talk; “you” ambiguity
General: emiru is the streamer; “chat” = audience. Local: streamer just returned; greetings happening.
Inputs:
(1) "Everyone missed you beautiful Emiru <3 <3 WutFace"
(2) "emiru is cracked today"
(3) "you guys missed her too right? lol"
Correct:
(1) DirectAddress=Y (name + “you”), Attachment=Y (“missed you”, warmth), Emotes=Y, Dimension=P
(2) About-talk only: DirectAddress blank; Dimension=P (praise)
(3) Targets chat (“you guys”): DirectAddress blank; Dimension=P (light/affiliative)
```

```
Avoid:
- Don’t set DirectAddress just because streamer name appears (2)
- Don’t set DirectAddress from “you” when it clearly addresses chat (3)

Demo B — Loyalty/investment as Attachment; recall is not SelfDisclose; mixed valence punchline rule
General: hasanabi political streams; “since YEAR” indicates tenure.
Inputs:
(10) "Been watching since 2019. Never miss a VOD."
(11) "Remember that 2019 meltdown? downhill ever since lol"
(12) "I love you but you’re being an idiot tonight"
Correct:
(10) Attachment=Y (investment/loyalty), SelfDisclose=Y, Dimension=P
(11) Dimension=N (criticism); SelfDisclose blank (recall is not personal disclosure)
(12) Attachment=Y (“love you”), Dimension=N (dominant thrust is insult/punchline)
Avoid:
- Don’t force SelfDisclose from narrative recall (11)
- Don’t label Dimension=P just because affection phrase exists (12)

Demo C — USERNOTICE monetary override + co-occurrence
Rule: msgTag="USERNOTICE" => Monetary=Y ALWAYS.
Input:
(20) msgTag=USERNOTICE, "Renewed my sub for 24 months — love you Emiru <3"
Correct:
Monetary=Y (forced), Attachment=Y, DirectAddress=Y (name used as addressee), Emotes=Y (<3),
↪  Dimension=P
Avoid:
- Never leave Monetary blank on USERNOTICE even if message is mostly affection

Return ONLY a JSON array, no text, no code fences. Use the provided id values exactly as given in
↪  INPUT_MESSAGES (0..K-1). Do not renumber or reorder items.
"""

annotation_schema = {
    "name": "psi_annotation_batch",
    "schema": {
        "type": "object",
        "additionalProperties": False,
        "properties": {
            "items": {
                "type": "array",
                "items": {
                    "type": "object",
                    "additionalProperties": False,
                    "properties": {
                        "id": {"type": "integer"},
                        "DirectAddress": {"type": "string"},
                        "Attachment": {"type": "string"},
                        "SelfDisclose": {"type": "string"},
                        "ReplySeeking": {"type": "string"},
                        "CommRitual": {"type": "string"},
                        "Monetary": {"type": "string"},
                        "Backseat": {"type": "string"},
                        "Emotes": {"type": "string"},
                        "Dimension": {"type": "string"}
                    },
                   "required": ["id","DirectAddress","Attachment","SelfDisclose","ReplySeeking","Com⌋
                    ↪  mRitual","Monetary","Backseat","Emotes","Dimension"]
                }
            }
        },
        "required": ["items"]
    }
}

# ------------------------------
#Streamer profile (RAG via web_search) -> saved once per streamer
# ------------------------------

profile_schema = {
        "name": "streamer_profile",
        "schema": {
            "type": "object",
```

```
            "additionalProperties": False,
            "properties": {
                "streamer": {"type": "string"},
                "Name": {"type": "string"},
                "Nicknames": {"type": "string"},
                "Location": {"type": "string"},
                "Affiliation": {"type": "string"},
                "Friends": {"type": "string"},
                "Interests": {"type": "string"},
                "ContentStyle": {"type": "string"},
                "Controversies": {"type": "string"},
                "Summary500": {"type": "string"}
            },
            "required": ["streamer", "Summary500"]
        }
    }

prompt = f"""
You are building a compact streamer profile for downstream Twitch-chat annotation.
Search the web as needed. Return JSON ONLY matching the schema.

Target streamer handle: {streamer}

Fill (short fragments; if uncertain -> ""):
Name, Nicknames, Location, Affiliation, Friends, Interests, ContentStyle, Controversies

Summary500: a single compact summary <= 500 characters.
Be conservative: do not guess.
"""
```