# SCBO: SEMANTICALLY COHERENT BATCHING AND ORDERING FOR LLM-BASED SOCIAL SURVEYS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large Language Models (LLMs) offer a scalable way to simulate survey respondents conditioned on demographic profiles and observed reference responses. However, the conventional one-question-per-prompt paradigm is limited in three respects: it repeatedly encodes the same context, incurring substantial token and inference costs; it restricts each target to its own narrow subset of observed responses, preventing reference evidence from being shared across targets; and it predicts every answer in isolation, preventing later predictions from leveraging information in earlier answers. To address these limitations, we predict multiple questions in a single prompt, which amortizes the shared context, allows multiple targets to share a broader pool of observed reference responses, and enables later predictions to condition on earlier ones. However, such a method faces two challenges: (1) how to form semantically coherent batches and select shared references, and (2) how to order questions and references to improve autoregressive generation. We propose Semantically Coherent Batching and Ordering (SCBO), a training-free framework that addresses these challenges through two modules, preceded by a Question Instantiation step in which an LLM extracts compact semantic representations from survey items to filter out template noise. (1) Semantic Batching and Selection: SCBO groups related questions into semantic batches and constructs a shared reference bank by combining target-specific retrieval with centroid-based completion. (2) Curriculum-based Ordering: It then applies a heuristic easy-to-hard ordering to target questions and orders references within the shared bank according to their semantic alignment with the ordered questions. Experiments on four large-scale survey datasets and four LLMs show that SCBO substantially reduces token consumption and inference time while generally improving prediction accuracy over the non-batched baseline. Code is available at https://anonymous.4open.science/r/SCBO-41D8.

## 1 INTRODUCTION

Social surveys, which ask people directly about their attitudes, values, and political preferences, are a foundational tool in the social sciences for tracking public opinion and evaluating policy (Groves et al., 2011; Roopa & Rani, 2012; Tourangeau et al., 2000; Squazzoni et al., 2020; Hamill & Gilbert, 2009; Wang et al., 2025). However, asking real people is hard: respondents are costly to recruit, may not answer honestly, and often refuse the most revealing questions, such as income or political extremes (Wright et al., 2010; Heffetz & Reeves, 2019; Kalton, 2009). As a result, traditional surveys remain limited in scale, reliability, and coverage.

Large Language Models (LLMs) (Achiam et al., 2023; Schaeffer et al., 2023; Kosinski, 2023) offer an opportunity to alleviate these difficulties. Trained on vast amounts of human-written Internet text, which contains traces of how people express attitudes, judgments, and preferences, LLMs acquire rich prior knowledge of human attitudes and behavior (Argyle et al., 2023; Park et al., 2023; Aher et al., 2023; Cao et al., 2023; Zhou et al., 2025; Durmus et al., 2023). Recent work therefore uses LLMs as synthetic respondents: given a respondent's demographic profile and reference responses, a model can predict their responses to other survey questions, extending survey scale and coverage at a fraction of the effort (Santurkar et al., 2023; Simmons, 2023; Bisbee et al., 2024).



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Table 1: Estimated token usage and API cost for exhaustive one-question-per-prompt prediction. Prices are based on the public rates listed at https://aiapi.world/pricing.

| Dataset | Users | Questions | (User, Question) | Avg. tokens | | Token budget (M) | | Estimated cost (USD) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Context | Question | Input | Output | GPT-6 | Fable-5 | Gemini-3.8 |
| WVS | 97,220 | 285 | 25,038,918 | 491.1 | 70.6 | 14,064 | 125 | **$11,126+** | **$58,552+** | **$11,017+** |
| ANES | 5,521 | 952 | 3,541,672 | 551.4 | 62.9 | 2,176 | 18 | $1,714+ | $9,026+ | $1,698+ |
| GSS | 3,986 | 539 | 1,001,083 | 539.5 | 55.1 | 595 | 5 | $470+ | $2,473+ | $465+ |
| BSA | 4,120 | 251 | 531,064 | 509.5 | 63.2 | 304 | 3 | $240+ | $1,265+ | $238+ |

While LLM-based simulation relieves the burden of collecting answers from people, the prevailing "one-question-per-prompt" practice (Argyle et al., 2023; Santurkar et al., 2023; Bisbee et al., 2024) is limited in three respects. *(1) It is expensive:* every prompt repeats the task instruction, the respondent's demographic profile, and a few of that respondent's observed answers as reference examples, so this shared context is re-encoded once per target question. As shown in Table 1, exhaustive prediction involves 25.0 million user-question pairs for WVS and 3.5 million for ANES, requiring 14.06B and 2.18B input tokens; for WVS alone, this lower-bound estimate costs at least $11,126 on GPT-6-Astra, $58,552 on Claude-Fable-5, and $11,017 on Gemini-3.8-Flash. *(2) It prevents reference sharing:* each target question independently retrieves $n$ references. When targets are predicted separately, references retrieved for one target are unavailable to the others, so each prediction uses only a small portion of the respondent's observed responses. *(3) It discards information:* answering each question in a prompt of its own treats a respondent's answers as mutually independent, whereas items probing the same underlying attitude are strongly correlated, and the answer to one question shapes the answer to the next. Each prediction is thus made without access to the answers that would inform it most.

We propose to batch target questions into a prompt: the shared context is encoded once per batch rather than once per question, questions within the same batch can share a larger pool of observed reference responses under the same context budget, and later predictions can condition on answers generated earlier in the batch. However, answering questions together makes each prediction depend on which questions accompany it and in what order, giving rise to two challenges: ***Challenge 1: What to put in a batch.*** All questions in a batch share one set of references, and that set has a limited budget. If unrelated questions are grouped together, say one about trust in government and one about family values, examples that suit one are of little use to the other, and most questions end up with no useful reference. Batches therefore have to be formed so that a single set of examples can serve every question in them, and the examples have to be chosen to cover the whole batch rather than any one question. ***Challenge 2: How to order a batch.*** An LLM generates batched answers autoregressively, so later predictions can condition on answers generated earlier, but not vice versa. If an earlier question is predicted correctly, its answer can provide useful evidence for subsequent predictions. For example, a correct prediction of religious importance may help predict a respondent's view on same-sex marriage. Questions should therefore be ordered so that reliable earlier predictions support harder ones that follow, with references ordered to align semantically with the target sequence.

To address these challenges, we propose a **Semantically Coherent Batching and Ordering (SCBO) Framework** that maximizes accuracy while reducing token consumption and inference time via two modules. *(1) Semantic Batching and Selection:* To address Challenge 1, we utilize fixed-size clustering to group cohesive questions. Within each batch, we retrieve target-specific references into a shared reference bank and fill remaining slots with centroid-based retrieval when budget allows. *(2) Curriculum-based Ordering:* To address Challenge 2, we exploit the autoregressive dependency among batched predictions and draw inspiration from curriculum learning. We place easier questions earlier, where their predictions are more likely to provide reliable context for harder questions that follow, and arrange references to align with the target sequence.

Our contributions are threefold: ❶ We formulate the LLM-based survey batching problem, which amortizes respondent context across questions, enables multiple targets to share a broader pool of observed reference responses, and allows later predictions to condition on earlier answers. We identify its two challenges: what to put in a batch and how to order what is in it. ❷ We propose **SCBO**, a training-free framework with two modules: semantic batching, which groups cohesive questions and selects a shared reference bank, and curriculum-based ordering, which arranges questions and references so that earlier predictions inform later ones. ❸ Experiments on four large-scale survey datasets across four LLMs show that SCBO reduces token cost by over 50% and accelerates inference by 3.3–14.0×, while generally improving accuracy over the one-question-per-prompt baseline.

## 2 PROBLEM FORMULATION: LLM-BASED SOCIAL SURVEYS

Let $u$ denote a respondent associated with a textual demographic profile $P_u$ (e.g., age, gender, education, and income). For this respondent we have a reference set $(\mathcal{Q}_{pool}, \mathcal{A}_{pool})$ of survey questions that $u$ has answered before, together with the given answers, which serve as candidate in-context examples. The task is to predict how $u$ would answer a target set $\mathcal{Q}_{target} = \{q_1, \ldots, q_N\}$ of $N$ questions that $u$ has not been asked.

Under the prevailing one-question-per-prompt paradigm, an LLM $\mathcal{M}$ predicts each target answer from its own prompt. Let $\mathrm{card}(\cdot)$ denote the number of examples in a reference set:

$$\hat{a}_i = \mathcal{M}\big(\mathcal{I} \oplus P_u \oplus S_i \oplus q_i\big), \qquad S_i \subset (\mathcal{Q}_{pool}, \mathcal{A}_{pool}), \quad \mathrm{card}(S_i) = n. \tag{1}$$

where $\mathcal{I}$ is the task instruction, $S_i$ is a reference set retrieved for $q_i$ under a per-target-question budget $n$, and $\oplus$ denotes concatenation into a single prompt. Each prediction $\hat{a}_i$ is evaluated against the withheld ground-truth answer $a_i$ in terms of ACC, MAE, and F1 over the $N$ target questions.

Only $S_i$ and $q_i$ in Eq. (1) vary across target questions, whereas $\mathcal{I}$ and $P_u$ are identical for all of them. Let $T_{\text{single}}$ denote the total number of prompt tokens needed to predict all $N$ target answers of one respondent under Eq. (1). Writing $|\cdot|$ for the token count of a prompt component,

$$T_{\text{single}} = \sum_{i=1}^{N} \big(|\mathcal{I}| + |P_u| + |S_i| + |q_i|\big) = \underbrace{N\big(|\mathcal{I}| + |P_u|\big)}_{\text{redundant}} + \sum_{i=1}^{N} \big(|S_i| + |q_i|\big). \tag{2}$$

This formulation is limited in three respects. *Cost:* the first term of Eq. (2) grows linearly in $N$ while carrying no new information, and it dominates in practice, as on WVS the shared context averages $491.1$ tokens per prediction against $70.6$ tokens for the target question itself (Table 1). *History:* under a fixed reference budget,, each target question independently retrieves and uses its reference set $S_i$. Because they are used separately, each target can access only the portion of respondent history contained in its own set, even when references retrieved for other targets could also be informative. *Independence:* Eq. (1) conditions $\hat{a}_i$ only on $q_i$ and the reference set, so the $N$ answers are predicted independently, without conditioning on answers generated for other target questions. Survey responses are not: items that probe the same underlying attitude are strongly correlated, and an answer given to one question shapes the answer given to the next.

To address all three limitations, we let one prompt answer several questions at once, partitioning $\mathcal{Q}_{target}$ into $K = N/C$ disjoint batches $\mathcal{B} = \{B_1, \ldots, B_K\}$ of size $C$ and rewriting Eq. (1) as

$$\{\hat{a}_i\}_{q_i \in B_k} = \mathcal{M}\big(\mathcal{I} \oplus P_u \oplus \boldsymbol{\phi}(S_k) \oplus \boldsymbol{\pi}(B_k)\big), \qquad S_k \subset (\mathcal{Q}_{pool}, \mathcal{A}_{pool}), \quad \mathrm{card}(S_k) = m. \tag{3}$$

where $S_k$ is a reference bank shared by all questions in $B_k$, with a budget of $m = nC$ examples, matching the aggregate reference budget of the $C$ individual target questions. The functions $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$ order the questions and references within the prompt. The corresponding token cost is

$$T_{\text{batch}} = K\big(|\mathcal{I}| + |P_u|\big) + \sum_{k=1}^{K} \Big(|S_k| + \textstyle\sum_{q_i \in B_k} |q_i|\Big). \tag{4}$$

Comparing Eq. (3) with Eq. (1), the shared context $\mathcal{I} \oplus P_u$ now serves $C$ questions instead of one, the questions in each batch share a reference bank $S_k$ rather than using separate per-target sets $S_i$, and the answers of a batch are generated within an autoregressive sequence rather than in separate prompts. This addresses all three limitations. On cost, the dominant term drops from $N$ copies in Eq. (2) to $K = N/C$ in Eq. (4), a reduction by a factor of $C$, while the questions are still encoded exactly once. On history, the shared bank allows each target to access references retrieved for other questions in the same batch, exposing it to a broader range of respondent history under the reference budget $m = nC$. On independence, later answers can condition on answers generated earlier in the batch, so the correlations among a respondent's answers can be exploited rather than ignored.

The core of Eq. (3) therefore lies in three quantities it leaves unspecified: the partition $\mathcal{B}$, the shared bank $S_k$, and the orderings $(\boldsymbol{\pi}, \boldsymbol{\phi})$. These respectively determine which target questions are grouped together, which historical responses are shared within each batch, and how the target questions and references are ordered. All three decide what each answer is conditioned on, and hence the prediction accuracy. The next section specifies how SCBO determines them.

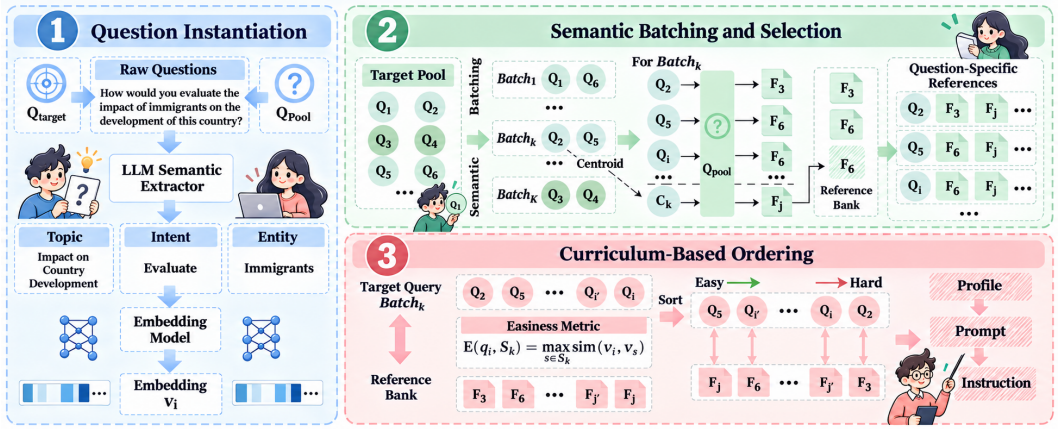



Figure 1: Overview of SCBO. As preprocessing, Question Instantiation extracts compact topic, intent, and entity from survey questions to filter template noise for the two modules: (1) Semantic Batching clusters semantically cohesive questions and builds shared reference banks through target-specific and centroid-based retrieval; and (2) Curriculum-based Ordering sorts target questions from easy to hard and orders references within each shared bank by semantic alignment.

## 3 The SCBO Framework

As illustrated in Figure 1, **SCBO** determines the three quantities that Eq. (3) leaves unspecified. We first map every survey item to a compact semantic representation that strips template noise, a preprocessing step we call Question Instantiation (Section 3.1). Then, operating on these representations, Semantic Batching and Selection fixes the partition $\mathcal{B}$ and the shared reference bank $S_k$, which addresses Challenge 1 (Section 3.2). Finally, Curriculum-based Ordering fixes the orderings $\boldsymbol{\pi}$ and $\boldsymbol{\phi}$, which addresses Challenge 2 (Section 3.3).

### 3.1 Preprocessing: Question Instantiation for Semantic Denoising

Raw survey items often contain redundant conversational templates and standardized instructions that obscure the core semantic intent. To avoid embedding this noise, we extract the semantic essence of each question $q_i \in \mathcal{Q}$ using an LLM $\mathcal{M}$ with a prompt $\mathcal{P}_{extract}$, which distills a representation $\tilde{q}_i = \mathcal{M}(q_i, \mathcal{P}_{extract}) = \{T_i, I_i, E_i\}$, where $T_i$, $I_i$, and $E_i$ denote the Topic, Core Intent, and Key Entities, respectively; the prompt is given in Appendix D.1. For instance, the raw item *"How would you evaluate the impact of immigrants on the development of this country?"* is reduced to $T_i =$ *impact on country development*, $I_i =$ *evaluate*, and $E_i =$ *immigrants*: the respondent-addressing frame *"How would you"*, which recurs across items and contributes little to topical discrimination, is discarded, while the country-level scope expressed by *"of this country"* is retained in the topic representation *impact on country development*. An embedding model encodes each representation as $\mathbf{h}_i = f_{emb}(\tilde{q}_i) \in \mathbb{R}^d$. We then apply L2 normalization, $\mathbf{v}_i = \mathbf{h}_i / \|\mathbf{h}_i\|_2$. Subsequent clustering, retrieval, and ordering operations use these normalized embeddings. Consequently, cosine similarity is equivalent to the dot product, and the squared Euclidean distance between two embeddings satisfies $\|\mathbf{v}_i - \mathbf{v}_j\|_2^2 = 2 - 2\,\mathrm{sim}(\mathbf{v}_i, \mathbf{v}_j)$.

### 3.2 Semantic Batching and Selection

**Semantic Batching.** A batch should group questions that are semantically close, so that one shared reference bank is relevant to all of them and the answers generated within the batch are correlated enough to inform one another. We therefore form batches by clustering the target questions in semantic space. For notational simplicity and experimental convenience, assume $C \mid N$, so that $K = N/C$ and each batch contains $C$ questions. Given the embeddings $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$ of the target questions, let $z_{ik} \in \{0, 1\}$ indicate whether question $q_i$ is assigned to batch $B_k$. We then obtain

the partition $\mathcal{B}$ by minimizing the intra-batch variance:

$$\min_{\mathbf{Z},\,\{\mathbf{c}_k\}_{k=1}^{K}} \sum_{k=1}^{K}\sum_{i=1}^{N} z_{ik}\|\mathbf{v}_i-\mathbf{c}_k\|_2^2 \quad \text{s.t.} \quad \textstyle\sum_{i=1}^{N} z_{ik}=C \;\; \forall k, \qquad \sum_{k=1}^{K} z_{ik}=1 \;\; \forall i, \tag{5}$$

where $\mathbf{c}_k=\frac{1}{C}\sum_{i=1}^{N} z_{ik}\mathbf{v}_i$ is the semantic centroid of batch $B_k$.

**Reference Bank Selection.** For each batch $B_k$, we select the shared reference bank $S_k$ of Eq. (3) from the reference pool, preserving target-specific evidence before sharing it across the batch. Each target question $q_i \in B_k$ first retrieves its own top-$n$ references $R_i^{(n)}=\arg\mathrm{Top}_{n\, s\in\mathcal{Q}_{pool}}\,\mathrm{sim}(\mathbf{v}_i,\mathbf{v}_s)$, where $n$ is the per-question budget and $\mathrm{sim}(\cdot,\cdot)$ is cosine similarity. The bank is then initialized as their union,

$$S_k^{\text{init}}=\bigcup_{q_i\in B_k} R_i^{(n)}, \tag{6}$$

so that each question contributes its most relevant evidence while all questions in the batch have access to the same bank. The total budget is $m=nC$. If $\mathrm{card}(S_k^{\text{init}})<m$ because target-specific references overlap, we fill the remaining slots by repeatedly adding the unselected example closest to the batch centroid, $s^*=\arg\max_{s\in\mathcal{Q}_{pool}\setminus S_k}\mathrm{sim}(\mathbf{v}_s,\mathbf{c}_k)$, until $\mathrm{card}(S_k)=m$ or the pool is exhausted. The resulting bank combines target-specific evidence with references representative of the overall batch semantics. To facilitate reference utilization, we annotate each target question with its relevant examples from the shared bank: for $q_i\in B_k$ we compute a wider retrieval set $R_i=\arg\mathrm{Top}_{r\, s\in\mathcal{Q}_{pool}}\mathrm{sim}(\mathbf{v}_i,\mathbf{v}_s)$ with $r\geq n$ and take $\mathcal{A}_i=S_k\cap R_i$. The prompt presents the full bank $S_k$ once, while each target question is accompanied by its specific subset $\mathcal{A}_i$. This allows a question to leverage references introduced by other semantically related questions in the same batch without increasing the overall bank size.

### 3.3 Curriculum-based Ordering

**Target Question Ordering.** Since each answer is generated conditioned only on what precedes it, we order questions within each batch $B_k$ so that answers produced earlier are the ones best able to inform those produced later. Let $\boldsymbol{\pi}=(\pi_1,\ldots,\pi_C)$ denote a permutation of the question indices in $B_k$, and let $\mathcal{E}(q_i,S_k)=\max_{s\in S_k}\mathrm{sim}(\mathbf{v}_i,\mathbf{v}_s)$ be a heuristic easiness score, since similar references provide more relevant information and make a question easier to answer. Inspired by the curriculum learning paradigm, we apply a training-free heuristic that orders the questions from easy to hard:

$$\boldsymbol{\pi}^{\text{heur}}=\text{argsort}_{i\in B_k}\big(-\mathcal{E}(q_i,S_k)\big), \tag{7}$$

which ensures $\mathcal{E}(q_{\pi_1^{\text{heur}}},S_k)\geq\cdots\geq\mathcal{E}(q_{\pi_C^{\text{heur}}},S_k)$.

**Reference Bank Ordering.** Given the ordered target sequence, we order the references within the shared bank by maximizing their semantic alignment with the target questions:

$$\boldsymbol{\phi}^*=\arg\max_{\boldsymbol{\phi}}\sum_{\ell=1}^{|S_k|}\mathrm{sim}\left(\mathbf{v}_{s_{\phi(\ell)}},\mathbf{v}_{q_{\pi_{g(\ell)}^{\text{heur}}}}\right), \tag{8}$$

where $\boldsymbol{\phi}$ is a permutation over references in $S_k$, and $g(\ell)=\min(C,\lceil\ell/n\rceil)$ maps the $\ell$-th reference slot to the corresponding target question. This places the references most relevant to the earlier target questions first, so that every question is preceded by the evidence it needs. Together, $\boldsymbol{\pi}^{\text{heur}}$ and $\boldsymbol{\phi}^*$ instantiate the orderings of Eq. (3). Appendix D.2 shows how the resulting batches, reference banks, and orderings are serialized into the batch prediction prompt.

## 4 Experiments

In this section, we conduct extensive experiments on four large-scale survey datasets across four LLMs to evaluate SCBO against the non-batched baseline.

**(RQ1)** How effective is SCBO compared with the non-batched baseline?
**(RQ2)** How efficient is SCBO compared with the non-batched baseline?
**(RQ3)** How does batch size affect the accuracy and cost of SCBO?
**(RQ4)** How do components of SCBO affect the performance?

### 4.1 Experimental Setup

**Datasets.** We evaluate our framework on four public opinion survey datasets. World Values Survey (WVS) captures cross-national values and social attitudes; General Social Survey (GSS) measures long-term social trends in the United States; American National Election Studies (ANES) focuses on political behavior and electoral attitudes; and British Social Attitudes Survey (BSA) examines attitudes in the United Kingdom. We use the most recent wave of each dataset. Dataset filtering, respondent-profile construction, and reference–target pool construction are detailed in Appendix C.

**Large Language Models.** We evaluate four widely used LLMs: DeepSeek-V4-Flash, DeepSeek-V4-Pro, Qwen3.7-Max, and GPT-4.1. These models span different capacity levels, allowing us to examine whether our framework consistently improves prediction performance across varying model sizes. For convenience, we use tiktoken to obtain a unified token count across all four models.

**Evaluation Metrics.** We evaluate effectiveness using ACC, MAE, and F1, and efficiency using TPQ and SPQ. ACC measures prediction accuracy, while MAE measures the absolute difference between predicted and ground-truth option indices. F1 is computed separately for each question over its question-specific answer categories and averaged using question sample counts as weights. TPQ and SPQ denote the average prompt tokens and inference time per target question, respectively. Better performance corresponds to higher ACC and F1 and lower MAE, TPQ, and SPQ.

**Evaluation Setting.** The non-batched baseline (w/o) follows the one-question-per-prompt paradigm and corresponds to the degenerate case of SCBO with batch size $C = 1$. Each target question is predicted in its own prompt together with the respondent profile and its retrieved reference examples. We use $x$-budget to denote a setting with per-question reference budget $n = x$ (Section 3.2), i.e., each target question contributes up to $x$ retrieved references. All reported comparisons between the non-batched baseline (w/o) and SCBO (w) use the same reference budget to ensure a fair comparison.

### 4.2 Implementation Details

We use GPT-5.5 to extract semantics from raw survey questions, generating instantiated representations that filter out template noise. We embed the instantiated target and reference questions using `text-embedding-3-small`; reference answers are used as labels and excluded from the embedding input. These embeddings support clustering, retrieval, and ordering operations. Since the non-batched baseline and SCBO share the same instantiation and embedding process, the token costs are excluded from reported efficiency metrics to ensure a fair comparison. For each dataset and reference budget, we select respondents with sufficient question coverage to support batch-size comparisons. Each respondent is assigned a fixed number of target questions, ensuring that under a $k$-budget setting, sufficient reference responses are available for reference selection. While our framework supports arbitrary batch sizes, we focus on divisor-based batch sizes for convenience, as they allow even partitioning without incomplete tail batches. For each dataset–budget setting, we randomly split respondents into 15% validation and 85% test sets. We select the batch size by validation ACC for each dataset–model–budget configuration and report results on the test set. Questions are grouped into fixed-size batches using KMeans++ clustering (Arthur et al., 2007) over instantiated question embeddings, followed by capacity-constrained assignment with the Hungarian algorithm (Kuhn, 1955). The complete capacity-constrained batching procedure is detailed in Appendix B. Within each batch, we set $r = 10$ for the wider target-specific retrieval used for relevant-reference annotation.

### 4.3 Effectiveness of SCBO (RQ1)

Table 2 presents the overall performance of our framework across four datasets, four LLMs, and three reference budgets. Although a few configurations show small declines in ACC or F1, our method (w) generally achieves higher ACC, lower MAE, and higher F1 scores than the non-batched baseline (w/o). Across the 48 dataset–model–budget configurations, SCBO reduces MAE in all configurations, improves ACC in 46, and improves F1 in 45. On WVS, ACC gains reach at least 5% in most configurations, while MAE reductions are observed across all reported configurations. The improvement is particularly notable on BSA, where ACC gains reach up to 12.2% under the 1-budget and 2-budget settings. Similar positive trends are observed across most configurations on GSS and ANES. Moreover, our method improves prediction performance across models with different capability levels, including more capable models such as DeepSeek-V4-Pro and less capable models

Table 2: Effectiveness across reference budgets, LLMs, and datasets. $(w/o)$ denotes the non-batched baseline, $(w)$ denotes SCBO, and $\Delta(\%)$ reports relative change. **Green** indicates improvement, while **gray** indicates degradation.

| LLM | Budget | Setting | WVS | | | GSS | | | ANES | | | BSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ACC | MAE | F1 | ACC | MAE | F1 | ACC | MAE | F1 | ACC | MAE | F1 |
| Deepseek V4-Flash | 1-budget | w/o | 0.506 | 0.928 | 0.493 | 0.491 | 0.742 | 0.478 | 0.508 | 0.917 | 0.492 | 0.486 | 0.713 | 0.476 |
| | | w | 0.543 | 0.798 | 0.516 | 0.517 | 0.678 | 0.488 | 0.531 | 0.861 | 0.510 | 0.545 | 0.607 | 0.533 |
| | | $\Delta(\%)$ | **7.3** | **14.0** | **4.6** | **5.3** | **8.6** | **2.1** | **4.5** | **6.1** | **3.7** | **12.1** | **14.9** | **12.0** |
| | 2-budget | w/o | 0.522 | 0.883 | 0.512 | 0.528 | 0.676 | 0.510 | 0.527 | 0.868 | 0.508 | 0.492 | 0.674 | 0.485 |
| | | w | 0.548 | 0.761 | 0.525 | 0.547 | 0.615 | 0.527 | 0.556 | 0.793 | 0.535 | 0.552 | 0.589 | 0.532 |
| | | $\Delta(\%)$ | **5.0** | **13.8** | **2.6** | **3.6** | **9.0** | **3.3** | **5.5** | **8.6** | **5.3** | **12.2** | **12.6** | **9.8** |
| | 3-budget | w/o | 0.526 | 0.877 | 0.508 | 0.553 | 0.647 | 0.547 | 0.541 | 0.846 | 0.524 | 0.497 | 0.670 | 0.490 |
| | | w | 0.553 | 0.798 | 0.534 | 0.537 | 0.622 | 0.527 | 0.569 | 0.777 | 0.553 | 0.535 | 0.600 | 0.512 |
| | | $\Delta(\%)$ | **5.1** | **9.0** | **5.1** | **2.8** | **3.9** | **3.7** | **5.2** | **8.2** | **5.5** | **7.6** | **10.4** | **4.6** |
| Deepseek V4-Pro | 1-budget | w/o | 0.505 | 0.868 | 0.487 | 0.495 | 0.696 | 0.480 | 0.514 | 0.884 | 0.504 | 0.501 | 0.677 | 0.484 |
| | | w | 0.551 | 0.749 | 0.524 | 0.532 | 0.642 | 0.512 | 0.534 | 0.836 | 0.513 | 0.542 | 0.590 | 0.526 |
| | | $\Delta(\%)$ | **9.1** | **13.7** | **7.6** | **7.5** | **7.8** | **6.6** | **3.9** | **5.4** | **1.7** | **8.2** | **12.9** | **8.6** |
| | 2-budget | w/o | 0.520 | 0.813 | 0.516 | 0.542 | 0.619 | 0.518 | 0.535 | 0.838 | 0.522 | 0.495 | 0.676 | 0.485 |
| | | w | 0.559 | 0.764 | 0.544 | 0.547 | 0.595 | 0.530 | 0.561 | 0.780 | 0.547 | 0.539 | 0.598 | 0.525 |
| | | $\Delta(\%)$ | **7.5** | **6.0** | **5.5** | **0.9** | **3.9** | **2.2** | **4.9** | **6.9** | **4.9** | **8.9** | **11.5** | **8.1** |
| | 3-budget | w/o | 0.533 | 0.817 | 0.518 | 0.552 | 0.622 | 0.541 | 0.556 | 0.792 | 0.548 | 0.498 | 0.707 | 0.482 |
| | | w | 0.567 | 0.746 | 0.552 | 0.569 | 0.581 | 0.559 | 0.586 | 0.731 | 0.569 | 0.537 | 0.588 | 0.524 |
| | | $\Delta(\%)$ | **6.4** | **8.7** | **6.7** | **3.1** | **6.6** | **3.3** | **5.4** | **7.7** | **3.9** | **7.8** | **16.8** | **8.5** |
| Qwen3.7 Max | 1-budget | w/o | 0.509 | 0.845 | 0.479 | 0.509 | 0.681 | 0.487 | 0.506 | 0.888 | 0.487 | 0.500 | 0.644 | 0.477 |
| | | w | 0.544 | 0.749 | 0.512 | 0.533 | 0.623 | 0.498 | 0.524 | 0.843 | 0.502 | 0.543 | 0.576 | 0.521 |
| | | $\Delta(\%)$ | **6.9** | **11.4** | **7.1** | **4.7** | **8.5** | **2.3** | **3.6** | **5.1** | **3.1** | **8.6** | **10.6** | **9.4** |
| | 2-budget | w/o | 0.520 | 0.837 | 0.502 | 0.558 | 0.602 | 0.533 | 0.532 | 0.819 | 0.515 | 0.501 | 0.652 | 0.485 |
| | | w | 0.554 | 0.738 | 0.532 | 0.552 | 0.593 | 0.521 | 0.546 | 0.798 | 0.529 | 0.550 | 0.586 | 0.516 |
| | | $\Delta(\%)$ | **6.5** | **11.8** | **6.1** | **1.1** | **1.5** | **2.3** | **2.6** | **2.6** | **2.9** | **9.8** | **10.1** | **6.2** |
| | 3-budget | w/o | 0.515 | 0.849 | 0.493 | 0.548 | 0.614 | 0.531 | 0.567 | 0.770 | 0.551 | 0.518 | 0.643 | 0.504 |
| | | w | 0.554 | 0.743 | 0.540 | 0.572 | 0.570 | 0.548 | 0.577 | 0.756 | 0.562 | 0.560 | 0.557 | 0.532 |
| | | $\Delta(\%)$ | **7.6** | **12.5** | **9.6** | **4.4** | **7.2** | **3.1** | **1.8** | **1.8** | **2.0** | **8.1** | **13.4** | **5.6** |
| GPT-4.1 | 1-budget | w/o | 0.491 | 0.947 | 0.480 | 0.512 | 0.701 | 0.500 | 0.505 | 0.906 | 0.489 | 0.510 | 0.668 | 0.495 |
| | | w | 0.530 | 0.812 | 0.505 | 0.531 | 0.637 | 0.512 | 0.525 | 0.840 | 0.512 | 0.542 | 0.618 | 0.519 |
| | | $\Delta(\%)$ | **7.9** | **14.3** | **5.1** | **3.7** | **9.1** | **2.3** | **4.0** | **7.3** | **4.8** | **6.3** | **7.5** | **4.9** |
| | 2-budget | w/o | 0.523 | 0.877 | 0.511 | 0.518 | 0.704 | 0.508 | 0.524 | 0.860 | 0.508 | 0.512 | 0.657 | 0.490 |
| | | w | 0.527 | 0.838 | 0.509 | 0.558 | 0.588 | 0.542 | 0.551 | 0.810 | 0.538 | 0.568 | 0.578 | 0.553 |
| | | $\Delta(\%)$ | **0.8** | **4.4** | **0.3** | **7.7** | **16.5** | **6.8** | **5.2** | **5.8** | **5.9** | **10.9** | **12.0** | **12.9** |
| | 3-budget | w/o | 0.519 | 0.893 | 0.505 | 0.533 | 0.680 | 0.525 | 0.548 | 0.826 | 0.534 | 0.542 | 0.640 | 0.517 |
| | | w | 0.543 | 0.836 | 0.528 | 0.556 | 0.597 | 0.543 | 0.584 | 0.747 | 0.571 | 0.560 | 0.567 | 0.533 |
| | | $\Delta(\%)$ | **4.6** | **6.4** | **4.7** | **4.3** | **12.2** | **3.5** | **6.6** | **9.6** | **6.9** | **3.3** | **11.4** | **3.1** |

such as DeepSeek-V4-Flash. This generally positive pattern across diverse survey domains and different LLMs provides evidence of the broad generalizability of our approach, further supporting the effectiveness of our semantic batching and curriculum-based ordering strategies in improving LLM-based social survey prediction performance.

### 4.4 Efficiency of SCBO (RQ2)

Table 3 compares the token cost and inference time of the non-batched baseline and our framework. SCBO consistently reduces both TPQ and SPQ across all evaluated configurations. For token cost, which is our primary optimization objective, the reduction ranges from 30.1% to 71.1%, with an average reduction of approximately 58.0%; reductions exceed 50% in most configurations. Although SCBO primarily targets token efficiency, batching also substantially reduces inference time by amortizing communication latency and other per-request overheads across multiple questions. The observed speedups range from 5.8× to 11.6× for DeepSeek-V4-Flash, 6.6× to 9.1× for DeepSeek-V4-Pro, 7.9× to 14.0× for GPT-4.1, and 3.3× to 10.5× for Qwen3.7-Max. These results indicate that SCBO can improve the efficiency of large-scale LLM-based social survey prediction in terms of both token consumption and inference time in realistic large-scale social survey deployment scenarios.

Table 3: Efficiency across reference budgets, LLMs, and datasets. Columns 1–3 denote the per-question reference budget; $(w/o)$ denotes the non-batched baseline and $(w)$ denotes SCBO. **Green** indicates improvement, while **gray** indicates degradation.

| LLM | Metric | Setting | WVS | | | GSS | | | ANES | | | BSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| DeepSeek V4-Flash | TPQ | w/o | 577 | 661 | 747 | 589 | 659 | 727 | 621 | 698 | 774 | 541 | 613 | 681 |
| | | w | 194 | 280 | 380 | 170 | 253 | 401 | 181 | 257 | 344 | 187 | 247 | 329 |
| | | Redu.(%) | **66.3** | **57.7** | **49.1** | **71.1** | **61.6** | **44.8** | **70.8** | **63.3** | **55.5** | **65.4** | **59.7** | **51.7** |
| | SPQ | w/o | 0.947 | 1.085 | 0.839 | 0.851 | 0.927 | 0.829 | 0.889 | 1.026 | 0.859 | 1.162 | 1.063 | 0.978 |
| | | w | 0.093 | 0.115 | 0.146 | 0.105 | 0.114 | 0.142 | 0.086 | 0.093 | 0.100 | 0.101 | 0.117 | 0.124 |
| | | Speedup | **10.2×** | **9.4×** | **5.8×** | **8.2×** | **8.1×** | **5.8×** | **10.3×** | **11.0×** | **8.6×** | **11.6×** | **9.1×** | **7.9×** |
| DeepSeek V4-Pro | TPQ | w/o | 577 | 661 | 747 | 589 | 659 | 727 | 621 | 698 | 774 | 541 | 613 | 681 |
| | | w | 194 | 280 | 363 | 170 | 278 | 325 | 181 | 257 | 327 | 173 | 268 | 329 |
| | | Redu.(%) | **66.3** | **57.7** | **51.4** | **71.1** | **57.9** | **55.3** | **70.8** | **63.3** | **57.8** | **68.0** | **56.3** | **51.7** |
| | SPQ | w/o | 1.148 | 1.435 | 1.423 | 1.241 | 1.434 | 1.634 | 1.130 | 1.233 | 1.454 | 1.349 | 1.346 | 1.353 |
| | | w | 0.137 | 0.169 | 0.175 | 0.146 | 0.157 | 0.179 | 0.143 | 0.153 | 0.160 | 0.152 | 0.170 | 0.204 |
| | | Speedup | **8.4×** | **8.5×** | **8.2×** | **8.5×** | **9.1×** | **9.1×** | **7.9×** | **8.1×** | **9.1×** | **8.9×** | **7.9×** | **6.6×** |
| Qwen3.7 Max | TPQ | w/o | 577 | 661 | 747 | 589 | 659 | 727 | 621 | 698 | 774 | 541 | 613 | 681 |
| | | w | 210 | 287 | 446 | 170 | 242 | 316 | 181 | 465 | 541 | 193 | 247 | 320 |
| | | Redu.(%) | **63.5** | **56.5** | **40.2** | **71.1** | **63.3** | **56.5** | **70.8** | **33.5** | **30.1** | **64.3** | **59.7** | **53.0** |
| | SPQ | w/o | 0.899 | 0.985 | 1.173 | 1.346 | 1.671 | 1.155 | 1.326 | 1.197 | 2.960 | 1.271 | 1.331 | 1.748 |
| | | w | 0.275 | 0.286 | 0.301 | 0.288 | 0.298 | 0.315 | 0.275 | 0.280 | 0.283 | 0.279 | 0.309 | 0.319 |
| | | Speedup | **3.3×** | **3.4×** | **3.9×** | **4.7×** | **5.6×** | **3.7×** | **4.8×** | **4.3×** | **10.5×** | **4.6×** | **4.3×** | **5.5×** |
| GPT-4.1 | TPQ | w/o | 577 | 661 | 747 | 589 | 659 | 727 | 621 | 698 | 774 | 541 | 613 | 681 |
| | | w | 194 | 280 | 371 | 170 | 253 | 325 | 200 | 465 | 429 | 227 | 252 | 329 |
| | | Redu.(%) | **66.3** | **57.7** | **50.3** | **71.1** | **61.6** | **55.3** | **67.7** | **33.5** | **44.5** | **58.1** | **58.8** | **51.7** |
| | SPQ | w/o | 1.663 | 1.724 | 1.738 | 1.979 | 1.686 | 1.669 | 1.674 | 1.752 | 1.774 | 1.590 | 1.617 | 1.713 |
| | | w | 0.174 | 0.170 | 0.188 | 0.141 | 0.170 | 0.213 | 0.143 | 0.144 | 0.161 | 0.158 | 0.173 | 0.184 |
| | | Speedup | **9.6×** | **10.1×** | **9.3×** | **14.0×** | **9.9×** | **7.9×** | **11.7×** | **12.1×** | **11.0×** | **10.1×** | **9.4×** | **9.3×** |

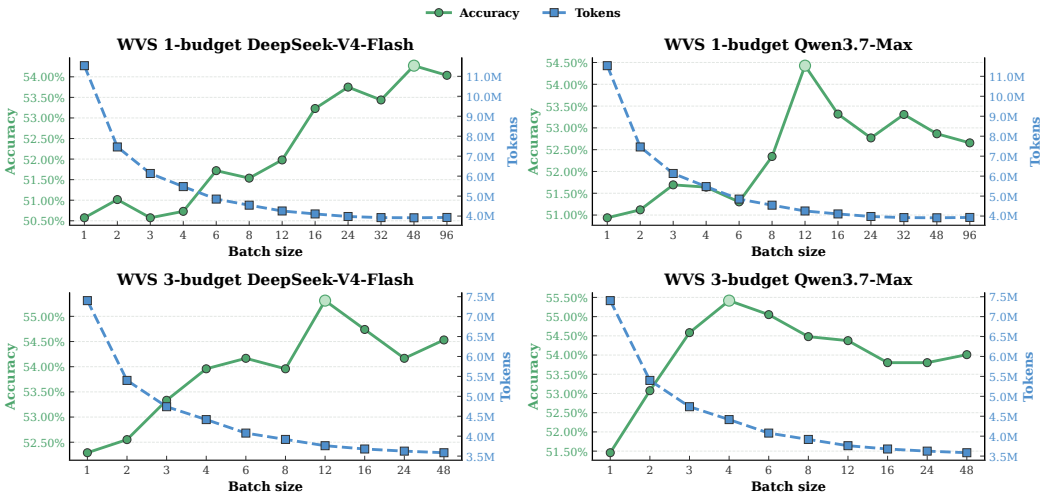



Figure 2: Accuracy and token cost across batch sizes on WVS. The figure compares 1-budget and 3-budget settings using DeepSeek-V4-Flash and Qwen3.7-Max.

### 4.5 Impact of Batch Size (RQ3)

Figure 2 illustrates how batch size affects ACC and TPQ on WVS under the 1-budget and 3-budget settings with DeepSeek-V4-Flash and Qwen3.7-Max. Our framework demonstrates strong robustness to batch size selection: across all tested batch sizes, the batched method consistently outperforms the non-batched baseline in both accuracy and token cost. As batch size increases, TPQ decreases monotonically due to greater context amortization, achieving substantial token reductions even at non-optimal batch sizes. ACC exhibits a non-monotonic pattern, initially improving with moderate batch sizes (around 8–16 for DeepSeek-V4-Flash), then plateauing or slightly declining at very large batch sizes, suggesting that excessively large batches may introduce attentional disruption despite

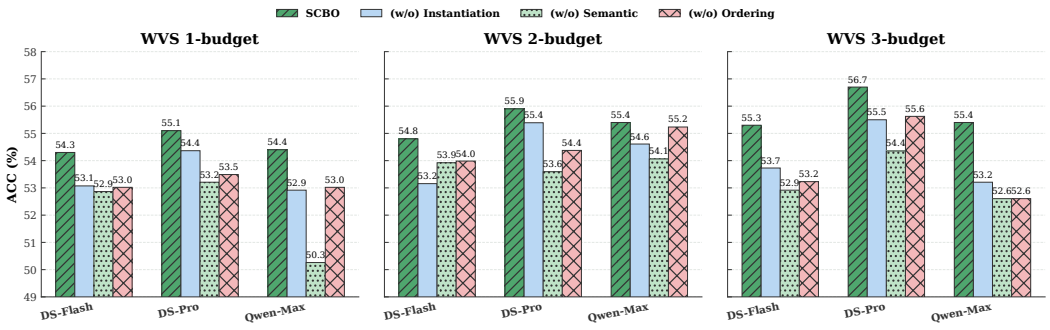



Figure 3: Ablation study on WVS across different component variants and reference budgets.

semantic clustering. These findings confirm that our framework reliably improves over single-query prompting regardless of batch size choice, while also highlighting that moderate batch sizes offer the best balance between token efficiency and prediction accuracy.

### 4.6 Ablation Study (RQ4)

In this section, we perform ablation studies to examine the contribution of each component in SCBO. All experiments are conducted on WVS across three LLMs (DeepSeek-V4-Flash, DeepSeek-V4-Pro, and Qwen3.7-Max) under 1-budget, 2-budget, and 3-budget. The following variants are evaluated: **(w/o) Instantiation**: This variant excludes question instantiation. Raw survey questions are directly embedded without extracting core semantics, introducing template noise into clustering and retrieval. **(w/o) Semantic**: This variant removes semantic batching. Questions are randomly grouped into batches instead of being clustered by semantic similarity, which increases attentional disruption and reduces reference utility across batch questions. **(w/o) Ordering**: This variant omits curriculum-based ordering. Both target questions and reference examples are arranged randomly instead of following the easy-to-hard curriculum and alignment.

Figure 3 reports accuracy for these variants across models and reference budgets. SCBO outperforms all ablation variants. Removing semantic batching (**(w/o) Semantic**) results in the largest performance drop, confirming that semantic clustering is the most critical component: it makes the shared bank relevant to each question and groups mutually informative questions. Removing ordering (**(w/o) Ordering**) degrades performance, highlighting the importance of curriculum-based ordering for exploiting autoregressive dependencies. Removing question instantiation (**(w/o) Instantiation**) shows a smaller decline, indicating that filtering template noise improves embedding quality. These results validate that both modules contribute synergistically, with semantic batching playing the dominant role, followed by ordering, while instantiation provides a useful basis for both.

## 5 Conclusion

In this paper, we addressed three limitations of the one-question-per-prompt paradigm in LLM-based social survey prediction: it re-encodes the same context for every question, restricts each target question to a narrow subset of the respondent's reference responses, and predicts every answer in isolation. We presented SCBO, which batches multiple questions into a single prompt, thereby amortizing the shared context, allowing target questions to share a broader pool of reference responses, and enabling correlated answers to inform one another. Operating on compact question representations produced by a question instantiation step that filters out template noise, SCBO integrates two modules: semantic batching, which groups similar questions and selects a shared reference bank balancing target-specific relevance with batch-level sharing; and curriculum-based ordering, which arranges questions and references so that earlier answers inform later ones. Extensive experiments across diverse datasets and LLMs demonstrate that SCBO generally outperforms the non-batched baseline, achieving superior accuracy while reducing token costs by over 50% and accelerating inference by 3.3–14.0×. Future work will explore model-adaptive batching, which constructs batches based on feedback from the LLM itself rather than relying on heuristic clustering strategies.

## AI USE STATEMENT

AI models serve as experimental components in this work. GPT-5.5 is used for Question Instantiation, extracting compact topic, intent, and entity representations from raw survey items. The `text-embedding-3-small` model encodes these representations into semantic vectors used for clustering, reference retrieval, and ordering. The evaluated LLMs generate predicted survey responses under both the non-batched baseline and SCBO settings. We also used generative AI tools to assist with English translation and language editing. The research idea, theoretical formulation, methodology, experimental design, implementation decisions, and interpretation of results are the authors' own. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final text, claims, code, figures, and reported results.

## ETHICS STATEMENT

This work conducts secondary analysis of publicly released survey datasets, including WVS, GSS, ANES, and BSA. We use de-identified respondent records in accordance with the data-use conditions and do not attempt to identify individual respondents. No new human participants were recruited, and results are reported only in aggregate. Because model-based survey prediction may inherit biases from survey data and LLMs, the predictions should not be treated as substitutes for real respondents or used for high-stakes decisions about individuals or demographic groups. The study is intended to evaluate the methodological feasibility and efficiency of LLM-based social survey prediction.

## REPRODUCIBILITY STATEMENT

We have made an effort to make SCBO reproducible. The complete framework is specified in Section 3, and the experimental setup and implementation details are provided in Section 4. Appendix B describes the capacity-constrained batching procedure; Appendix C documents dataset filtering, respondent-profile construction, reference–target pool construction, and evaluation statistics; and Appendix D provides the prompt templates used for question instantiation and batch prediction. Source code and supporting materials are available at `https://anonymous.4open.science/r/SCBO-41D8`.

## REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Gati V Aher, Rosa I Arriaga, and Adam Tauman Kalai. Using large language models to simulate multiple humans and replicate human subject studies. In *International conference on machine learning*, pp. 337–371. PMLR, 2023.

Lisa P Argyle, Ethan C Busby, Nancy Fulda, Joshua R Gubler, Christopher Rytting, and David Wingate. Out of one, many: Using language models to simulate human samples. *Political Analysis*, 31(3):337–351, 2023.

David Arthur, Sergei Vassilvitskii, et al. k-means++: The advantages of careful seeding. In *Soda*, volume 7, pp. 1027–1035, 2007.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286 (5439):509–512, 1999. doi: 10.1126/science.286.5439.509. URL `https://www.science.org/doi/abs/10.1126/science.286.5439.509`.

Garrett Bernstein and Kyle O'Brien. Stochastic agent-based simulations of social networks. In *Proceedings of the 46th Annual Simulation Symposium*, ANSS 13, San Diego, CA, USA, 2013. Society for Computer Simulation International. ISBN 9781627480307.

James Bisbee, Joshua D Clinton, Cassy Dorff, Brenton Kenkel, and Jennifer M Larson. Synthetic replacements for human survey data? the perils of large language models. *Political Analysis*, 32 (4):401–416, 2024.

Yong Cao, Li Zhou, Seolhwa Lee, Laura Cabello Piqueras, Min Chen, and Daniel Hershcovich. Assessing cross-cultural alignment between chatgpt and human societies: An empirical study. In *Proceedings of the first workshop on cross-cultural considerations in NLP (C3NLP)*, pp. 53–67, 2023.

Xu Chen, Yuanzi Li, Lei Wang, Nan Lu, Yang Wang, Anding Wang, Lei Shi, Xiaoxing Fu, and Ji-Rong Wen. Benchmarking llms for community governance simulation with life-history narratives. *arXiv preprint arXiv:2605.23783*, 2026.

Zhoujun Cheng, Jungo Kasai, and Tao Yu. Batch prompting: Efficient inference with large language model apis. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 792–810, 2023.

Yun-Shiuan Chuang, Agam Goyal, Nikunj Harlalka, Siddharth Suresh, Robert Hawkins, Sijia Yang, Dhavan Shah, Junjie Hu, and Timothy Rogers. Simulating opinion dynamics with networks of llm-based agents. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3326–3346, 2024.

Guillaume Deffuant, David Neau, Frédéric Amblard, and Gérard Weisbuch. Mixing beliefs among interacting agents. *Adv. Complex Syst.*, 3:87–98, 2000. URL https://api.semanticscholar.org/CorpusID:15604530.

Esin Durmus, Karina Nguyen, Thomas I Liao, Nicholas Schiefer, Amanda Askell, Anton Bakhtin, Carol Chen, Zac Hatfield-Dodds, Danny Hernandez, Nicholas Joseph, et al. Towards measuring the representation of subjective global opinions in language models. *arXiv preprint arXiv:2306.16388*, 2023.

Joshua M Epstein and Robert Axtell. *Growing artificial societies: social science from the bottom up*. Brookings Institution Press, 1996.

Antonino Ferraro, Antonio Galli, Valerio La Gatta, Marco Postiglione, Gian Marco Orlando, Diego Russo, Giuseppe Riccio, Antonio Romano, and Vincenzo Moscato. Agent-based modelling meets generative ai in social network simulations. In Luca Maria Aiello, Tanmoy Chakraborty, and Sabrina Gaito (eds.), *Social Networks Analysis and Mining*, pp. 155–170, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-78541-2.

Noah E Friedkin and Eugene C Johnsen. Social influence and opinions. *Journal of mathematical sociology*, 15(3-4):193–206, 1990.

Chen Gao, Xiaochong Lan, Zhihong Lu, Jinzhu Mao, Jinghua Piao, Huandong Wang, Depeng Jin, and Yong Li. $S^3$: Social-network simulation system with large language model-empowered agents, 2025. URL https://arxiv.org/abs/2307.14984.

Robert M Groves, Floyd J Fowler Jr, Mick P Couper, James M Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey methodology*. John Wiley & Sons, 2011.

Lynne Hamill and Geoffrey Gilbert. Social circles: A simple structure for agent-based social network models. *Journal of Artificial Societies and Social Simulation*, 12(2), 2009.

Ori Heffetz and Daniel B Reeves. Difficulty of reaching respondents and nonresponse bias: Evidence from large government surveys. *Review of Economics and Statistics*, 101(1):176–191, 2019.

Sho Hoshino and Peinan Zhang. Cascaded batch prompting. *arXiv preprint arXiv:2608.27038*, 2026.

Graham Kalton. Methods for oversampling rare subpopulations in social surveys. *Survey methodology*, 35(2):125–141, 2009.

Unchitta Kan, Michelle Feng, and Mason A Porter. An adaptive bounded-confidence model of opinion dynamics on networks. *Journal of Complex Networks*, 11(1):415–444, 2023. doi: 10.1093/comnet/cnac055.

Michal Kosinski. Theory of mind may have spontaneously emerged in large language models. *arXiv preprint arXiv:2302.02083*, 4(169):2, 2023.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Yuanzi Li, Quanyu Dai, Xueyang Feng, Zihang Tian, Junhao Wang, Xu Chen, Zhenhua Dong, and Huifeng Guo. Towards fast domain adaptation and fine-grained user simulation for evaluating conversational recommender systems. *arXiv preprint arXiv:2606.22803*, 2026.

Jianzhe Lin, Maurice Diesendruck, Liang Du, and Robin Abraham. Batchprompt: Accomplish more with less. In *International Conference on Learning Representations*, volume 2024, pp. 21590–21612, 2024.

Yuhan Liu, Zirui Song, Juntian Zhang, Xiaoqing Zhang, Xiuying Chen, and Rui Yan. The stepwise deception: Simulating the evolution from true news to fake news with llm agents, 2025. URL https://arxiv.org/abs/2410.19064.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th annual acm symposium on user interface software and technology*, pp. 1–22, 2023.

Siddegowda Roopa and Menta Satya Rani. Questionnaire designing for a survey. *Journal of Indian Orthodontic Society*, 46(4_suppl1):273–277, 2012.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. Whose opinions do language models reflect? In *International conference on machine learning*, pp. 29971–30004. PMLR, 2023.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.

Gabriel Simmons. Moral mimicry: Large language models produce moral rationalizations tailored to political identity. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 282–297, 2023.

Flaminio Squazzoni, J Gareth Polhill, Bruce Edmonds, Petra Ahrweiler, Patrycja Antosz, Geeske Scholz, Émile Chappin, Melania Borit, Harko Verhagen, Francesca Giardini, et al. Computational models that matter during a global pandemic outbreak: A call to action. *Jasss*, 23(2), 2020.

R Tourangeau, LJ Rips, and K Rasinski. The psychology of survey response. cambridge university press; cambridge core, 2000.

Lei Wang, Jingsen Zhang, Hao Yang, Zhi-Yuan Chen, Jiakai Tang, Zeyu Zhang, Xu Chen, Yankai Lin, Hao Sun, Ruihua Song, et al. User behavior simulation with large language model-based agents. *ACM Transactions on Information Systems*, 43(2):1–37, 2025.

Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393: 440–442, 1998. URL https://api.semanticscholar.org/CorpusID:3034643.

James D Wright, Peter V Marsden, et al. Survey research and social science: History, current practice, and future prospects. *Handbook of survey research*, pp. 3–26, 2010.

Haofei Yu, Zhaochen Hong, Zirui Cheng, Kunlun Zhu, Keyang Xuan, Jinwei Yao, Tao Feng, and Jiaxuan You. Researchtown: Simulator of human research community, 2025. URL https://arxiv.org/abs/2412.17767.

Ke Zhou, Marios Constantinides, and Daniele Quercia. Should llms be weird? exploring weirdness and human rights in large language models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 8, pp. 2808–2820, 2025.

Zhibin Zhou, Yaoqi Li, and Junnan Yu. Exploring the application of llm-based ai in ux design: an empirical case study of chatgpt. *Human–Computer Interaction*, pp. 1–33, 2024.

# A Related Work

This section reviews three related areas: LLM-based social simulation, LLM-based survey respondent simulation, and batch prompting.

**Large Language Models for Social Simulation.** Social simulation seeks to explain how macro-level collective phenomena arise from micro-level individual interactions. Early formal treatments model this process through mathematical frameworks in which localized update rules—including Friedkin–Johnsen opinion dynamics and bounded-confidence protocols—drive the evolution of agent states across network topologies (Friedkin & Johnsen, 1990; Deffuant et al., 2000; Kan et al., 2023). Agent-based modeling builds on this bottom-up perspective by populating structured environments with autonomous agents whose interactions give rise to emergent societal patterns (Epstein & Axtell, 1996; Watts & Strogatz, 1998; Barabási & Albert, 1999; Bernstein & O'Brien, 2013). LLM-powered simulation has recently advanced this paradigm by replacing hand-coded behavioral rules with language-grounded agents capable of reasoning over rich demographic profiles and contextual information, substantially expanding the expressiveness of synthetic societies (Park et al., 2023; Yu et al., 2025). Within networked multi-agent frameworks, these LLM agents have been applied to investigate a range of collective phenomena, encompassing opinion shift dynamics, emotional contagion, and the spread of information or misinformation (Gao et al., 2025; Ferraro et al., 2025; Chuang et al., 2024; Liu et al., 2025). Beyond social networks, LLMs have been employed to simulate individual user behaviors in interactive systems and social governance (Wang et al., 2025; Li et al., 2026). In this work, we focus on the individual level of this simulation stack: accurately predicting each respondent's answers to survey questions as a foundation for downstream collective modeling.

**Simulating Survey Respondents with LLMs.** Traditional survey methods face challenges including high costs, logistical constraints, and limited scalability (Wright et al., 2010; Heffetz & Reeves, 2019; Kalton, 2009). Recent advances in Large Language Models have opened new possibilities for social simulation and survey research (Argyle et al., 2023; Park et al., 2023; Aher et al., 2023; Cao et al., 2023; Zhou et al., 2025; Durmus et al., 2023; Wang et al., 2025; Chen et al., 2026). Early works demonstrated that LLMs can exhibit human-like behaviors and attitudes when properly conditioned (Argyle et al., 2023; Park et al., 2023; Zhou et al., 2024). Building on these findings, researchers have explored using LLMs to simulate survey respondents at scale (Santurkar et al., 2023; Simmons, 2023; Bisbee et al., 2024; Cao et al., 2023). These approaches condition LLMs on demographic profiles and observed reference responses to predict answers to survey questions. However, most existing work adopts a one-question-per-prompt paradigm, which incurs prohibitive token costs for large-scale simulations. Our work addresses this limitation by proposing an efficient batching and ordering framework that maintains prediction quality while reducing computational costs.

**Batch Prompting.** Batch prompting improves LLM inference efficiency by processing multiple instances in a single prompt and amortizing shared instructions and demonstrations (Cheng et al., 2023). Cheng et al. also examine semantic and diversity-based grouping, but find no consistent improvement over random batching. Lin et al. (Lin et al., 2024) show that batched predictions are sensitive to instance position and order, and propose Batch Permutation and Ensembling together with early stopping to improve stability while controlling additional cost. Cascaded batch prompting further separates reasoning from output-symbol grounding to improve batched classification (Hoshino & Zhang, 2026). Although closely related, existing batch-prompting methods differ from SCBO in both objective and problem structure. They primarily batch independent instances to reduce inference cost, whereas survey questions for the same respondent are often semantically related and grounded in shared respondent evidence. SCBO exploits this structure by constructing a broader respondent-specific reference bank shared across questions and by ordering questions from easy to hard with semantically aligned references. These designs allow SCBO to improve prediction accuracy in addition to inference efficiency.

# B Implementation Details

For notational simplicity and to match the experimental settings, the main text assumes that the number of target questions is divisible by the batch capacity, so that every batch contains exactly $C$ questions; here, we consider the general non-divisible case, in which the final batch may contain fewer than $C$ questions (Appendix B.1). We then describe how to obtain the resulting capacity-constrained

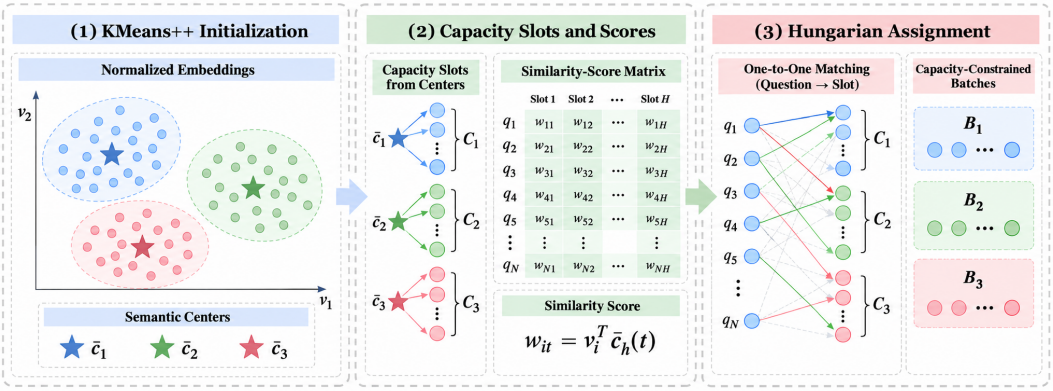



Figure 4: Capacity-constrained semantic batching. KMeans++ identifies semantic centers from question embeddings. Each center is expanded into slots according to its capacity, and the Hungarian algorithm computes a one-to-one assignment that produces capacity-constrained batches.

partition in practice (Appendix B.2), as illustrated in Figure 4. For each respondent, we first use KMeans++ to obtain initial semantic centers and then apply the Hungarian algorithm to assign target questions to slots with prescribed capacities. The resulting assignment satisfies the prescribed capacities and provides a deterministic approximation to the fixed-size clustering objective, and all operations are performed independently for each respondent.

### B.1 Notation and Boundary Cases

For a respondent $u$, let $\mathcal{Q}^u_{target}$ contain $N_u$ target questions and let $\mathcal{Q}^u_{pool}$ contain $H_u$ observed question–answer pairs. We use $C$ as the maximum number of target questions in a batch, $n$ as the per-question reference budget, and $r$ as the retrieval depth used to expose relevant reference IDs, where $r \geq n$. The number of batches is

$$K_u = \left\lceil \frac{N_u}{C} \right\rceil. \tag{9}$$

The first $K_u - 1$ batches have capacity $C$, and the final batch has capacity $N_u - C(K_u - 1)$. Thus, for a batch $B_k$ of size $C_k = |B_k|$, its shared reference-bank budget is respondent- and batch-specific:

$$m_k = nC_k. \tag{10}$$

If $C \geq N_u$, then $K_u = 1$ and all target questions form a single batch. If $N_u$ is not divisible by $C$, only the final batch has capacity smaller than $C$. When $H_u < m_k$, the reference bank cannot reach its nominal budget and contains at most $H_u$ reference pairs. These cases do not alter the batching assignment, but they affect the realized batch or reference-bank size.

### B.2 Capacity-Constrained Semantic Batching

We approximate the fixed-size clustering objective in Section 3.2 in two stages. First, we run KMeans with KMeans++ initialization, $K_u$ clusters, ten initializations, and respondent-specific random seed $s_u = s + u$. The resulting cluster centers are then L2-normalized. Second, we enforce the prescribed batch capacities through a linear-assignment problem.

Specifically, let $\bar{\mathbf{c}}_1, \ldots, \bar{\mathbf{c}}_{K_u}$ be the normalized KMeans centers, and create $C_k$ identical assignment slots for cluster $k$. If slot $t$ belongs to cluster $h(t)$, its assignment score for target question $q_i$ is

$$w_{it} = \mathbf{v}_i^\top \bar{\mathbf{c}}_{h(t)}. \tag{11}$$

We solve

$$\max_x \sum_{i=1}^{N_u} \sum_{t=1}^{N_u} x_{it} w_{it} \quad \text{s.t.} \quad \sum_t x_{it} = 1, \quad \sum_i x_{it} = 1, \quad x_{it} \in \{0, 1\}, \tag{12}$$

using the Hungarian algorithm. Mapping each assigned slot to its associated cluster produces batches with the prescribed capacities. Given the respondent-specific seed, this provides a deterministic capacity-constrained approximation to Equation 5.

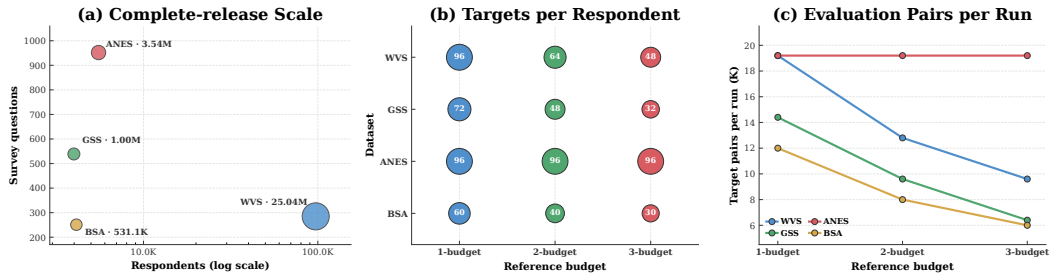



Figure 5: Dataset scale and evaluation allocation. (a) Complete-release scale, where the horizontal axis denotes the number of respondents, the vertical axis denotes the number of survey questions, and marker area represents the number of observed respondent–question pairs. (b) Number of target questions assigned to each respondent under reference budgets. (c) Total target pairs evaluated in each method–model run. WVS uses Wave 7, while GSS, ANES, and BSA use their 2024 releases.

### B.2.1 Inference-Time Measurement

All models are accessed through OpenAI-compatible APIs: the official DeepSeek endpoint (`https://api.deepseek.com`) for DeepSeek-V4-Flash and DeepSeek-V4-Pro, Alibaba Cloud DashScope (`https://dashscope.aliyuncs.com/compatible-mode/v1`) for Qwen3.7-Max, and an OpenAI-compatible third-party gateway (`https://aiapi.world/v1`) for GPT-4.1. We use greedy decoding with temperature 0, disable streaming and model thinking, and set the maximum output length to 4,096 tokens. Both the non-batched baseline and SCBO use 30 concurrent workers. Failed or empty requests are retried up to five times with exponential backoff starting at two seconds. For each request, elapsed time is measured with a monotonic wall-clock timer from prompt construction through API completion and output parsing. SPQ is computed by summing the de-duplicated request times and dividing by the number of target questions. It therefore measures aggregate request latency amortized per question rather than end-to-end wall-clock time under parallel execution. The client configuration is held fixed across methods, but we cannot control provider-side region, routing, network conditions, or server load; the reported speedups should be interpreted as measurements under our API environment rather than provider-independent latency guarantees.

## C Dataset Processing

We standardize the four survey releases into respondent–question prediction records and construct disjoint reference and target pools. All compared methods use the same processed records and respondent-specific pool assignments.

### C.1 Survey Sources and Response Processing

We use World Values Survey Wave 7 (WVS), General Social Survey 2024 (GSS), American National Election Studies 2024 (ANES), and British Social Attitudes 2024 (BSA). Figure 5(a) summarizes the complete-release scale underlying the exhaustive cost estimate in Table 1. An observed pair comprises a respondent, a survey question, and the respondent's recorded answer.

Question text and response options are extracted from each survey's official codebook and matched to the corresponding respondent-level variables. We retain closed-ended, single-choice questions with complete option sets. Open-ended questions, multiple-selection questions, multi-field questions, and items containing unresolved template placeholders are excluded.

We remove administrative and non-substantive response categories, including missing, skipped, refused, invalid, and interviewer-only codes. Substantive choices, such as neutral positions and respondent-selectable "other" categories, are retained. Questions without a valid answer set after this filtering are discarded.

The remaining options are ordered according to their source codes and mapped to consecutive indices $\{0, \ldots, L_q - 1\}$ for a question with $L_q$ valid choices. Each respondent answer is transformed using

the corresponding question-specific mapping, and records that cannot be mapped to a retained option are removed. Consequently, every retained answer $a_{ui}$ satisfies

$$0 \leq a_{ui} < L_{q_i}. \tag{13}$$

### C.2 Respondent Profiles

Background and demographic items are separated from prediction questions and converted into a third-person natural-language profile for each respondent. Profile items do not enter either the reference pool or the target pool, and respondents without an available profile are excluded.

### C.3 Reference–Target Pool Construction

For each dataset, we collect all valid non-profile responses and construct respondent-specific reference and target pools separately for each reference budget. Under a given budget, all methods receive identical respondent profiles, references, and target questions.

Let $k \in \{1, 2, 3\}$ denote the per-target reference budget and $T_{d,k}$ the number of target questions assigned to each retained respondent in dataset $d$. Valid non-profile responses are randomly ordered using seed 42. The first $T_{d,k}$ records define the target pool, and the subsequent $kT_{d,k}$ records define the reference pool. The two pools therefore satisfy

$$\mathcal{Q}^u_{target} \cap \mathcal{Q}^u_{pool} = \emptyset, \qquad \mathrm{card}(\mathcal{Q}^u_{target}) = T_{d,k}, \qquad \mathrm{card}(\mathcal{Q}^u_{pool}) = kT_{d,k}. \tag{14}$$

A respondent is eligible only if at least $(k+1)T_{d,k}$ valid non-profile responses are available. Respondents with insufficient coverage are excluded, and target responses are never reused as references. Pool assignments remain unchanged across models, batch sizes, and compared methods.

The per-respondent target allocation is shown in Figure 5(b). Because the coverage requirement depends on $k$ and $T_{d,k}$, the eligible respondent set may differ across reference budgets.

SCBO retrieves references exclusively from the corresponding respondent's reference pool. Embeddings are computed from question text and answer options; respondent answers serve only as labels for retrieved references and are excluded from the embedding input.

### C.4 Evaluation Data Statistics

For each dataset–budget setting, respondents are randomly split into 15% validation and 85% test sets. Batch size is selected by validation ACC for each dataset–model–budget configuration, and all reported metrics are computed on the shared test set. Figure 5(c) reports the number of test target pairs per method–model run.

## D Prompt Templates

We report the exact templates used for question instantiation (Section 3.1) and batch prediction (Section 3.3).

### D.1 Question Instantiation Prompt

The following prompt implements $\mathcal{P}_{extract}$ in Section 3.1. Given a raw question $q_i$, the instantiation model $\mathcal{M}$ produces $\tilde{q}_i = \{T_i, I_i, E_i\}$, representing its topic, core intent, and key entities. The returned `instance` field is embedded by $f_{emb}$. The placeholder `{question_payload}` contains the question identifier, text, and answer options.

**Prompt A: Question Instantiation**

```
You extract compact structured instances from survey questions for
semantic embedding.
Remove generic survey wording, respondent-addressing phrases, and
repeated answer-scale phrasing.
```

```
Keep only the core topic, entities, and intent.
Return valid JSON only with these keys:
{ "topic": "...", "intent": "...", "entities": ["..."], "instance":
 "Topic: ...; Intent: ...; Entities: ..." }

Survey question:
{question_payload}

[The {question_payload} placeholder is replaced with:]
Question ID: {qid}
Question: {question_text}
Options:
{options_text}
```

### D.2 Batch Prediction Prompt

For each batch $B_k$, the target LLM receives the respondent profile, shared reference bank $S_k$, and ordered target questions through the following prompt.

**Prompt B: Batch Prediction**

```
You are predicting how a specific person would answer survey
questions.

## Person Profile
The following are known background and profile answers from this
person:
{persona_text}

## Shared Reference Bank
The following are this person's actual answers to related questions.
 Use the whole bank to calibrate the person's values and response
style. Each target question below includes Relevant reference IDs.
These IDs point to entries in this Shared Reference Bank and mark
the references most semantically related to that target question.
For each target question, first use its listed Relevant reference
IDs as primary evidence, then use the rest of the bank and the
person profile as secondary context:

{fs_block}

## Task
Based on the person profile and shared Reference Bank above,
predict this person's answer to each question below.

## Questions
{question_blocks}

# Output Format
Output valid JSON only. Do not wrap in markdown code blocks.
Return exactly one answer object per sample_id. Each answer is a
single integer option index, starting from 0.
{"answers": [{"sample_id": "0", "answer": 0}]}

[The {fs_block} placeholder is replaced with:]
### Reference Entries
### Reference {local_idx} (reference_id: "{local_idx}")
Question: {question_text}

Options:
{options_text}
```

```
Person's actual answer: {answer}

[The {question_blocks} placeholder contains easy-to-hard target
questions in this format:]
### Question {local_idx} (sample_id: "{local_idx}")
Relevant reference IDs: [{ref_ids}]
Question: {question_text}

Options:
{options_text}
```

For target question $q_i$, `Relevant reference IDs` lists the entries in $S_k \cap R_i$ by decreasing target–reference similarity (Section 3.2). The complete shared bank remains available as context.