TRACE: BENCHMARKING AGENTIC CAUSAL ESTIMATION FROM TEXT UNDER COVARIATE AND POSTERIOR SHIFT

Anonymous authors
Paper under double-blind review

ABSTRACT

Average treatment effect (ATE) estimation is central to causal inference, but extending it to settings with open-ended text covariates is not straightforward. An estimator must connect information in individual narratives to a target-population effect, even when source and target populations differ in their covariates and causal mechanisms. Large language models (LLMs) can interpret text and perform mathematical reasoning, but those abilities alone do not ensure reliable effect estimates. To study this gap, we introduce TRACE-Bench, an end-to-end benchmark built from social narratives for evaluating ATE estimation across shifted source and target domains. Generic LLM agents, including direct estimation and planning-based approaches, are unreliable on this benchmark, while structured statistical estimators remain competitive. We therefore propose TRACE, a shift-adaptive framework that retains transferable source information and adapts the mechanisms that change. TRACE-AGENT learns and adapts natural-language contexts for causal estimation before doubly robust aggregation. TRACE-NN is an LLM-free, factor-augmented neural realization of the same framework. Across foundation models, TRACE-AGENT consistently improves ATE estimation over generic agent baselines, while TRACE-NN remains competitive with substantially larger LLMs. We finally prove that the proposed TARCE is doubly robust, which will give consistent estimation of ATE as long as either the propensity or outcome model is consistently estimated.

1 INTRODUCTION

Many questions in social science are recorded as narratives rather than rows in a table. Interviews, case reports, policy documents, and online discussions can describe who received an intervention, what happened afterward, and which treatment circumstances may have affected both. The scientific goal is often a population-level quantity such as the average treatment effect (ATE), instead of a prediction or judgment about one narrative. Estimating this quantity from text requires a model to recover confounding information across many accounts and use it in a valid causal estimator.

This problem becomes even more complicated when the population of interest has little labeled data. A related source population may be much larger, but it can be hard to make the assumption that the source narrative distribution is the same as the target population since they may cover different topics and their treatment and outcome mechanisms can differ. These changes create covariate shift and posterior shift. Pooling the two populations can then introduce bias, while estimating every target mechanism from the small target sample can have high variance. A useful transfer method must decide which source information still applies and which parts need to be relearned for the target population.

While the large language models (LLMs) offer a natural way to work with raw narratives (Dhawan et al., 2024; Guo et al., 2025; Ma et al., 2026), reading a narrative to estimate an ATE is a different task. Existing causal benchmarks mainly test verbal reasoning or tabular analysis (Jin et al., 2024; Liu et al., 2024; Hu et al., 2024); they do not measure whether a system can complete this statistical task from raw text under population shift. Nor do they reveal whether a general-purpose agent can execute the analysis reliably without being given a specialized causal pipeline.

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.

We introduce **TRACE-Bench** to evaluate this capability. TRACE-Bench contains 21 source–target ATE estimation tasks built from 60,738 posts on Reddit's r/AmITheAsshole (Guo et al., 2025), where users describe social dilemmas and discuss if there is a fault. Each unit is one post: the covariate X is the post text before comment exposure; the treatment is exposure to the top-upvoted comment ($T = 1$) or a randomly selected comment from the same thread ($T = 0$); and the outcome is an LLM-generated moral verdict, coded as $Y = 1$ for fault detected and $Y = 0$ for no fault detected. For each post, an LLM judge provides the paired potential outcomes $Y(1)$ and $Y(0)$. The estimator observes only the outcome under one simulated treatment assignment, while the mean paired difference in the target population serves as the known semi-synthetic ground-truth ATE. The benchmark does not prescribe an agent architecture, a text representation, or a causal estimator. A system is evaluated only by the accuracy of its target ATE estimates. On this benchmark, directly asking six LLMs to read the observed records and estimate the ATE gives RMSEs from 0.0387 to 0.3653. The compact neural estimator TRACE-NN obtains an RMSE of 0.0327, making even the best LLM under direct estimation 18% worse. This result shows that general language ability alone does not solve the statistical problem. It also motivates a dedicated pipeline that lets an LLM read the text while controlling how its predictions enter the ATE estimator.

We propose **TRACE** (**T**ext **R**easoning and **A**daptive **C**ausal **E**stimation) to address the limitations of existing methods. TRACE has two implementations that follow the same four stages: learn source nuisance models, diagnose shift, adapt to the target population, and estimate the target ATE by augmented inverse propensity weighting (Bang & Robins, 2005; Farrell, 2015). The source-to-target interface consists of three predictions: the propensity score and the two outcome regressions. **TRACE-AGENT** implements the four stages with a frozen LLM: it learns propensity and outcome contexts from source labels, tests their transferability on held-out target data, and reuses or adapts them before ATE estimation. This procedure works with different LLM backbones and, across all six tested models, outperforms generic agents and target-only context learning. **TRACE-NN** is a simplified, LLM-free realization of TRACE-AGENT. It replaces the LLM components with factor-augmented sparse neural networks (Fan & Gu, 2024; Fan et al., 2025), while retaining the same transfer interface and doubly robust estimator. TRACE-NN outperforms every generic LLM-agent configuration with approximately 100.46 million parameters, about $40\times$ fewer than the smallest LLM tested. During target adaptation, both realizations consume target data and source predictions rather than source records or source-model weights, and neither repeats source training. This interface keeps the source and target records separate and limits data movement. Under the conditions in Section 5, the shared estimator is consistent when either both outcome models or the propensity model are consistently estimated, allowing the other component to be misspecified.

Our contributions can be summarized as follows:

- **Benchmark.** We introduce TRACE-Bench, 21 tasks for evaluating target ATE estimation from raw narratives under covariate and posterior shift. Its paired outcomes provide numerical ground truth without imposing an agent architecture or estimation procedure. We also find the limitations of existing LLMs on this task as demonstrated by our experiments.
- **Method and package.** We develop TRACE, a four-stage pipeline that learns source nuisance models, diagnoses shift, adapts the mechanisms that change, and applies doubly robust aggregation. The package includes TRACE-AGENT for different LLM backbones and TRACE-NN as its lighter neural case. Both transfer source predictions without requiring source records, weights, or source retraining at the target site. Our methods significantly improved the capacity of LLMs to accomplish the task as verified by experiments on multiple different models.
- **Statistical grounding.** We establish double robustness of the shared AIPW estimator: under the stated identification, moment, clipping, and nuisance-convergence conditions, consistent estimation of either both outcome models or the propensity model suffices for consistent target ATE estimation.

Related work. Our work is closely related to three branches of literature. First, prior work on machine learning in social applications identifies limitations in LLM causal reasoning (Jin et al., 2024; Du et al., 2026) and privacy risks in processing personal narratives and adapting models (Staab et al., 2024; Marek et al., 2026). Combining model predictions with gold-standard data through bias-corrected estimators enables valid statistical inference (Egami et al., 2023; Angelopoulos et al.,

2023), though label savings can be limited in LLM-based evaluation (Dorner et al., 2025). Motivated by this literature, TRACE separates prediction from causal adjustment and uses predictions rather than individual records for cross-domain adaptation. Second, the treatment effect literature develops flexible nuisance models primarily for structured covariates (Robins et al., 1994; Farrell et al., 2021; Fan & Gu, 2024; Fan et al., 2025). In terms of text covariates, existing methods mostly rely on intermediate inputs for causal estimation (Dhawan et al., 2024; Guo et al., 2025; Ma et al., 2026). A separate line of work addresses data scarcity through transfer across related populations (Bica & van der Schaar, 2022; Wei et al., 2025; Chai et al., 2026). TRACE extends these approaches by predicting nuisance functions directly from raw text and transferring source predictions under population shift. Third, recent advances in agent frameworks and reinforcement learning have enabled complex reasoning and long-horizon decision-making for LLM systems (Yao et al., 2022; Chung et al., 2026; Shao et al., 2024; Shi et al., 2026), and test-time learning improves these systems at deployment through feedback and accumulated experience without weight updates (Madaan et al., 2023; Shinn et al., 2023; Agrawal et al., 2026; Zhang et al., 2026; Suzgun et al., 2026; Ouyang et al., 2026; Karten et al., 2026). TRACE further shows how test-time context learning can optimize LLMs as nuisance function estimators for population-level causal inference.

2 Causal Estimation from Text under Covariate and Posterior Shift

In this section, we first formulate the problem of target ATE estimation from text when the source and target populations may differ in both covariates and causal mechanisms in section 2.1. We then introduce TRACE-Bench, which serves as a benchmark for evaluating LLMs' ability to estimate the target ATE in section 2.2.

2.1 Problem Setup and Challenges

Here we provide a rigorous but stylized mathematical formulation of the problem of interest. The problem that we studied is ATE estimation from text data under distribution shift. In this problem, the source distribution and the target distribution are different. Let P and Q denote the source and target distributions, respectively. We assume that the observations are independent samples

$$\mathcal{D}^P = \{(X_i^P, T_i^P, Y_i^P)\}_{i=1}^{n^P} \overset{\text{i.i.d.}}{\sim} P, \qquad \mathcal{D}^Q = \{(X_j^Q, T_j^Q, Y_j^Q)\}_{j=1}^{n^Q} \overset{\text{i.i.d.}}{\sim} Q,$$

where $n^P \gg n^Q$. For either domain $* \in \{P, Q\}$, $X^* \in \mathcal{X}$ is a pretreatment covariate, $T^* \in \{0, 1\}$ is a binary treatment, and $Y^*(0), Y^*(1)$ are the potential outcomes. We observe only $Y^* = T^*Y^*(1) + (1-T^*)Y^*(0)$. Here, $\mathcal{X}$ is the space of open-ended pretreatment narratives. TRACE-AGENT reads a narrative $x \in \mathcal{X}$ directly. TRACE-NN uses a numerical representation map $\varphi : \mathcal{X} \to \mathbb{R}^p$, with $\boldsymbol{X}^* = \varphi(X^*)$ and realized feature vector $\boldsymbol{x} = \varphi(x)$ (e.g., bag-of-words counts or text embeddings). We reserve unbolded X, x for narratives and bold $\boldsymbol{X}, \boldsymbol{x}$ for numerical covariates.

For $* \in \{P, Q\}$, define the propensity score $\pi^*(x) = \mathbb{P}_*(T^* = 1 \mid X^* = x)$ and outcome regressions $\mu_t^*(x) = \mathbb{E}_*\{Y^*(t) \mid X^* = x\}$ for $t \in \{0, 1\}$. Under consistency, conditional unconfoundedness, and overlap, the ATE in the target population is identified as

$$\tau^Q = \mathbb{E}_Q\{Y^Q(1) - Y^Q(0)\} = \mathbb{E}_Q\{\mu_1^Q(X^Q) - \mu_0^Q(X^Q)\}. \tag{1}$$

Section 5 states these conditions formally. We aim to estimate τ^Q when the source sample is large, but the target sample is small. Because τ^Q is a population quantity, estimating it requires the target nuisance functions π^Q, μ_0^Q, and μ_1^Q, rather than a collection of unrelated judgments about individual narratives.

The large source sample is useful only if its information transfers to the target domain. Yet the two domains need not agree: *covariate shift*, $P_X \neq Q_X$, changes the distribution of the narratives, while *posterior shift*, $(\pi^P, \mu_0^P, \mu_1^P) \neq (\pi^Q, \mu_0^Q, \mu_1^Q)$, changes the treatment or outcome mechanisms. Pooling the samples or applying a source-only model can then introduce bias. At the other extreme, fitting all three nuisance functions on the small target sample can be unstable.

The textual covariates make this transfer problem harder. Information about confounding may be spread across the people involved, their relationships, the order of events, and the surrounding con-

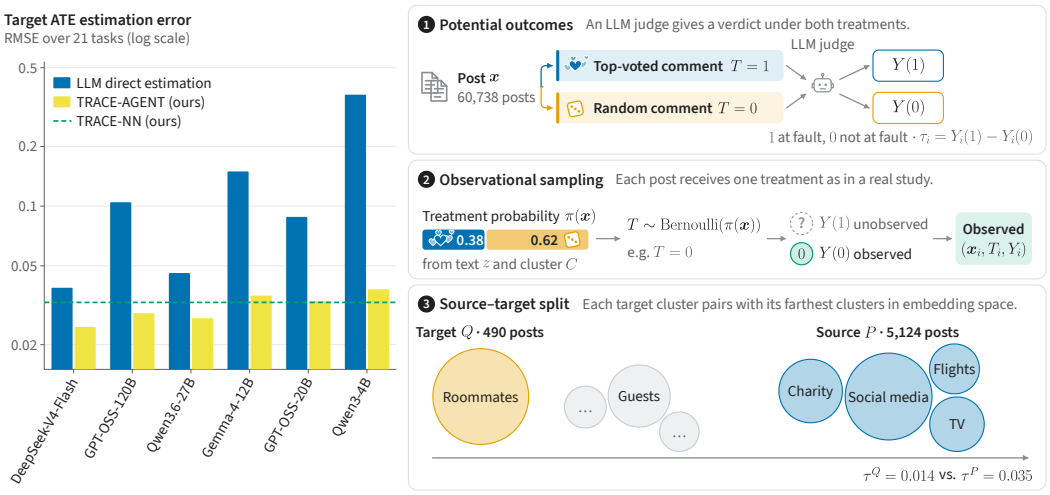


Figure 1: TRACE-Bench overview. **Left:** ATE estimation error (RMSE over 21 tasks, log scale; lower is better). LLM direct estimation is less accurate than both TRACE methods with every foundation model. **Right:** Benchmark construction. An LLM judge gives both potential outcomes of each post. A text-dependent propensity score assigns one treatment per post, which yields observational data with a ground truth ATE. Source and target domains are distant topic clusters.

text. An estimator must recover this information consistently across many narratives before estimating a population effect. To formalize what is transferred, let $\boldsymbol{s}^*(x) = (\pi^*(x), \mu_0^*(x), \mu_1^*(x))$ denote the nuisance signal in domain $*$. We use the source estimates $\widehat{\boldsymbol{s}}^P(x)$ together with the limited target data to estimate $\boldsymbol{s}^Q(x)$ and τ^Q. Since the three nuisance functions may shift differently, the estimator must decide which source predictions to retain and which to correct or discard.

2.2 TRACE-Bench: Text-Based Causal Estimation under Population Shift

The challenges above raise an empirical question: can an LLM estimate a target ATE from raw-text observations across shifted source and target populations? This question matters because understanding individual narratives does not guarantee an accurate population-level causal effect.

Existing benchmarks usually separate verbal causal reasoning from tabular analysis (Jin et al., 2024; Liu et al., 2024; Hu et al., 2024), so they do not test this capability. **TRACE-Bench** (**T**ext **R**easoning and **A**daptive **C**ausal **E**stimation Benchmark) fills the gap by retaining narratives as covariates, constructing shifted source and target populations, and scoring the target ATE against ground truth.

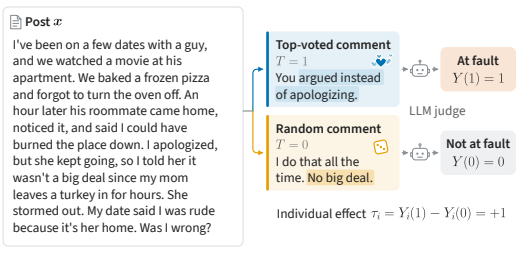


Figure 2: Example post and paired potential outcomes used as semi-synthetic ground truth.

Benchmark construction. TRACE-Bench builds on 60,738 social narratives from the semi-synthetic moral-judgment dataset of Guo et al. (2025). Each unit is one post, as in Figure 2: X is the post written before comment exposure, T indicates exposure to the top-upvoted or a random comment, and Y is the resulting LLM verdict. Following Marmarelis et al. (2024), paired verdicts define $Y(1)$ and $Y(0)$, whose average target-population difference is τ^Q, but each estimator observes only the outcome under one simulated treatment. We form 21 tasks by pairing each target topic cluster with semantically distant source clusters; Appendix D.1 gives the full construction.

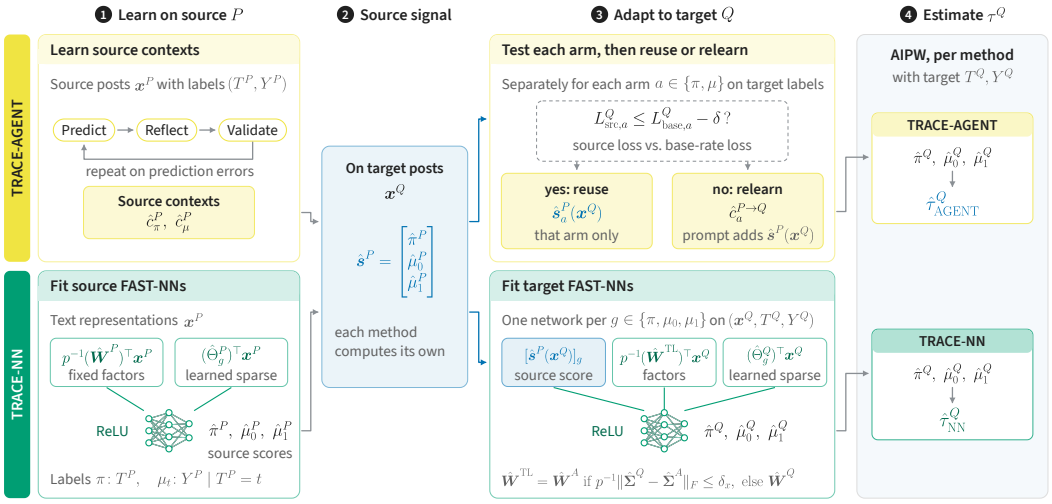


Figure 3: Overview of TRACE-AGENT and TRACE-NN. The two methods share a four-stage pipeline consisting of source learning, shift diagnosis, target adaptation, and AIPW estimation of the target ATE. TRACE-AGENT learns LLM contexts, while TRACE-NN fits factor-augmented neural networks.

What TRACE-Bench reveals. We ask six LLMs to estimate the target ATE directly from raw-text source and target records, without an explicit causal design. As Figure 1 shows, their RMSEs range from 0.0387 to 0.3653, all above the 0.0327 achieved by TRACE-NN; even the best model under direct estimation has about 18% higher error. TRACE-NN is the neural branch of our framework, so this gap suggests that reading narratives alone is not enough to estimate population effects under domain shift; the information must also be organized through a causal estimation procedure. This gap motivates the shared TRACE pipeline developed next.

3 TRACE: Shift-Adaptive Causal Estimation

To address the limitations of existing LLM-based approaches, we introduce *TRACE* (**T**ext **R**easoning and **A**daptive **C**ausal **E**stimation). TRACE follows four stages: *source learning*, *shift diagnosis*, *target adaptation*, and *AIPW*. This framework is adapted to both agentic and neural settings for both text and numerical data: TRACE-AGENT implements the four stages by computations of an LLM from the given context and the source nuisance signal; TRACE-NN replaces the agentic components with factor-augmented neural networks taking numerical text representations and the source nuisance signal as inputs, which reduces the computational cost compared to an LLM (Figure 3).

3.1 TRACE-AGENT

TRACE-AGENT learns source contexts for propensity and outcome prediction, screens their usefulness on target data, and adapts them when needed. It then combines the target predictions with observed target treatments and outcomes through AIPW to estimate the target ATE.

Step 1: Source learning. Source learning estimates treatment and outcome probabilities from each narrative. Here c is a natural-language instruction context supplied with narrative x to the frozen LLM predictor $F_a(x; c)$. Let $\mathcal{C}^P_a$ be the candidate contexts and $D^P_{V,a}$ the source validation set for arm $a \in \{\pi, \mu\}$, where μ jointly predicts both outcomes. We choose a context c as the natural-language instruction context by factual binary log loss $\mathcal{L}_a$ and use it to predict from x:

$$\widehat{s}^P_a(x) = F_a(x; \widehat{c}^P_a), \qquad \widehat{c}^P_a \in \arg\min_{c \in \mathcal{C}^P_a} \mathcal{L}_a\big(F_a(\cdot; c); D^P_{V,a}\big). \tag{2}$$

The propensity call returns $\widehat{s}_\pi^P(x) = \widehat{\pi}^P(x)$; the joint outcomes call returns $\widehat{s}_\mu^P(x) = (\widehat{\mu}_0^P(x), \widehat{\mu}_1^P(x))$, the probabilities under $T = 0$ and $T = 1$ respectively, which together form the signal from source learning $\widehat{\boldsymbol{s}}^P(x) = (\widehat{\pi}^P(x), \widehat{\mu}_0^P(x), \widehat{\mu}_1^P(x))$. The propensity loss compares $\widehat{\pi}^P$ with observed T; the outcome loss compares only $\widehat{\mu}_T^P$ with observed Y. The reflective context search follows Agrawal et al. (2026).

Step 2: Shift diagnosis. Shift diagnosis checks whether source predictions improve on a simple target baseline. For target fold I_k, use only labeled data D_{-k}^Q outside the fold to compare source predictions with a constant predictor $\overline{q}_{a,-k}$. The baseline uses the Laplace-smoothed average $(1 + \sum_{i=1}^n z_i)/(n+2)$ of n observed binary labels: T for propensity, and Y separately among observations with $T = 0$ and $T = 1$ for outcomes. For outcomes, the loss scores only the prediction for each unit's observed treatment. Reuse arm $a \in \{\pi, \mu\}$ only if its mean factual binary log loss is lower by at least $\delta = 0.01$:

$$G_{a,k} = \mathbf{1}\Big\{\mathcal{L}_a(\widehat{s}_a^P; D_{-k}^Q) \le \mathcal{L}_a(\overline{q}_{a,-k}; D_{-k}^Q) - \delta\Big\}, \qquad a \in \{\pi, \mu\}. \tag{3}$$

Step 3: Target adaptation. Target adaptation produces propensity and outcome predictions for each held-out target narrative. For an arm with $G_{a,k} = 1$, we reuse the source score as the target prediction. For an arm with $G_{a,k} = 0$, the reflection agent takes the current target context, the gate diagnostic, target prediction errors, and the source predictions on D_{-k}^Q, and returns the transferred context $\widehat{c}_{a,k}^{P\to Q}$. The frozen predictor then takes the held-out narrative x, $\widehat{\boldsymbol{s}}^P(x)$, and this context as input. With $\widehat{s}_\pi^P = \widehat{\pi}^P$ and $\widehat{s}_\mu^P = (\widehat{\mu}_0^P, \widehat{\mu}_1^P)$, the output is

$$\widehat{s}_{a,k}^Q(x) = \begin{cases} \widehat{s}_a^P(x), & G_{a,k} = 1, \\ F_a\Big(x, \widehat{\boldsymbol{s}}^P(x); \widehat{c}_{a,k}^{P\to Q}\Big), & G_{a,k} = 0. \end{cases} \tag{4}$$

The propensity arm returns one target probability, and the outcome arm returns two. Repeating the procedure across folds outputs one held-out $\widehat{\pi}_i$, $\widehat{\mu}_{0,i}$, and $\widehat{\mu}_{1,i}$ for each target observation.

Step 4: Augmented Inverse Propensity Weighting (AIPW) Estimator. In the end, we use the doubly robust Augmented Inverse Propensity Weighting (AIPW) estimator to combine the held-out probabilities and observed target (T_i^Q, Y_i^Q) into one estimate of the target ATE. With propensities clipped away from zero and one, we compute the ATE estimate $\widehat{\tau}^Q$ on the target population by the AIPW estimator (Bang & Robins, 2005; Farrell, 2015) without calling any LLM:

$$\widehat{\tau}^Q = \frac{1}{n^Q}\sum_{i=1}^{n^Q}\left[\widehat{\mu}_{1,i} - \widehat{\mu}_{0,i} + \frac{T_i^Q(Y_i^Q - \widehat{\mu}_{1,i})}{\widehat{\pi}_i} - \frac{(1-T_i^Q)(Y_i^Q - \widehat{\mu}_{0,i})}{1-\widehat{\pi}_i}\right]. \tag{5}$$

Algorithm 1 and Appendices D.2 and D.5 give the context search, data splits, prompts, parsing rules, and computational settings for TRACE-AGENT.

3.2 TRACE-NN: A Factor-Augmented Neural Special Case

TRACE-NN implements TRACE with factor-augmented sparse neural networks (FAST-NNs) (Fan et al., 2025; Fan & Gu, 2024) on high-dimensional numerical covariates, including text-derived features $\boldsymbol{x} = \varphi(x)$. Under the factor model $\boldsymbol{x} = \boldsymbol{B}\boldsymbol{f} + \boldsymbol{u}$, a FAST-NN predictor has the form $g(\boldsymbol{x}) = h([p^{-1}W^\top \boldsymbol{x}, \overline{T}_M(\Theta_g^\top \boldsymbol{x})])$, where the bounded ReLU network $h \in \mathcal{G}$ combines estimated factors with a sparsity-penalized covariate projection. TRACE-NN follows the four TRACE-AGENT stages: *(1) Source learning* fits source FAST-NNs for π, μ_0, μ_1 to produce $\widehat{\boldsymbol{s}}^P$ in place of source-context predictions; *(2) Shift diagnosis* uses a covariance gate (Chai et al., 2026) on unlabeled source and target data to choose pooled or target-only factors; *(3) Target adaptation* fits target FAST-NNs with those factors, sparse target features, and the matching source prediction $\widehat{g}^P(\boldsymbol{x})$ using target labels; and *(4) AIPW* combines the target nuisance estimates through (5). Appendix B gives the full procedure, network architecture, and fitting objectives.

Table 1: Target-domain ATE estimation RMSE on TRACE-Bench across 21 tasks (lower is better). Results use per-task median ATE estimates from three runs, with task-level bootstrap standard errors in parentheses. **Top:** LLM methods evaluated with six models. **Bottom:** Statistical ML methods.

Model	Direct estimation	KIR	Plan-and-execute	**TRACE-AGENT (ours)**
DeepSeek-V4-Flash	0.0387 (0.0051)	0.0378 (0.0041)	0.0399 (0.0047)	**0.0246** (0.0029)
GPT-OSS-120B	0.1046 (0.0187)	0.0357 (0.0038)	0.0460 (0.0059)	**0.0288** (0.0034)
Qwen3.6-27B	0.0460 (0.0042)	0.0418 (0.0063)	0.0525 (0.0091)	**0.0272** (0.0032)
Gemma-4-12B	0.1497 (0.0209)	0.1291† (0.0292)	0.0909 (0.0364)	**0.0354** (0.0047)
GPT-OSS-20B	0.0883 (0.0122)	0.0361 (0.0037)	0.0533 (0.0079)	**0.0332** (0.0048)
Qwen3-4B	0.3653 (0.0871)	0.1079 (0.0490)	0.0720 (0.0164)	**0.0381** (0.0036)

ℓ_1-TCL (DragonNet)	ℓ_1-TCL (TARNet)	Finetune LASSO	Finetune Vanilla-NN	FIDDLE (source-only)	FIDDLE (target-only)	**TRACE-NN (ours)**
0.0625 (0.0089)	0.0342 (0.0041)	0.0440 (0.0048)	0.0863 (0.0152)	0.0457 (0.0056)	0.0368 (0.0042)	**0.0327** (0.0036)

†16/21 tasks; remaining runs returned no parsable estimate.

4 Experiments and Ablation Studies

We first describe TRACE-Bench evaluation and the LLM-agent and statistical baselines in section 4.1. We then compare TRACE-AGENT and TRACE-NN with these baselines in section 4.2 before isolating the effects of prompt optimization and source-to-target transfer in section 4.3.

4.1 Experimental setup

In this section, we evaluate TRACE-NN and TRACE-AGENT on the 21 disjoint target-domain ATE estimation tasks in TRACE-Bench (Sections 2 and 4.2). We compare both methods with LLM agent and statistical ML baselines using root mean squared error (RMSE) across tasks. We also examine reflective prompt optimization and cross-domain transfer in TRACE-AGENT through ablation studies.

LLM agent baselines. We compare TRACE with three baselines adapted from established LLM workflows. **Direct estimation** uses tool-assisted interaction (Yao et al., 2022) to inspect raw-text records through a file-reading tool and estimate the target ATE without a prescribed estimation procedure. **Key information retrieval (KIR)** extracts task-relevant phrases from each narrative and uses the compressed narratives as prompt input for one-shot ATE prediction, following prior work on context compression (Xu et al., 2024). **Plan-and-execute** uses a coding agent that plans before execution (Wang et al., 2023). The same foundation model acts first as planner and then as executor, specifying an estimation procedure and implementing it with sandboxed Python. We evaluate TRACE-AGENT and all three baselines with the six foundation models listed in Table 1.

Statistical ML baselines. **ℓ_1-TCL** (Wei et al., 2025) transfers TARNet (Shalit et al., 2017) or DragonNet (Shi et al., 2019) through ℓ_1-regularized training for doubly robust ATE estimation. **Finetune Vanilla-NN** and **Finetune LASSO** replace FAST-NN in the TRACE-NN pretraining-finetuning pipeline with a fully connected ReLU network or LASSO, respectively. **FIDDLE** (Fan et al., 2025) uses only source-domain or target-domain data without transfer. All statistical ML methods, including TRACE-NN, represent each post by bag-of-words counts over the corpus vocabulary ($p = 67{,}398$).

4.2 Main results

The results of our main experiments are reported in Table 1, from which we have the following observations.

TRACE-AGENT improves ATE estimation accuracy and reduces sensitivity to the foundation model. As shown in Table 1, TRACE-AGENT achieves the lowest reported RMSE among LLM methods, with substantially less variation across models. The baseline results suggest that translating individual records into population-level effect estimates remains challenging for general-

Table 2: TRACE-AGENT transfer versus target-only context learning on TRACE-Bench. "Target only" learns contexts from target data alone with the same context optimization budget. (a) ATE RMSE across 21 tasks, using median estimates from three runs per task. (b) Held-out MSE against oracle probabilities, pooled across six models by TRACE-AGENT's gate decision. "Relearn" updates the target prompt using target prediction errors and source predictions, then uses the updated prompt and source predictions to make target predictions. "Reuse" applies the source-learned context to target narratives without adaptation when the gate selects reuse. Lower is better.

(a) ATE estimation RMSE

Model	TRACE-AGENT	Target only
DeepSeek-V4	**0.0246**	0.0270
Gemma-4-12B	**0.0354**	0.0362
GPT-OSS-120B	**0.0288**	0.0326
GPT-OSS-20B	**0.0332**	0.0363
Qwen3.6-27B	**0.0272**	0.0276
Qwen3-4B	**0.0381**	0.0493

(b) Nuisance prediction MSE

Gate decision	Prediction	TRACE-AGENT	Target only
	Factual	**0.15326**	0.15960
Relearn	Counterfactual	**0.15754**	0.16458
	Propensity	**0.00812**	0.01543
	Factual	**0.13583**	0.13891
Reuse	Counterfactual	**0.13776**	0.14094
	Propensity	**0.00758**	0.01527

purpose agents, as a simple difference-in-means (DiM) estimator ($\bar{Y}_{T=1} - \bar{Y}_{T=0}$) in the target domain achieves a better RMSE (0.0375) than 16 of 18 reported baseline agent configurations. The results of Plan-and-execute further suggest that, even when an LLM selects a standard estimator for the ATE estimation, difficulties remain in downstream model selection and training (see Appendix E.1). TRACE-AGENT addresses these difficulties through a structured procedure for propensity and outcome prediction, target adaptation, and doubly robust aggregation.

TRACE-NN achieves competitive accuracy with compact prediction models. TRACE-NN achieves the lowest RMSE among statistical ML methods and outperforms all general-purpose agent configurations. It also surpasses TRACE-AGENT with Qwen3-4B, Gemma-4-12B, and GPT-OSS-20B. These results show that TRACE can achieve competitive ATE estimation with a compact neural predictor and bag-of-words covariates. Its advantage over the fine-tuning baselines and source-only or target-only FIDDLE is consistent with the benefit of combining factor-based prediction, source learning, and target adaptation.

TRACE is robust across prediction models and data settings. TRACE-NN and TRACE-AGENT use the same estimation framework with neural networks and LLM predictors, respectively. Both implementations use source information to improve target-domain prediction and combine the resulting nuisance estimates through doubly robust aggregation. Appendices E.3 and E.4 provide further validation through the TRACE-NN implementation. Together with the TRACE-Bench results, these experiments support the versatility of TRACE and the robustness of its statistical design across the evaluated prediction models and data settings.

4.3 Ablation studies

Our ablation studies address two questions: (1) How does reflective prompt optimization affect ATE estimation? (2) Does learning from source data improve target-domain estimation over learning from target data alone?

Reflective prompt optimization improves ATE accuracy, with diminishing gains from additional rounds. In a source-only ablation with Qwen3-4B, we independently vary the propensity and outcome optimization budgets from 0 to 8 rounds to isolate these gains from transfer (Figure 4). The initial prompts yield high ATE error, while one round per estimator reduces RMSE by 88.2%. Further optimization of both estimators yields complementary gains, but improvements beyond four rounds per estimator are modest.

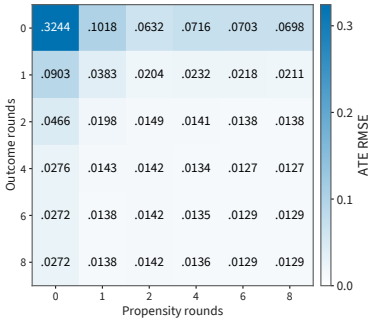


Figure 4: Source-only ATE RMSE vs. prompt-optimization budgets (Qwen3-4B) over 21 tasks, using per-task medians of three runs.

Source learning improves target-domain ATE estimation beyond target-only context learning. Under the same optimization budget, TRACE-AGENT achieves lower ATE RMSE than target-only context learning across all six foundation models (Table 2a). Held-out nuisance predictions pooled across models also have lower MSE for factual outcomes, counterfactual outcomes, and propensity scores, both when the gate reuses source-context predictions and when it invokes target adaptation (Table 2b). The largest relative gains are in propensity estimation, with MSE reduced by roughly half in both groups. These results support the benefit of source information through selective reuse and target-specific adaptation.

5 Statistical Grounding

Under the setup of Section 2, we establish double robustness of the AIPW estimator $\widehat{\tau}^Q$ in (5), which is shared by TRACE-NN and TRACE-AGENT.

Assumption 1 (Causal identification)**.** *For each $* \in \{P, Q\}$: (i) (Unconfoundedness) T^* is independent of $(Y^*(0), Y^*(1))$ given X^*; (ii) (Overlap) there exists a universal constant $\alpha \in (0, 1/2)$ such that $\pi^*(x) \in (\alpha, 1-\alpha)$ for all $x \in \mathcal{X}$.*

Assumption 1 gives the standard conditions for identifying the ATE from observational data (Rosenbaum & Rubin, 1983). For TRACE-NN on text, the fitted nuisance functions in this section are understood as functions of the narrative through composition with φ. The following result assumes that the nuisance estimates converge to fixed limit functions in both population and empirical norms. It allows some of these limits to be misspecified: consistency of the target ATE estimator requires only that either both outcome limits or the propensity limit equal their true counterparts.

Theorem 1 (Double robustness)**.** *Suppose Assumption 1 holds. Fix $0 < \underline{\alpha} \le \alpha$ with $\underline{\alpha} < 1/2$, and clip $\widehat{\pi}^Q$ to $[\underline{\alpha}, 1-\underline{\alpha}]$. Let Q_X denote the target covariate marginal. Assume $\mathbb{E}_Q[Y^2] < \infty$, $\mu_t^{Q,*} \in L_2(Q_X)$ for $t \in \{0,1\}$, and $\pi^{Q,*}(x) \in [\underline{\alpha}, 1-\underline{\alpha}]$ Q_X-a.s. Let Q_{X,n^Q} be its empirical distribution and define $\|g\|^2_{Q_X,2} := \int g(x)^2\, dQ_X(x)$, $\|g\|^2_{Q_{X,n^Q},2} := \frac{1}{n^Q}\sum_{j=1}^{n^Q} g(X_j^Q)^2$. Suppose further that the estimators converge to limit functions $\mu_0^{Q,*}, \mu_1^{Q,*}, \pi^{Q,*}$ in both the population and empirical norms: $\|\widehat{\mu}_t^Q - \mu_t^{Q,*}\|_{Q_X,2} + \|\widehat{\mu}_t^Q - \mu_t^{Q,*}\|_{Q_{X,n^Q},2} = o_p(1)$, $t \in \{0,1\}$, $\|\widehat{\pi}^Q - \pi^{Q,*}\|_{Q_X,2} + \|\widehat{\pi}^Q - \pi^{Q,*}\|_{Q_{X,n^Q},2} = o_p(1)$. Then, if either (i) $\mu_t^{Q,*} = \mu_t^Q$ a.s. for $t \in \{0,1\}$, or (ii) $\pi^{Q,*} = \pi^Q$ a.s., the AIPW estimator* (5) *satisfies $\widehat{\tau}^Q - \tau^Q = o_p(1)$.*

Under its convergence conditions, Theorem 1 ensures consistency when either both outcome models or the propensity model are consistently estimated, allowing the other component to be misspecified. The result applies to either realization of TRACE when these conditions hold. The proof uses the standard AIPW bias decomposition (Robins et al., 1994; Bang & Robins, 2005) and is given in Appendix C.

6 Discussion, Limitations, and Broader Implications

TRACE-Bench and TRACE serve as a starting point for studying causal estimation from unstructured data under population shift by language models. The current evaluation is limited to semi-synthetic tasks from one text corpus and one family of topic shifts, and the guarantees still require causal identification and nuisance convergence. Future work should study more post-training across source–target tasks and extend the framework to multimodal covariates or predictions from multiple specialized models. These directions can make agentic causal estimates more robust, auditable, and broadly applicable.

AI use statement

In this work, we used generative AI tools for generating synthetic datasets, implementing methods, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting

in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, or interpreting results. Translation assistance is not applicable to this work. Additionally, we used generative AI tools for modifying scientific figures or images and editing the manuscript to improve readability. We have reviewed all AI-assisted work. Two authors double-checked and edited the AI-generated results and implemented tests to verify them. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics statement

Our study uses an existing research dataset derived from publicly available Reddit posts and released by prior work. We do not directly collect data from or interact with Reddit users, and we do not attempt to identify or contact individual authors. Usernames are anonymized, and we report only aggregate results without revealing individual-level information. As the narratives may contain sensitive or offensive content, we avoid reproducing identifiable information and use the data solely in accordance with the source dataset's research terms.

Reproducibility statement

We provide our code with detailed reproduction instructions as anonymous supplementary material. In the paper, Appendix D.1 describes each step of the TRACE-Bench construction, including the embeddings, topic clustering, domain splitting, treatment simulation, and random seed. Section 3.1 and Algorithm 1 specify TRACE-AGENT, and Appendix D.2 gives its data splits, budgets, parsing rules, and the foundation model checkpoints and decoding settings. All prompts for TRACE-AGENT and the LLM baselines are listed in Appendix D.5. Section 3.2 and Appendix B describe TRACE-NN, and Appendix D.3 reports the hyperparameters of TRACE-NN and the statistical ML baselines. Appendix D.4 states the evaluation protocol and hardware. Section 5 states the double-robustness theorem and its conditions, and Appendix C provides the proof.

References

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Anastasios N Angelopoulos, Stephen Bates, Clara Fannjiang, Michael I Jordan, and Tijana Zrnic. Prediction-powered inference. *Science*, 382(6671):669–674, 2023.

Heejung Bang and James M Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61(4):962–973, 2005.

Ioana Bica and Mihaela van der Schaar. Transfer learning on heterogeneous feature spaces for treatment effects estimation. *Advances in Neural Information Processing Systems*, 35:37184–37198, 2022.

Jinhang Chai, Jianqing Fan, Cheng Gao, and Qishuo Yin. Smart fine-tuning factor augmented neural lasso. *arXiv preprint arXiv:2604.12288*, 2026.

Jui-Hui Chung, Ziyang Cai, Zihao Li, Qishuo Yin, Rohit Agarwal, Simon Park, Rodrigo Porto, Narutatsu Ri, Ziran Yang, Shange Tang, et al. Goedel-architect: Streamlining formal theorem proving with blueprint generation and refinement. *arXiv preprint arXiv:2606.06468*, 2026.

Nikita Dhawan, Leonardo Cotta, Karen Ullrich, Rahul G Krishnan, and Chris J Maddison. End-to-end causal effect estimation from unstructured natural language data. *Advances in Neural Information Processing Systems*, 37:77165–77199, 2024.

Florian Eddie Dorner, Vivian Nastl, and Moritz Hardt. Limits to scalable evaluation at the frontier: Llm as judge won't beat twice the data. In *International Conference on Learning Representations*, volume 2025, pp. 26467–26491, 2025.

Jin Du, Li Chen, Xun Xian, An Luo, Fangqiao Tian, Ganghua Wang, Charles Doss, Xiaotong Shen, and Jie Ding. Ice cream doesn't cause drowning: Benchmarking llms against statistical pitfalls in causal inference. In *International Conference on Learning Representations*, volume 2026, pp. 90314–90325, 2026.

Naoki Egami, Musashi Hinck, Brandon Stewart, and Hanying Wei. Using imperfect surrogates for downstream inference: Design-based supervised learning for social science applications of large language models. *Advances in Neural Information Processing Systems*, 36:68589–68601, 2023.

Jianqing Fan and Yihong Gu. Factor augmented sparse throughput deep relu neural networks for high dimensional regression. *Journal of the American Statistical Association*, 119(548):2680–2694, 2024.

Jianqing Fan, Soham Jana, Sanjeev Kulkarni, and Qishuo Yin. Factor informed double deep learning for average treatment effect estimation. *arXiv preprint arXiv:2508.17136*, 2025.

Max H Farrell. Robust inference on average treatment effects with possibly more covariates than observations. *Journal of Econometrics*, 189(1):1–23, 2015.

Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference. *Econometrica*, 89(1):181–213, 2021.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. Late chunking: Contextual chunk embeddings using long-context embedding models. *arXiv preprint arXiv:2409.04701*, 2024.

Siyi Guo, Myrl G Marmarelis, Fred Morstatter, and Kristina Lerman. Estimating causal effects of text interventions leveraging llms. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 170–190, 2025.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, et al. Infiagent-dabench: Evaluating agents on data analysis tasks. *arXiv preprint arXiv:2401.05507*, 2024.

Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, David Ha, et al. Can large language models infer causation from correlation? In *International Conference on Learning Representations*, volume 2024, pp. 28663–28679, 2024.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pp. 3020–3029. PMLR, 2016.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026.

Xiao Liu, Zirui Wu, Xueqing Wu, Pan Lu, Kai-Wei Chang, and Yansong Feng. Are llms capable of data-based statistical and causal reasoning? benchmarking advanced quantitative reasoning with data. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9215–9235, 2024.

Yuchen Ma, Dennis Frauen, Jonas Schweisthal, and Stefan Feuerriegel. Llm-driven treatment effect estimation under inference time text confounding. *Advances in Neural Information Processing Systems*, 38:10457–10490, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Bartłomiej Marek, Lorenzo Rossi, Vincent Hanke, Xun Wang, Michael Backes, Franziska Boenisch, and Adam Dziedzic. Benchmarking empirical privacy protection for adaptations of large language models. *arXiv preprint arXiv:2606.09401*, 2026.

Myrl G Marmarelis, Greg Ver Steeg, Aram Galstyan, and Fred Morstatter. Ensembled prediction intervals for causal outcomes under hidden confounding. In *Causal Learning and Reasoning*, pp. 18–40. PMLR, 2024.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long Le, Samira Daruki, Xiangru Tang, et al. Reasoningbank: Scaling agent self-evolving with reasoning memory. In *International Conference on Learning Representations*, volume 2026, pp. 94327–94354, 2026.

Qwen Team. Qwen3.6-27B: Flagship-level coding in a 27B dense model, April 2026. URL `https://qwen.ai/blog?id=qwen3.6-27b`.

James M Robins, Andrea Rotnitzky, and Lue Ping Zhao. Estimation of regression coefficients when some regressors are not always observed. *Journal of the American statistical Association*, 89 (427):846–866, 1994.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *International conference on machine learning*, pp. 3076–3085. PMLR, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Chengshuai Shi, Wenzhe Li, Xinran Liang, Yizhou Lu, Wenjia Yang, Ruirong Feng, Seth Karten, Ziran Yang, Zihan Ding, Gabriel Sarch, et al. Odysseus: Scaling vlms to 100+ turn decision-making in games via reinforcement learning. *arXiv preprint arXiv:2605.00347*, 2026.

Claudia Shi, David Blei, and Victor Veitch. Adapting neural networks for the estimation of treatment effects. *Advances in neural information processing systems*, 32, 2019.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652, 2023.

Robin Staab, Mark Vero, Mislav Balunovic, and Martin Vechev. Beyond memorization: Violating privacy via inference with large language models. In *International Conference on Learning Representations*, volume 2024, pp. 33832–33878, 2024.

Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7080–7106, 2026.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2609–2634, 2023.

Song Wei, Hanyu Zhang, Ronald Moore, Rishikesan Kamaleswaran, and Yao Xie. Transfer learning for causal effect estimation. In *Big Data Analytics in Biostatistics and Bioinformatics*, pp. 269–324. Springer, 2025.

Anyi Xu, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chenchen Ling, et al. Deepseek-v4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *International Conference on Learning Representations*, volume 2024, pp. 43478–43502, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026.

A Overview of Appendix

Appendix B gives the detailed TRACE-NN architecture, factor projections, and source and target fitting objectives. Appendix C proves the double-robustness result in Theorem 1. Appendix D gives implementation and benchmark details, and Appendix E analyzes the baseline LLM agents and reports the controlled simulation and semi-synthetic News results.

B TRACE-NN: Detailed Procedure

This appendix specifies the TRACE-NN architecture and the four-stage procedure summarized in Section 3.2. Here $\boldsymbol{x}$ denotes numerical covariates; for text observations, $\boldsymbol{x}_i^* = \varphi(X_i^*)$ for $* \in \{P, Q\}$.

B.1 TRACE-NN Method Overview

TRACE-NN follows the four stages in Section 3.2. It uses labeled source data to learn nuisance functions, separate unlabeled source and target covariates to choose a factor representation, and labeled target data to adapt the source predictions. The steps are:

1. **Source learning.** Estimate a source factor projection from the unlabeled source covariates. Fit separate source FAST-NNs for propensity π^P and outcomes μ_0^P, μ_1^P using the labeled source data. Applied to a target covariate vector $\boldsymbol{x}$, they produce the source signal $\widehat{\boldsymbol{s}}^P(\boldsymbol{x}) = (\widehat{\pi}^P(\boldsymbol{x}), \widehat{\mu}_0^P(\boldsymbol{x}), \widehat{\mu}_1^P(\boldsymbol{x}))$.
2. **Shift diagnosis.** Compare the target covariance with the pooled source–target covariance, both estimated from reserved unlabeled samples. When they are close, use the pooled covariance to estimate target factors; otherwise, use the target covariance. The leading eigenvectors of the chosen matrix yield $\widehat{\boldsymbol{W}}^{TL}$ and the target factor surrogate $\widehat{\boldsymbol{f}}^Q(\boldsymbol{x}) = p^{-1}(\widehat{\boldsymbol{W}}^{TL})^\top \boldsymbol{x}$. This gate selects the factor representation; source nuisance predictions remain available in either case.
3. **Target adaptation.** For each $g \in \{\pi, \mu_0, \mu_1\}$, fit a target FAST-NN using the target factor surrogate, a sparse target covariate projection, and the matching source prediction $\widehat{g}^P(\boldsymbol{x})$:

$$\widehat{g}^Q(\boldsymbol{x}) = \widehat{h}_g\Big(\Big[\widehat{\boldsymbol{f}}^Q(\boldsymbol{x}), \overline{T}_M((\widehat{\boldsymbol{\Theta}}_g^Q)^\top \boldsymbol{x}), \widehat{g}^P(\boldsymbol{x})\Big]\Big).$$

   Target treatment labels train propensity, and observed outcomes within each treatment group train the outcome models. Each network can retain, correct, or ignore its source prediction.

4. **AIPW.** Combine the three fitted target nuisance functions with observed target treatments and outcomes using (5) to estimate τ^Q. The reported experiments fit target networks on the full target sample.

The following subsections specify the network architecture, factor projections, and fitting objectives. Appendix D.3 reports the experimental hyperparameters and optimization settings.

B.2 Factor Representation and Network Architecture

TRACE-NN uses a factor representation of the source and target covariates,

$$\boldsymbol{x}^P = \boldsymbol{B}^P \boldsymbol{f}^P + \boldsymbol{u}^P, \qquad \boldsymbol{x}^Q = \boldsymbol{B}^Q \boldsymbol{f}^Q + \boldsymbol{u}^Q, \tag{6}$$

where $\boldsymbol{B}^* \in \mathbb{R}^{p\times r}$ contains factor loadings, $\boldsymbol{f}^* \in \mathbb{R}^r$ contains latent factors, and $\boldsymbol{u}^* \in \mathbb{R}^p$ contains residual covariate variation, for $* \in \{P, Q\}$. A FAST-NN combines estimated factors with a learned sparse projection of the raw covariates. At the target stage, the matching source nuisance prediction is an additional input.

Deep ReLU networks. The estimators are built from fully connected deep neural networks with ReLU activation $\sigma(\cdot) = \max\{\cdot, 0\}$. For input dimension d, depth L, width N, output truncation level M, and weight bound B, the deep ReLU network class is

$$\mathcal{G}(d, L, N, M, B) = \big\{\tilde{g}(\boldsymbol{x}) = \mathrm{trun}_M(g(\boldsymbol{x})) : g = \mathcal{L}_{L+1} \circ \bar{\sigma}_L \circ \mathcal{L}_L \circ \cdots \circ \bar{\sigma}_1 \circ \mathcal{L}_1, \\ \|\boldsymbol{W}_\ell\|_{\max} \le B,\ \|\boldsymbol{b}_\ell\|_{\max} \le B\big\},$$

where $\mathcal{L}_\ell(\boldsymbol{z}) = \boldsymbol{W}_\ell \boldsymbol{z} + \boldsymbol{b}_\ell$ are affine maps with $\boldsymbol{W}_\ell \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$, hidden dimensions $(d_0, d_1, \ldots, d_{L+1}) = (d, N, \ldots, N, 1)$, $\bar{\sigma}_\ell$ applies ReLU element-wise, and $\mathrm{trun}_M(z) = \mathrm{sign}(z) \cdot \min\{|z|, M\}$.

For source fitting, the network class is $\mathcal{G}(\bar{r} + N^P, L^P, N^P, M, B^P)$; for target fitting, it is $\mathcal{G}(\bar{r} + N + 1, L, N, M, B)$. Here $\bar{r}$ is the number of estimated factors, and the superscript P distinguishes source network settings from target settings. The source and target sparsity weights are λ^P and λ, respectively.

B.3 The TRACE-NN Procedure

In addition to the labeled analysis samples $\mathcal{D}^P$ and $\mathcal{D}^Q$, the procedure uses independent covariate-only samples

$$\mathcal{U}^P = \{\boldsymbol{x}_i^{P,\mathrm{DP}}\}_{i\in[m^P]}, \qquad \mathcal{U}^Q = \{\boldsymbol{x}_j^{Q,\mathrm{DP}}\}_{j\in[m^Q]}.$$

Here $\mathcal{U}^P$ and $\mathcal{U}^Q$ are i.i.d. from the source and target covariate marginals, respectively, and are mutually independent and independent of $\mathcal{D}^P \cup \mathcal{D}^Q$. In the implementation they are reserved unlabeled splits. Here DP denotes diversified projection. The sizes m^P, m^Q refer to the projection samples, whereas n^P, n^Q denote the labeled sample sizes used for nuisance fitting and AIPW estimation.

Diversified projection matrices. Following Chai et al. (2026, Section 3.2), we compute the covariance matrices from the independent DP samples as follows:

$$\widehat{\boldsymbol{\Sigma}}^P_{\mathrm{DP}} = \frac{1}{m^P}\sum_{i=1}^{m^P} \boldsymbol{x}_i^{P,\mathrm{DP}} (\boldsymbol{x}_i^{P,\mathrm{DP}})^\top, \qquad \widehat{\boldsymbol{\Sigma}}^Q_{\mathrm{DP}} = \frac{1}{m^Q}\sum_{j=1}^{m^Q} \boldsymbol{x}_j^{Q,\mathrm{DP}} (\boldsymbol{x}_j^{Q,\mathrm{DP}})^\top,$$

$$\widehat{\boldsymbol{\Sigma}}^A_{\mathrm{DP}} = \frac{m^P}{m^P + m^Q}\widehat{\boldsymbol{\Sigma}}^P_{\mathrm{DP}} + \frac{m^Q}{m^P + m^Q}\widehat{\boldsymbol{\Sigma}}^Q_{\mathrm{DP}},$$

and select

$$\widehat{\boldsymbol{\Sigma}}^{TL}_{\mathrm{DP}} = \begin{cases} \widehat{\boldsymbol{\Sigma}}^A_{\mathrm{DP}}, & \text{if } p^{-1}\|\widehat{\boldsymbol{\Sigma}}^Q_{\mathrm{DP}} - \widehat{\boldsymbol{\Sigma}}^A_{\mathrm{DP}}\|_F \le \delta_x, \\ \widehat{\boldsymbol{\Sigma}}^Q_{\mathrm{DP}}, & \text{otherwise,} \end{cases}$$

where $\delta_x > 0$ controls source borrowing in the covariance gate, separately from the TRACE-AGENT loss margin δ. Let $\widehat{\boldsymbol{v}}_1^{TL}, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}}^{TL}$ be the top-$\bar{r}$ eigenvectors of $\widehat{\boldsymbol{\Sigma}}^{TL}_{\mathrm{DP}}$ and set

$$\widehat{\boldsymbol{W}}^{TL} = p^{1/2}[\widehat{\boldsymbol{v}}_1^{TL}, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}}^{TL}]. \tag{7}$$

Similarly, define $\widehat{\boldsymbol{W}}^P = p^{1/2}[\widehat{\boldsymbol{v}}_1^P, \ldots, \widehat{\boldsymbol{v}}_{\bar{r}}^P]$ from the top-$\bar{r}$ eigenvectors of $\widehat{\boldsymbol{\Sigma}}_{\mathrm{DP}}^P$. The factor surrogates are

$$\widehat{\boldsymbol{f}}_i^P = p^{-1}(\widehat{\boldsymbol{W}}^P)^\top \boldsymbol{x}_i^P, \qquad \widehat{\boldsymbol{f}}_j^Q = p^{-1}(\widehat{\boldsymbol{W}}^{TL})^\top \boldsymbol{x}_j^Q.$$

Source model estimation. Following the selector classes in Fan & Gu (2024, Appendix G) and Chai et al. (2026, Appendix B.1), all source and target candidate selectors satisfy

$$\|\boldsymbol{\Theta}_g^P\|_{\max} \le B^P, \qquad \|\boldsymbol{\Theta}_g^Q\|_{\max} \le B, \qquad g \in \{\pi, \mu_0, \mu_1\}, \tag{8}$$

where $\|\boldsymbol{\Theta}\|_{\max} := \max_{j,k} |\Theta_{jk}|$ and B^P, B are the respective network weight bounds. These bounds are part of the optimization domains below. The source propensity and outcome models are estimated with FAST-NN:

$$\begin{aligned}
\widehat{\pi}^P(\cdot), \widehat{\boldsymbol{\Theta}}_\pi^P = & \underset{\substack{\pi \in \mathcal{G}(\cdot) \\ \boldsymbol{\Theta}_\pi^P \in [-B^P, B^P]^{p \times N^P}}}{\arg\min} \frac{1}{n^P} \sum_{i=1}^{n^P} \left(T_i^P - \pi \left(\left[\widehat{\boldsymbol{f}}_i^P, \overline{T}_M((\boldsymbol{\Theta}_\pi^P)^\top \boldsymbol{x}_i^P) \right] \right) \right)^2 \\
& \qquad + \lambda^P \sum_{i,j} \psi_\eta(\boldsymbol{\Theta}_{\pi,i,j}^P), \\
\widehat{\mu}_t^P(\cdot), \widehat{\boldsymbol{\Theta}}_{\mu_t}^P = & \underset{\substack{\mu_t \in \mathcal{G}(\cdot) \\ \boldsymbol{\Theta}_{\mu_t}^P \in [-B^P, B^P]^{p \times N^P}}}{\arg\min} \frac{1}{n_t^P} \sum_{i: T_i^P = t} \left(y_i^P - \mu_t \left(\left[\widehat{\boldsymbol{f}}_i^P, \overline{T}_M((\boldsymbol{\Theta}_{\mu_t}^P)^\top \boldsymbol{x}_i^P) \right] \right) \right)^2 \\
& \qquad + \lambda^P \sum_{i,j} \psi_\eta(\boldsymbol{\Theta}_{\mu_t,i,j}^P),
\end{aligned} \tag{9}$$

where $t \in \{0, 1\}$, $n_t^P = \sum_{i=1}^{n^P} \mathbf{1}(T_i^P = t)$, and $\overline{T}_M(\cdot)$ applies trun_M entrywise. The clipped-ℓ_1 penalty of FAST-NN (Fan & Gu, 2024) is

$$\psi_\eta(z) := \min\{|z|/\eta, 1\}, \qquad J_\eta(\boldsymbol{\Theta}) := \sum_{j,k} \psi_\eta(\Theta_{jk}). \tag{10}$$

The variable selection matrices $\boldsymbol{\Theta}$ pass learned combinations of raw covariates to the network alongside the factor surrogates, with sparsity encouraged by the penalty.

Target model finetuning. Applying the source models to the target covariates yields pretrained signals

$$\begin{aligned}
\widehat{s}_\pi(\boldsymbol{x}_j^Q) &= \widehat{\pi}^P \left(\left[p^{-1}(\widehat{\boldsymbol{W}}^P)^\top \boldsymbol{x}_j^Q, \overline{T}_M((\widehat{\boldsymbol{\Theta}}_\pi^P)^\top \boldsymbol{x}_j^Q) \right] \right), \\
\widehat{s}_{\mu_t}(\boldsymbol{x}_j^Q) &= \widehat{\mu}_t^P \left(\left[p^{-1}(\widehat{\boldsymbol{W}}^P)^\top \boldsymbol{x}_j^Q, \overline{T}_M((\widehat{\boldsymbol{\Theta}}_{\mu_t}^P)^\top \boldsymbol{x}_j^Q) \right] \right), \quad j \in [n^Q],\ t \in \{0, 1\}.
\end{aligned}$$

The source models remain fixed. The target networks $h_\pi, h_{\mu_t} \in \mathcal{G}(\bar{r} + N + 1, L, N, M, B)$ are fitted with the pretrained signals as additional inputs:

$$\begin{aligned}
\widehat{h}_\pi(\cdot), \widehat{\boldsymbol{\Theta}}_\pi^Q = & \underset{\substack{h_\pi(\cdot) \\ \boldsymbol{\Theta}_\pi^Q \in [-B, B]^{p \times N}}}{\arg\min} \frac{1}{n^Q} \sum_{j=1}^{n^Q} \left(T_j^Q - h_\pi \left(\left[\widehat{\boldsymbol{f}}_j^Q, \overline{T}_M((\boldsymbol{\Theta}_\pi^Q)^\top \boldsymbol{x}_j^Q), \widehat{s}_\pi(\boldsymbol{x}_j^Q) \right] \right) \right)^2 \\
& \qquad + \lambda \sum_{i,j} \psi_\eta(\boldsymbol{\Theta}_{\pi,i,j}^Q), \\
\widehat{h}_{\mu_t}(\cdot), \widehat{\boldsymbol{\Theta}}_{\mu_t}^Q = & \underset{\substack{h_{\mu_t}(\cdot) \\ \boldsymbol{\Theta}_{\mu_t}^Q \in [-B, B]^{p \times N}}}{\arg\min} \frac{1}{n_t^Q} \sum_{j: T_j^Q = t} \left(y_j^Q - h_{\mu_t} \left(\left[\widehat{\boldsymbol{f}}_j^Q, \overline{T}_M((\boldsymbol{\Theta}_{\mu_t}^Q)^\top \boldsymbol{x}_j^Q), \widehat{s}_{\mu_t}(\boldsymbol{x}_j^Q) \right] \right) \right)^2 \\
& \qquad + \lambda \sum_{i,j} \psi_\eta(\boldsymbol{\Theta}_{\mu_t,i,j}^Q),
\end{aligned} \tag{11}$$

where $n_t^Q = \sum_{j=1}^{n^Q} \mathbf{1}(T_j^Q = t)$. The final finetuned estimators for the target domain are

$$\begin{aligned}\widehat{\pi}^Q(\boldsymbol{x}) &= \widehat{h}_\pi\left(\left[p^{-1}(\widehat{\boldsymbol{W}}^{TL})^\top \boldsymbol{x}, \overline{T}_M((\widehat{\boldsymbol{\Theta}}_\pi^Q)^\top \boldsymbol{x}), \widehat{s}_\pi(\boldsymbol{x})\right]\right), \\ \widehat{\mu}_t^Q(\boldsymbol{x}) &= \widehat{h}_{\mu_t}\left(\left[p^{-1}(\widehat{\boldsymbol{W}}^{TL})^\top \boldsymbol{x}, \overline{T}_M((\widehat{\boldsymbol{\Theta}}_{\mu_t}^Q)^\top \boldsymbol{x}), \widehat{s}_{\mu_t}(\boldsymbol{x})\right]\right), \quad t \in \{0,1\},\end{aligned} \tag{12}$$

If $n_t^* = 0$, set $\widehat{\mu}_t^* \equiv 0$ and $\widehat{\Theta}_{\mu_t}^* = 0$, for $* \in \{P, Q\}$. Before AIPW aggregation, the target propensity output is clipped to $[\underline{\alpha}, 1-\underline{\alpha}]$, where $0 < \underline{\alpha} < 1/2$, and the outcome outputs are bounded by M. The TRACE-NN estimator $\widehat{\tau}^Q$ is the AIPW estimator (5) built from these nuisance fits on the full target sample. For the double-robustness result in Section 5, the clipping level also satisfies $\underline{\alpha} \le \alpha$.

C Proof of Theorem 1

Proof of Theorem 1. In this proof of double robustness, all expectations are taken under the target distribution Q. We suppress the domain superscript on target observations, so $(X, T, Y) = (X^Q, T^Q, Y^Q) \sim Q$, and Q_X denotes the marginal law of X. The proof is broken down into two steps. We first establish the population identity that encodes double robustness at the limit nuisance functions. We then show that the sample estimator converges to this population quantity, which completes the proof of consistency.

Step 1. Population double-robust identity at $(\mu_0^{Q,*}, \mu_1^{Q,*}, \pi^{Q,*})$. Start from

$$\mathbb{E}_Q\left[\mu_1^{Q,*}(X) - \mu_0^{Q,*}(X) + \frac{T\{Y - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)} - \frac{(1-T)\{Y - \mu_0^{Q,*}(X)\}}{1-\pi^{Q,*}(X)}\right] - \tau^Q. \tag{13}$$

Using $Y = T\,Y(1) + (1-T)\,Y(0)$,

$$\frac{T\{Y - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)} = \frac{T\{Y(1) - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)}.$$

Conditioning on X and using Assumption 1(i),

$$\begin{aligned}\mathbb{E}_Q\left[\frac{T\{Y - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)} \,\middle|\, X\right] &= \frac{\mathbb{E}_Q[T \mid X]}{\pi^{Q,*}(X)}\mathbb{E}_Q\left[Y(1) - \mu_1^{Q,*}(X) \mid X\right] \\ &= \frac{\pi^Q(X)}{\pi^{Q,*}(X)}\{\mu_1^Q(X) - \mu_1^{Q,*}(X)\}.\end{aligned}$$

Similarly,

$$\mathbb{E}_Q\left[\frac{(1-T)\{Y - \mu_0^{Q,*}(X)\}}{1-\pi^{Q,*}(X)} \,\middle|\, X\right] = \frac{1-\pi^Q(X)}{1-\pi^{Q,*}(X)}\{\mu_0^Q(X) - \mu_0^{Q,*}(X)\}.$$

Substituting these expressions into (13) gives

$$\begin{aligned}&\mathbb{E}_Q\left[\mu_1^{Q,*}(X) - \mu_0^{Q,*}(X) + \frac{T\{Y - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)} - \frac{(1-T)\{Y - \mu_0^{Q,*}(X)\}}{1-\pi^{Q,*}(X)}\right] - \tau^Q \\ &\quad= \mathbb{E}_Q\Big[\mu_1^{Q,*}(X) - \mu_0^{Q,*}(X) - \{\mu_1^Q(X) - \mu_0^Q(X)\} \\ &\qquad + \frac{\pi^Q(X)}{\pi^{Q,*}(X)}\{\mu_1^Q(X) - \mu_1^{Q,*}(X)\} - \frac{1-\pi^Q(X)}{1-\pi^{Q,*}(X)}\{\mu_0^Q(X) - \mu_0^{Q,*}(X)\}\Big] \\ &\quad= \mathbb{E}_Q\Big[(\pi^{Q,*}(X) - \pi^Q(X))\Big\{\frac{\mu_1^{Q,*}(X) - \mu_1^Q(X)}{\pi^{Q,*}(X)} \\ &\qquad + \frac{\mu_0^{Q,*}(X) - \mu_0^Q(X)}{1-\pi^{Q,*}(X)}\Big\}\Big].\end{aligned} \tag{14}$$

The right-hand side of (14) is zero if either

(i) $\mu_t^{Q,*} = \mu_t^Q$ a.s. for $t = 0, 1$, or

(ii) $\pi^{Q,*} = \pi^Q$ a.s.

Therefore, under either condition,

$$\mathbb{E}_Q\left[\mu_1^{Q,*}(X) - \mu_0^{Q,*}(X) + \frac{T\{Y - \mu_1^{Q,*}(X)\}}{\pi^{Q,*}(X)} - \frac{(1-T)\{Y - \mu_0^{Q,*}(X)\}}{1 - \pi^{Q,*}(X)}\right] = \tau^Q. \tag{15}$$

Step 2. Sample-to-population convergence of $\widehat{\tau}^Q$. Write

$$\begin{aligned}\widehat{\tau}^Q - \tau^Q = &\left[\widehat{\mathbb{E}}_Q\left(\widehat{\mu}_1^Q - \widehat{\mu}_0^Q + \frac{T(Y - \widehat{\mu}_1^Q)}{\widehat{\pi}^Q} - \frac{(1-T)(Y - \widehat{\mu}_0^Q)}{1 - \widehat{\pi}^Q}\right)\right. \\ &\left. - \mathbb{E}_Q\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right)\right] \\ &+ \left[\mathbb{E}_Q\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right) - \tau^Q\right].\end{aligned}$$

For any measurable function f,

$$\widehat{\mathbb{E}}_Q[f] := \frac{1}{n^Q}\sum_{i=1}^{n^Q} f(X_i, T_i, Y_i),$$

This defines the empirical target expectation. We write Q_{n^Q} for the corresponding empirical joint distribution of the target observations. Its covariate marginal is Q_{X,n^Q}, as defined in Theorem 1. The second bracket is zero by (15).

For the first bracket, add and subtract

$$\widehat{\mathbb{E}}_Q\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right).$$

This yields

$$\begin{aligned}&\left[\widehat{\mathbb{E}}_Q\left(\widehat{\mu}_1^Q - \widehat{\mu}_0^Q + \frac{T(Y - \widehat{\mu}_1^Q)}{\widehat{\pi}^Q} - \frac{(1-T)(Y - \widehat{\mu}_0^Q)}{1 - \widehat{\pi}^Q}\right)\right. \\ &\qquad\left. - \mathbb{E}_Q\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right)\right] \\ &= \widehat{\mathbb{E}}_Q\left[(\widehat{\mu}_1^Q - \widehat{\mu}_0^Q) - (\mu_1^{Q,*} - \mu_0^{Q,*}) + T\left\{\frac{Y - \widehat{\mu}_1^Q}{\widehat{\pi}^Q} - \frac{Y - \mu_1^{Q,*}}{\pi^{Q,*}}\right\}\right. \\ &\qquad\left. - (1-T)\left\{\frac{Y - \widehat{\mu}_0^Q}{1 - \widehat{\pi}^Q} - \frac{Y - \mu_0^{Q,*}}{1 - \pi^{Q,*}}\right\}\right] \\ &\quad + (\widehat{\mathbb{E}}_Q - \mathbb{E}_Q)\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right).\end{aligned}$$

The fixed-limit score in the last display is square-integrable by the moment and clipping conditions in Theorem 1. Hence the ordinary LLN gives

$$(\widehat{\mathbb{E}}_Q - \mathbb{E}_Q)\left(\mu_1^{Q,*} - \mu_0^{Q,*} + \frac{T(Y - \mu_1^{Q,*})}{\pi^{Q,*}} - \frac{(1-T)(Y - \mu_0^{Q,*})}{1 - \pi^{Q,*}}\right) = o_p(1).$$

It remains to control the first empirical term without treating the fitted nuisances as fixed. Set $d_{\mu,t} := \widehat{\mu}_t^Q - \mu_t^{Q,*}$ and $d_\pi := \widehat{\pi}^Q - \pi^{Q,*}$. Because both propensity functions take values in $[\underline{\alpha}, 1 - \underline{\alpha}]$, direct algebra and the Cauchy–Schwarz inequality give, for

$$\phi(m_0, m_1, e) := m_1 - m_0 + \frac{T(Y - m_1)}{e} - \frac{(1-T)(Y - m_0)}{1 - e},$$

$$\widehat{\mathbb{E}}_Q\left|\phi(\widehat{\mu}_0^Q,\widehat{\mu}_1^Q,\widehat{\pi}^Q)-\phi(\mu_0^{Q,*},\mu_1^{Q,*},\pi^{Q,*})\right| \leq C_{\underline{\alpha}}\sum_{t=0}^{1}\left\{\|d_{\mu,t}\|_{Q_{X,n^Q},2}+\|d_\pi\|_{Q_{X,n^Q},2}\|Y-\mu_t^{Q,*}\|_{Q_{n^Q},2}\right\}, \tag{16}$$

where $\|v\|^2_{Q_{n^Q},2} := (n^Q)^{-1}\sum_{j=1}^{n^Q} v(X_j^Q,T_j^Q,Y_j^Q)^2$. The moment assumptions and the LLN imply $\|Y-\mu_t^{Q,*}\|_{Q_{n^Q},2}=O_p(1)$ for $t\in\{0,1\}$. The empirical-L_2 convergence assumed in Theorem 1 therefore makes the right-hand side of (16) $o_p(1)$. This argument is deterministic conditional on the fitted nuisances and hence remains valid when those nuisances are trained on the same target sample.

Combining the two empirical terms with the population identity (15) gives $\widehat{\tau}^Q-\tau^Q=o_p(1)$, and hence

$$\widehat{\tau}^Q \xrightarrow{p} \tau^Q.$$

□

D Additional Experimental Details

This appendix describes the implementation details of the experiments in Section 4.1: the construction of TRACE-Bench (Appendix D.1), the LLM agent pipelines (Appendix D.2), the statistical ML methods (Appendix D.3), the evaluation protocol and hardware (Appendix D.4), and the complete prompts (Appendix D.5).

D.1 TRACE-Bench Construction

Dataset and outcomes. TRACE-Bench builds on the semi-synthetic moral judgment dataset of Guo et al. (2025), which contains 60,738 posts describing social dilemmas, associated comments, and LLM-generated moral verdicts. The treatment is exposure to the top-upvoted comment ($T=1$) versus a randomly selected comment ($T=0$) on the same post. The dataset provides LLM-generated counterfactual verdicts for each post-comment pair (Marmarelis et al., 2024), binarized as $Y=1$ (fault assigned: YTA or ESH) or $Y=0$ (no fault: NTA or NAH), so that both potential outcomes $Y(1)$ and $Y(0)$ are available as semi-synthetic ground truth.

Topic labeling and clustering. Qwen3.6-27B assigns a short situational topic label to each post from its full story text, excluding the title. BERTopic clusters these labels using all-MiniLM-L6-v2 embeddings, UMAP, and HDBSCAN, yielding 354 topic clusters.

Embedding geometry and domain splitting. Separately, we compute late-chunk story embeddings using Qwen3-Embedding-8B (Günther et al., 2024) and reduce them to 128 dimensions with PCA. These representations define cluster centroids for source–target pairing and provide covariates for treatment simulation. Clusters containing 250–500 posts form the 21 target domains. For each target, we assemble a source domain by adding eligible non-target clusters in decreasing order of centroid distance until it contains at least 5,000 posts. Source clusters must contain at least 10 posts, and source and target domains are disjoint within each task.

Topic labeling

```
Consider this post describing a social dilemma. Label this post with
    the situational
topic of the conflict. The label should be usable as a Python dict key
    :
  - lower case letters and underscores only
  - 2 to 4 words joined by underscore
  - describes the SITUATION (who is involved, what kind of conflict),
    NOT stylistic features (length, tone, age of poster, narration
    style)
```

```
Examples of good labels:
  family_holiday_invite, roommate_chores_dispute,
    wedding_guest_etiquette,
  workplace_coworker_lunch, stranger_grocery_line,
    child_discipline_partner,
  pet_household_responsibility, money_loan_friend,
    neighbor_noise_complaint.

Output STRICT JSON with exactly these two keys, no prose, no preamble:
  {"topic": "<snake_case_topic>", "reason": "<one short sentence>"}

Story:

{story}
```

Treatment simulation. Treatments are simulated once globally by Bernoulli sampling with the calibrated cluster-conditional propensity $\pi_i = \mathrm{clip}(\sigma(w^\top z_i + c_{C_i}), 0.05, 0.95)$, where z_i is the 128-dimensional PCA projection of the late-chunk story embedding of post i and C_i is its topic cluster. We draw $w_j \overset{\text{iid}}{\sim} \mathrm{Unif}(-a, a)$ with $a = 0.5/\sqrt{128\bar{\lambda}}$, where $\bar{\lambda}$ is the mean of the retained PCA eigenvalues, and draw cluster offsets $c_k \overset{\text{iid}}{\sim} \mathrm{Unif}(-0.75, 0.75)$. We require mean propensities within $[0.15, 0.85]$ in at least 95% of clusters and a clipping fraction no greater than 5% in every cluster. The initial parameter draw generated with seed 2026 satisfied both criteria without rescaling. Treatment assignments are generated with seed 2026, and observed outcomes are determined by the assigned treatment and the corresponding potential-outcome labels. All 21 tasks reuse this single global realization of treatments and observed outcomes.

Bag-of-words features. For statistical ML methods on TRACE-Bench, we represent each post using bag-of-words counts computed with scikit-learn's `CountVectorizer`. We fit a shared vocabulary on the full story bodies of all 60,738 posts, excluding titles and without truncation. The default vectorizer lowercases text, extracts unigrams containing at least two word characters, retains stop words, and includes terms appearing in at least one document. This produces 67,398-dimensional raw-count vectors stored as `float32`.

D.2 LLM Agent Implementation Details

TRACE-AGENT. TRACE-AGENT learns separate natural-language contexts for propensity and joint outcome prediction (μ_0, μ_1), keeping model weights and prediction templates fixed during source learning and target finetuning. Each configuration uses the same foundation model for reflection and prediction. The reported method supplies the complete post as x. A label-blind extraction front end can replace the full text when context limits require shorter inputs, but this extension is not used in the reported experiments.

- **Prompt optimization.** We learn separate propensity and joint outcome contexts using the reflective search in Algorithm 1. The reflection agent receives minibatch error examples and can inspect the entire D_{learn} through `sample_rows`, `read_file`, `grep`, and `python`. Source validation contains $\min(128, \lfloor 0.1 N_P \rfloor)$ examples, with the remainder used for learning. Splits are stratified by T for propensity and (T, Y) for outcomes. Appendix D.5.1 provides the optimization prompts and initial contexts.
- **Target context finetuning.** The target finetuning procedure is specified in Algorithm 1 with the same prompt optimization algorithm plus K-fold target cross-fitting, using gate margin δ to determine whether to retain source predictions or optimize a target correction context. We set $K = 3$ and $\delta = 0.01$. Each training-fold complement contributes $\min(48, \lfloor 0.2 N_{-k} \rfloor)$ validation examples, with the remainder used for reflection and the same stratification as above. Target initial contexts include aggregate gate diagnostics computed over the entire complement, while the reflection agent's CSV contains only learning examples. Target predictor queries additionally receive the post's three source nuisance estimates. Appendix D.5.1 provides the finetuning prompts and initial contexts.

- **Agent budgets and estimation.** Reflection uses an ordinary tool loop with a 12-call budget checked between assistant turns and up to two output-format reminders. Each reflection completion allows 8,192 tokens (32,768 for GPT-OSS-20B and GPT-OSS-120B). Revised contexts are requested to stay within approximately 1,500 words; the parser enforces a 20,000-character limit. Nuisance prediction uses no tools and allows 8,192 tokens per completion, including reasoning. Predictions are elicited as integer percentages, clipped to $[1, 99]$, and divided by 100; unparseable outputs receive 0.5 for scoring and final estimation. Generation seeds remain fixed per post across candidate contexts within each run. We combine held-out nuisance predictions using AIPW, clipping denominator propensities to $[0.1, 0.9]$. Appendix D.5.1 gives the estimator prompts.

Baseline agents. The Direct estimation, key information retrieval (KIR), and Plan-and-execute baselines introduced in Section 4.1 use the following implementation settings. Appendix D.5.2 provides their complete prompts.

- **Plan-and-execute.** The planner and executor use the same foundation model, maintain separate conversation histories, and share a working directory within each run. The `read_file` tool returns a range of numbered lines, `grep` supports regular-expression searches, and `python` executes code in the shared directory. The model receives tool outputs and execution errors so that it can correct failed operations. Each stage allows 100 tool calls. If the planner stops without a usable plan or the executor stops without a finite numeric ATE estimate, that stage receives up to three continuation prompts. Once a stage exhausts its tool budget, it must produce a final response without further tool calls. The planner generates a new plan for each repetition.

- **Direct estimation.** The model is prompted to estimate the target ATE from the observed source and target datasets, in which covariates are provided as raw text. It inspects the records through a file-reading interface: each call to `read(file, start, n)` returns consecutive CSV rows with all columns, up to an output-size limit, along with the next row index. Each run allows up to 128 tool calls. When the conversation and reserved generation budget would exceed the context window, the model compacts its history into notes recording the row ranges already read, running counts, relevant observations, current estimate, and remaining work. The conversation restarts with the original instructions and these notes, retaining the tool-call count. Older tool outputs are truncated as needed to fit the request within the context window. If an estimate is missing, the model receives up to two continuation prompts. Once it exhausts its tool budget, it must produce a final response without further tool calls.

- **KIR.** The extraction model receives only the story text and study description; treatment and outcome labels are withheld. Each post is represented by at most three phrases, with at most four words per phrase and 50 characters in total. We normalize and deduplicate the phrases, remove explicit verdict or label mentions, and replace unusable extractions with `unknown`. We retain all source and target rows, serialize them as `phrases|T=...|Y=...`, and present the two populations separately. Before estimation, we check that the complete prompt fits the input-token budget. We cache extractions across tasks and repetitions and independently randomize row order for each estimation run.

Foundation model serving. We locally hosted six foundation models through vLLM. DeepSeek-V4-Flash-0731 (Xu et al., 2026), GPT-OSS-120B and GPT-OSS-20B (Agarwal et al., 2025), Gemma-4-12B-it (Team et al., 2026), and Qwen3-4B-Instruct-2507 (Yang et al., 2025) were served from their native Hugging Face checkpoints, and Qwen3.6-27B (Qwen Team, 2026) used its FP8 checkpoint. Reasoning was enabled by default for all models except the instruct-only Qwen3-4B, with high reasoning effort for DeepSeek and GPT-OSS. We followed model-card-recommended decoding presets when available. The $(\text{temperature}, \text{top-}p, \text{top-}k)$ settings were $(1.0, 0.95, \text{disabled})$ for DeepSeek, $(1.0, 1.0, \text{disabled})$ for GPT-OSS, $(0.6, 0.95, 20)$ for Qwen3.6-27B, $(1.0, 0.95, 64)$ for Gemma, and $(0.7, 0.8, 20)$ for Qwen3-4B.

Table 3: TRACE-NN hyperparameter settings for the simulation, News, and TRACE-Bench experiments. In the simulations, the target unlabeled subset size is $m^Q = 50$ for $n^Q = 500$ and $m^Q = 100$ for $n^Q \in \{1000, 1500, \ldots, 5000\}$.

Hyperparameter	Simulation	News	TRACE-Bench
Covariate dimension p	$\{1000, 2500, 5000, 10000\}$	3477	67398
Empirical factor dimension $\bar{r}$	10	10	20
Source borrowing threshold δ_x	1.0	0.1	1.0
Unlabeled subset m^P	100	100	100
Unlabeled subset m^Q	$\{50, 100\}$	25	25

D.3 Statistical ML Implementation Details

Network architecture. All TRACE-NN experiments (Table 1, Figure 5, and Table 6) use its two-stage source pretraining and target finetuning procedure (Appendix B.3). Following the notation in Appendix B.2, source pretraining uses depth $L^P = 4$ and width $N^P = 300$, while target finetuning uses depth $L = 4$ and width $N = 150$. The empirical factor dimension $\bar{r}$, source borrowing threshold δ_x, and reserved unlabeled subset sizes m^P and m^Q are dataset-specific and reported in Table 3.

Parameter accounting. We report the model size of the TRACE-NN configuration evaluated in Table 1, which includes the source and target networks for the propensity and outcome systems. Summing the trainable parameters across the source and target stages gives

$$N_{\text{trainable}} = 3\big(20{,}586{,}901 + 10{,}203{,}601\big) = 92{,}371{,}506.$$

This count includes the learned source parameters even when they are frozen during target adaptation. Including the diversified projection matrices registered in the networks gives

$$N_{\text{total}} = 3\big(21{,}934{,}861 + 11{,}551{,}561\big) = 100{,}459{,}266.$$

The two terms inside each pair of parentheses correspond to the source and target networks, respectively. The comparison with the nominal 4B parameters of Qwen3-4B uses N_{total}, yielding a size ratio of approximately 40.

Regularization and optimization. The clipped-ℓ_1 penalty ψ_η in (10) regularizes the variable selection matrix Θ using a clipping threshold $\eta = 0.005$ and an adaptive penalty weight $\lambda = 1.5\log(p)/n_{\text{train}}$. The clipping threshold is annealed from 20η to η over the course of training. All networks are trained with the `Adam` optimizer at an initial learning rate 10^{-4}, batch size 64, and 100 epochs, using `ReduceLROnPlateau` scheduling and early stopping.

Baseline implementation. FIDDLE (source-only), FIDDLE (target-only), Finetune Vanilla-NN, and Finetune LASSO are ablations of TRACE-NN sharing the same estimation pipeline and comparable hyperparameters. FIDDLE (source-only) and FIDDLE (target-only) each use the FAST-NN architecture matching the corresponding stage in TRACE-NN. Finetune Vanilla-NN replaces FAST-NN in both stages with a plain ReLU network of the same depth and width. Finetune LASSO replaces all neural network components with ℓ_1-penalized linear regression (LASSO) while retaining the AIPW estimation step. For external transfer baselines ℓ_1-TCL (TARNet) and ℓ_1-TCL (DragonNet), we use the recommended architectures and default hyperparameters from the original papers and released implementations.

D.4 Evaluation Protocol and Hardware

Evaluation protocol. All reported methods follow the same experimental protocol and are compared under identical data splits and random seeds. We report RMSE across the 21 tasks with task-level bootstrap standard errors computed as the sample standard deviation of RMSE over $B = 10{,}000$ bootstrap replicates, each obtained by sampling 21 tasks with replacement.

Hardware. All LLM experiments were conducted on NVIDIA H200 GPUs with 141GB of GPU memory each. DeepSeek-V4-Flash experiments used two GPUs per job, while all other LLM models used one. All neural-network-based statistical baselines ran on NVIDIA A100 GPU resources. TRACE-Bench and News experiments used 10GB Multi-Instance GPU (MIG) instances. Finetune LASSO ran on four CPU cores with 32GB of system memory.

D.5 Complete Prompts

We provide the complete prompts for TRACE-AGENT and the LLM baselines described in Appendix D.2. Placeholders in angle brackets or braces are filled at run time.

D.5.1 TRACE-AGENT Prompts

Predictor prompts. Source predictors use the learned context as the system message and the remaining text as the user message. These templates serve both context evaluation and prediction on target posts.

Source estimator: propensity

```
<LEARNED_SOURCE_PROPENSITY_CONTEXT>

Now here is a story:
---
<FULL_POST_TEXT>
---

An automated process decided, based on the post's content, whether the
     reviewer judging this post was shown the post's top-voted comment
     (T=1) or a random comment (T=0).
Estimate the probability that this post was assigned T=1 (reviewer
    shown the top-voted comment).
Answer strict JSON only: {"pi": <integer 0-100>}
```

Source estimator: outcomes

```
<LEARNED_SOURCE_OUTCOME_CONTEXT>

Now here is a story:
---
<FULL_POST_TEXT>
---

Estimate the probability that the reviewer's verdict on this post
    finds fault -- YTA or ESH, coded Y=1 (NTA / NAH count as no fault,
     Y=0):
(a) mu1: if the reviewer was shown the post's top-voted comment (T=1);
(b) mu0: if the reviewer was shown a random comment (T=0).
Answer strict JSON only: {"mu0": <integer 0-100>, "mu1": <integer
    0-100>}
```

When the gate selects adaptation, target predictors use the learned context and fixed transfer note as the system message; the remaining text forms the user message. Source estimates are integer percentages.

Target estimator: propensity

```
<LEARNED_TARGET_PROPENSITY_CONTEXT>
```

```
Note: each query below also carries estimates from a model fitted on
    the SOURCE segment for the same story; use the knowledge above to
    correct them for the TARGET segment.

Now here is a story from the TARGET segment:
---
<FULL_POST_TEXT>
---

A model fitted on the source segment estimates for this story: pi=<
    SOURCE_PI_PERCENT>%, mu1=<SOURCE_MU1_PERCENT>%, mu0=<
    SOURCE_MU0_PERCENT>%.

An automated process decided, based on the post's content, whether the
     reviewer judging this post was shown the post's top-voted comment
     (T=1) or a random comment (T=0).
Estimate the probability that this post was assigned T=1 (reviewer
    shown the top-voted comment).
Using the transfer knowledge above, correct the source model's
    estimates for the TARGET segment.
Answer strict JSON only: {"pi": <integer 0-100>}
```

Target estimator: outcomes

```
<LEARNED_TARGET_OUTCOME_CONTEXT>

Note: each query below also carries estimates from a model fitted on
    the SOURCE segment for the same story; use the knowledge above to
    correct them for the TARGET segment.

Now here is a story from the TARGET segment:
---
<FULL_POST_TEXT>
---

A model fitted on the source segment estimates for this story: pi=<
    SOURCE_PI_PERCENT>%, mu1=<SOURCE_MU1_PERCENT>%, mu0=<
    SOURCE_MU0_PERCENT>%.

Estimate the probability that the reviewer's verdict on this post
    finds fault -- YTA or ESH, coded Y=1 (NTA / NAH count as no fault,
     Y=0):
(a) mu1: if the reviewer was shown the post's top-voted comment (T=1);
(b) mu0: if the reviewer was shown a random comment (T=0).
Using the transfer knowledge above, correct the source model's
    estimates for the TARGET segment.
Answer strict JSON only: {"mu0": <integer 0-100>, "mu1": <integer
    0-100>}
```

Initial contexts. These source contexts initialize the nuisance predictors' system messages before prompt evolution.

Source initial context: propensity

```
You estimate treatment-assignment probabilities for posts from a semi-
    synthetic moral judgment dataset. For each post, an automated
    process decided, based on the post's content, whether the reviewer
     judging the post was shown the post's top-voted comment (T=1) or
```

```
    a random comment (T=0). Your job is to read the post text and
    estimate the probability that this post was assigned T=1; report a
     calibrated percentage, not a hard classification.
```

Source initial context: outcomes

```
You estimate moral-verdict probabilities for posts from a semi-
    synthetic moral judgment dataset. A reviewer reads each post and
    delivers a verdict that either finds fault (YTA 'you're the
    asshole' or ESH 'everyone sucks here' -- coded Y=1) or finds no
    fault (NTA 'not the asshole' or NAH 'no assholes here' -- coded Y
    =0); before judging, an automated process decided, based on the
    post's content, whether the reviewer judging the post was shown
    the post's top-voted comment (T=1) or a random comment (T=0). Your
     job is to estimate the probability of a FAULT verdict (Y=1) under
     each condition -- mu1 if the reviewer was shown the top-voted
    comment, mu0 if shown a random comment -- as calibrated
    percentages.
```

Target contexts are initialized when the gate selects adaptation. Gate diagnostics are specific to each fold and nuisance arm. Outcome diagnostics use the corresponding treatment subgroup and omit empty subgroups.

Target initial context: propensity

```
You estimate treatment-assignment probabilities for posts from a semi-
    synthetic moral judgment dataset. For each post, an automated
    process decided, based on the post's content, whether the reviewer
     judging the post was shown the post's top-voted comment (T=1) or
    a random comment (T=0). Your job is to read the post text and
    estimate the probability that this post was assigned T=1; report a
     calibrated percentage, not a hard classification. These posts
    come from a TARGET segment -- a different group of topic clusters
    from the same dataset -- than the SOURCE segment the accompanying
    model estimates were fitted on; correct those estimates for the
    target segment rather than predicting from scratch.

Gate diagnostics measured on labelled posts from this TARGET community
     (n=<N_GATE>), comparing the source model's estimates against the
    observed outcomes:
- Source estimates score a mean log-loss of <SOURCE_LOG_LOSS>, versus
    <BASE_RATE_LOG_LOSS> for a constant predictor that only knows this
     community's overall rate. They did not beat that baseline by the
    required 0.010.
- pi: the source model averages <PI_MEAN_PERCENT>% on these <PI_N>
    posts while the observed rate is <PI_OBSERVED_PERCENT>% -- <
    TOO_HIGH_OR_TOO_LOW> by <ABSOLUTE_GAP> percentage points.
Use these facts to correct the source estimates for this community;
    state the corrections you apply as concrete, checkable rules.
```

Target initial context: outcomes

```
You estimate moral-verdict probabilities for posts from a semi-
    synthetic moral judgment dataset. A reviewer reads each post and
    delivers a verdict that either finds fault (YTA 'you're the
    asshole' or ESH 'everyone sucks here' -- coded Y=1) or finds no
    fault (NTA 'not the asshole' or NAH 'no assholes here' -- coded Y
```

```
    =0); before judging, an automated process decided, based on the
    post's content, whether the reviewer judging the post was shown
    the post's top-voted comment (T=1) or a random comment (T=0). Your
     job is to estimate the probability of a FAULT verdict (Y=1) under
     each condition -- mu1 if the reviewer was shown the top-voted
    comment, mu0 if shown a random comment -- as calibrated
    percentages. These posts come from a TARGET segment -- a different
     group of topic clusters from the same dataset -- than the SOURCE
    segment the accompanying model estimates were fitted on; correct
    those estimates for the target segment rather than predicting from
     scratch.

Gate diagnostics measured on labelled posts from this TARGET community
     (n=<N_GATE>), comparing the source model's estimates against the
    observed outcomes:
- Source estimates score a mean log-loss of <SOURCE_LOG_LOSS>, versus
    <BASE_RATE_LOG_LOSS> for a constant predictor that only knows this
     community's overall rate. They did not beat that baseline by the
    required 0.010.
- mu1: the source model averages <MU1_MEAN_PERCENT>% on these <MU1_N>
    posts while the observed rate is <MU1_OBSERVED_PERCENT>% -- <
    TOO_HIGH_OR_TOO_LOW> by <MU1_ABSOLUTE_GAP> percentage points.
- mu0: the source model averages <MU0_MEAN_PERCENT>% on these <MU0_N>
    posts while the observed rate is <MU0_OBSERVED_PERCENT>% -- <
    TOO_HIGH_OR_TOO_LOW> by <MU0_ABSOLUTE_GAP> percentage points.
Use these facts to correct the source estimates for this community;
    state the corrections you apply as concrete, checkable rules.
```

Reflection prompts. Source reflection prompts concatenate the system and user messages. The nested `[SYSTEM]` and `[USER]` markers are literal text in the predictor description. Outcome predictions are scored against observed outcomes only for the observed treatment.

Source prompt optimization: propensity

```
You are improving the instruction CONTEXT of a fixed LLM predictor
    used inside a causal-inference pipeline on posts in TRACE-Bench.
    The predictor's prompt template is frozen; the only thing you can
    change is the context block that prefixes every query.
You have tools over a working directory holding the data the predictor
     was scored on: use sample_rows / read_file / grep for quick
    inspection and the python tool (pandas) for whole-dataset
    computation. Never paste whole files into your reply.
When you are done, output ONLY the revised context between <context>
    and </context> tags. It must be self-contained (it REPLACES the
    current context entirely), at most ~1500 words, and must not
    mention these instructions, the tools, or the files.

Predictor description:
The predictor is ONE LLM call per post with a FIXED template; the only
     thing you can change is the context, which becomes the entire
    system message. Template:

[SYSTEM]
<<< THE CONTEXT YOU ARE WRITING GOES HERE >>>

[USER]
Now here is a story:
---
<<< the full text of one post >>>
---
```

```
An automated process decided, based on the post's content, whether the
     reviewer judging this post was shown the post's top-voted comment
     (T=1) or a random comment (T=0).
Estimate the probability that this post was assigned T=1 (reviewer
    shown the top-voted comment).
Answer strict JSON only: {"pi": <integer 0-100>}

Scoring: mean binary log-loss of the elicited pi against the true T (
    treatment assignment), on a held-out set of posts you do not see.
    Lower is better, and calibration matters more than confident
    classification.

Current context (the candidate being revised):
<current_context>
<CURRENT_CONTEXT>
</current_context>

Worst failures of this context on a scored minibatch (one JSON object
    per line; higher loss = worse):
<HIGHEST_LOSS_TRAINING_EXAMPLES_AS_JSON_LINES>

Files in your working directory: d_learn_pi.csv

Diagnose WHY the current context produced these failures -- browse the
     data with your tools where helpful -- then write an improved
    context. Keep what works, fix what fails, and prefer concrete,
    checkable guidance (base rates, decision rules, calibration
    anchors) over platitudes.
The context is applied to UNSEEN posts, so it must generalise:
  - never mention a post id, row number, or any other identifier;
  - never write a rule that fires on one specific post -- a rule must
    describe a class you could point to at least 10 examples of;
  - never quote a rate computed from fewer than 10 rows; compute rates
     over the whole file with the python tool, or state them
    qualitatively;
  - the failures above are the WORST rows, not a sample -- do not
    infer base rates from them;
  - do not add a rule that contradicts one you are keeping.
Output ONLY the revised context inside <context>...</context>.
```

Source prompt optimization: outcomes

```
You are improving the instruction CONTEXT of a fixed LLM predictor
    used inside a causal-inference pipeline on posts in TRACE-Bench.
    The predictor's prompt template is frozen; the only thing you can
    change is the context block that prefixes every query.
You have tools over a working directory holding the data the predictor
     was scored on: use sample_rows / read_file / grep for quick
    inspection and the python tool (pandas) for whole-dataset
    computation. Never paste whole files into your reply.
When you are done, output ONLY the revised context between <context>
    and </context> tags. It must be self-contained (it REPLACES the
    current context entirely), at most ~1500 words, and must not
    mention these instructions, the tools, or the files.

Predictor description:
The predictor is ONE LLM call per post with a FIXED template; the only
     thing you can change is the context, which becomes the entire
    system message. Template:
```

```
[SYSTEM]
<<< THE CONTEXT YOU ARE WRITING GOES HERE >>>

[USER]
Now here is a story:
---
<<< the full text of one post >>>
---

Estimate the probability that the reviewer's verdict on this post
    finds fault -- YTA or ESH, coded Y=1 (NTA / NAH count as no fault,
     Y=0):
(a) mu1: if the reviewer was shown the post's top-voted comment (T=1);
(b) mu0: if the reviewer was shown a random comment (T=0).
Answer strict JSON only: {"mu0": <integer 0-100>, "mu1": <integer
    0-100>}

Scoring: mean binary log-loss of the elicited mu1 against Y on T==1
    rows and mu0 against Y on T==0 rows, on a held-out set of posts
    you do not see. Lower is better, and calibration matters more than
     confident classification.

Current context (the candidate being revised):
<current_context>
<CURRENT_CONTEXT>
</current_context>

Worst failures of this context on a scored minibatch (one JSON object
    per line; higher loss = worse):
<HIGHEST_LOSS_TRAINING_EXAMPLES_AS_JSON_LINES>

Files in your working directory: d_learn_mu.csv

Diagnose WHY the current context produced these failures -- browse the
     data with your tools where helpful -- then write an improved
    context. Keep what works, fix what fails, and prefer concrete,
    checkable guidance (base rates, decision rules, calibration
    anchors) over platitudes.
The context is applied to UNSEEN posts, so it must generalise:
  - never mention a post id, row number, or any other identifier;
  - never write a rule that fires on one specific post -- a rule must
    describe a class you could point to at least 10 examples of;
  - never quote a rate computed from fewer than 10 rows; compute rates
     over the whole file with the python tool, or state them
    qualitatively;
  - the failures above are the WORST rows, not a sample -- do not
    infer base rates from them;
  - do not add a rule that contradicts one you are keeping.
Output ONLY the revised context inside <context>...</context>.
```

In the target propensity-finetuning prompt, the 50% values illustrate the predictor format; actual queries use source estimates for the corresponding post. The outcome context guides both potential-outcome predictions, but validation uses only factual outcomes.

Target context finetuning: propensity

```
You are improving the instruction CONTEXT of a fixed LLM predictor
    used inside a causal-inference pipeline on posts in TRACE-Bench.
    The predictor's prompt template is frozen; the only thing you can
    change is the context block that prefixes every query.
```

```
You have tools over a working directory holding the data the predictor
     was scored on: use sample_rows / read_file / grep for quick
    inspection and the python tool (pandas) for whole-dataset
    computation. Never paste whole files into your reply.
When you are done, output ONLY the revised context between <context>
    and </context> tags. It must be self-contained (it REPLACES the
    current context entirely), at most ~1500 words, and must not
    mention these instructions, the tools, or the files.

Predictor description:
The predictor is ONE LLM call per post with a FIXED template; the only
     thing you can change is the context, which becomes the entire
    system message. Template:

[SYSTEM]
<<< THE CONTEXT YOU ARE WRITING GOES HERE >>>

Note: each query below also carries estimates from a model fitted on
    the SOURCE segment for the same story; use the knowledge above to
    correct them for the TARGET segment.

[USER]
Now here is a story from the TARGET segment:
---
<<< the full text of one post >>>
---

A model fitted on the source segment estimates for this story: pi=50%,
     mu1=50%, mu0=50%.

An automated process decided, based on the post's content, whether the
     reviewer judging this post was shown the post's top-voted comment
     (T=1) or a random comment (T=0).
Estimate the probability that this post was assigned T=1 (reviewer
    shown the top-voted comment).
Using the transfer knowledge above, correct the source model's
    estimates for the TARGET segment.
Answer strict JSON only: {"pi": <integer 0-100>}

(The source-model percentages above are illustrative; each query
    carries that post's real values.)

Scoring: mean binary log-loss of the elicited pi against the true T (
    treatment assignment), on a held-out set of posts you do not see.
    Lower is better, and calibration matters more than confident
    classification.

Current context (the candidate being revised):
<current_context>
<CURRENT_CONTEXT>
</current_context>

Worst failures of this context on a scored minibatch (one JSON object
    per line; higher loss = worse):
<HIGHEST_LOSS_TRAINING_EXAMPLES_AS_JSON_LINES>

Files in your working directory: d_learn_pi.csv

Diagnose WHY the current context produced these failures -- browse the
     data with your tools where helpful -- then write an improved
    context. Keep what works, fix what fails, and prefer concrete,
    checkable guidance (base rates, decision rules, calibration
    anchors) over platitudes.
```

```
The context is applied to UNSEEN posts, so it must generalise:
  - never mention a post id, row number, or any other identifier;
  - never write a rule that fires on one specific post -- a rule must
    describe a class you could point to at least 10 examples of;
  - never quote a rate computed from fewer than 10 rows; compute rates
     over the whole file with the python tool, or state them
    qualitatively;
  - the failures above are the WORST rows, not a sample -- do not
    infer base rates from them;
  - do not add a rule that contradicts one you are keeping.
Output ONLY the revised context inside <context>...</context>.
```

Target context finetuning: outcomes

```
You are improving the instruction CONTEXT of a fixed LLM predictor
    used inside a causal-inference pipeline on posts in TRACE-Bench.
    The predictor's prompt template is frozen; the only thing you can
    change is the context block that prefixes every query.
You have tools over a working directory holding the data the predictor
     was scored on: use sample_rows / read_file / grep for quick
    inspection and the python tool (pandas) for whole-dataset
    computation. Never paste whole files into your reply.
When you are done, output ONLY the revised context between <context>
    and </context> tags. It must be self-contained (it REPLACES the
    current context entirely), at most ~1500 words, and must not
    mention these instructions, the tools, or the files.

Predictor description:
The predictor is ONE LLM call per post with a FIXED template; the only
     thing you can change is the context, which becomes the entire
    system message. Template:

[SYSTEM]
<<< THE CONTEXT YOU ARE WRITING GOES HERE >>>

Note: each query below also carries estimates from a model fitted on
    the SOURCE segment for the same story; use the knowledge above to
    correct them for the TARGET segment.

[USER]
Now here is a story from the TARGET segment:
---
<<< the full text of one post >>>
---

A model fitted on the source segment estimates for this story: pi=50%,
     mu1=50%, mu0=50%.

Estimate the probability that the reviewer's verdict on this post
    finds fault -- YTA or ESH, coded Y=1 (NTA / NAH count as no fault,
     Y=0):
(a) mu1: if the reviewer was shown the post's top-voted comment (T=1);
(b) mu0: if the reviewer was shown a random comment (T=0).
Using the transfer knowledge above, correct the source model's
    estimates for the TARGET segment.
Answer strict JSON only: {"mu0": <integer 0-100>, "mu1": <integer
    0-100>}

(The source-model percentages above are illustrative; each query
    carries that post's real values.)
```

```
Scoring: mean binary log-loss of the elicited mu1 against Y on T==1
    rows and mu0 against Y on T==0 rows, on a held-out set of posts
    you do not see. Lower is better, and calibration matters more than
     confident classification.

Current context (the candidate being revised):
<current_context>
<CURRENT_CONTEXT>
</current_context>

Worst failures of this context on a scored minibatch (one JSON object
    per line; higher loss = worse):
<HIGHEST_LOSS_TRAINING_EXAMPLES_AS_JSON_LINES>

Files in your working directory: d_learn_mu.csv

Diagnose WHY the current context produced these failures -- browse the
     data with your tools where helpful -- then write an improved
    context. Keep what works, fix what fails, and prefer concrete,
    checkable guidance (base rates, decision rules, calibration
    anchors) over platitudes.
The context is applied to UNSEEN posts, so it must generalise:
  - never mention a post id, row number, or any other identifier;
  - never write a rule that fires on one specific post -- a rule must
    describe a class you could point to at least 10 examples of;
  - never quote a rate computed from fewer than 10 rows; compute rates
     over the whole file with the python tool, or state them
    qualitatively;
  - the failures above are the WORST rows, not a sample -- do not
    infer base rates from them;
  - do not add a rule that contradicts one you are keeping.
Output ONLY the revised context inside <context>...</context>.
```

D.5.2 BASELINE PROMPTS

Plan-and-execute. The executor's <PLAN> placeholder contains the plan generated in the current run.

Plan-and-execute: planner

```
You are an expert causal-inference statistician acting as the PLANNER.

Estimand: the Average Treatment Effect (ATE) on the TARGET population.
Each row represents one post from a semi-synthetic moral judgment
    dataset:
  - `story`   : the full raw post text. This is the ONLY covariate.
  - `T`       : binary treatment in {0, 1}.
  - `Y`       : observed binary outcome in {0, 1}.
  - `post_id` : an opaque id (not a covariate).
ATE = E[Y(1) - Y(0)] over the target population.

The data is OBSERVATIONAL: treatment assignment T may depend on the
    post content, so the naive difference of means E[Y|T=1] - E[Y|T=0]
     is confounded and biased. You must account for the confounding
    induced by the post content (`story`). Only the observed (T, Y)
    and the text are available -- there are no counterfactual outcomes
     or true assignment probabilities.

Two files are present in the sandbox:
```

```
  - `source.csv` : a larger, related but DISTRIBUTION-SHIFTED
    population.
  - `target.csv` : the population whose ATE you must estimate.
You MAY use `source.csv` to improve estimation via transfer learning,
    but the estimand is the TARGET ATE computed on `target.csv`.

You have tools (read_file, grep, python) operating inside a sandbox
    directory that holds the dataset CSV file(s). Use them to inspect
    the data BEFORE planning.

Your job is ONLY to produce a concrete, step-by-step PLAN that a
    separate, less capable EXECUTOR model will follow to estimate the
    target ATE. The plan must:
  - name exactly which file(s) and columns to use;
  - give a concrete, step-by-step procedure for estimating the target
    ATE that correctly accounts for the confounding described above;
  - be explicit and self-contained enough for a less capable executor
    to follow;
  - finish by instructing the executor to satisfy this contract:
When finished, you MUST write a file `result.json` into the sandbox
    directory containing a JSON object:
    {"ate": <float>, "method": "<short description>", "notes": "<
    optional>"}
and also print a final line exactly of the form:
    FINAL_ATE: <value>
where <value> is the numeric target-ATE estimate.

Also require these implementation safeguards:
Implementation requirements for the final computation:
  - Use the `python` tool for all computation. Put valid multi-line
    Python code directly in the tool's `code` field.
  - Do not use shell commands, heredocs, `python -c`, or command-
    string quoting.
  - Keep the final computation in one self-contained `python` tool
    call.
  - Before printing FINAL_ATE, verify the ATE is finite, write result.
    json with json.dump(..., allow_nan=False), then read it back with
    json.load.

Do NOT compute the ATE yourself. Inspect, then output the plan as your
     final message.

The sandbox directory contains these files: ['source.csv', 'target.csv
    '].
Inspect the data with your tools, then write the executor's plan.
```

Plan-and-execute: executor

```
You are the executor of the ATE estimation tasks for the target
    dataset.
Here is the plan: <PLAN>

You operate inside a sandbox directory containing: ['source.csv', '
    target.csv']. Use the tools (read_file, grep, python) to carry out
     the plan. Follow the plan faithfully.

When finished, you MUST write a file `result.json` into the sandbox
    directory containing a JSON object:
    {"ate": <float>, "method": "<short description>", "notes": "<
    optional>"}
and also print a final line exactly of the form:
```

```
    FINAL_ATE: <value>
where <value> is the numeric target-ATE estimate.

Implementation requirements for the final computation:
  - Use the `python` tool for all computation. Put valid multi-line
    Python code directly in the tool's `code` field.
  - Do not use shell commands, heredocs, `python -c`, or command-
    string quoting.
  - Keep the final computation in one self-contained `python` tool
    call.
  - Before printing FINAL_ATE, verify the ATE is finite, write result.
    json with json.dump(..., allow_nan=False), then read it back with
    json.load.

Begin executing the plan now and report the final ATE.
```

Direct estimation.

Direct estimation

```
You are estimating the average treatment effect (ATE).

Available read-only tool: `read`.
Use it to inspect the provided dataset or datasets.

When finished, reply exactly:
<answer>{"ate": <finite number>}</answer>
Do not write anything after the closing tag.

The sandbox contains:
- `source.csv`: the source dataset.
- `target.csv`: the target dataset.

Estimate the average treatment effect (ATE) for the target dataset.
```

Key information retrieval (KIR). KIR uses separate extraction and estimation prompts. Placeholders contain the input story, serialized source and target records, and their row counts.

KIR: extraction

```
You extract key information from posts describing social dilemmas for
    an observational causal study. You see ONE post's story and never
    its labels.

Study context: for each post, an automated process decided, based on
    the post's content, whether the reviewer judging the post was
    shown the post's top-voted comment (T=1) or a random comment (T=0)
    . The outcome is the reviewer's verdict: fault (YTA/ESH, Y=1) or
    no fault (NTA/NAH, Y=0).

Distill the story into 2-3 short key phrases (each at most 4 words)
    capturing the concrete facts a causal analysis would need: the
    situation, the people involved, what the poster did, and anything
    treatment assignment or a reviewer's moral judgment could turn on
    (tone, severity, relationships, stakes, repeated behavior).

Rules:
- Describe the story text only. Never predict or name T, Y, a verdict
    (YTA/NTA/ESH/NAH), or a population segment.
```

```
- Each phrase: at most 4 words, lowercase, no punctuation, never "|".
- Answer strict JSON only: {"phrases": ["<phrase>", ...]}

Story:
<STORY>
```

KIR: estimation

```
You are estimating the average treatment effect (ATE).

The entire dataset is given in this one message. There are no tools
    and no second turn. Each line is one row:
    <text>|T=<0 or 1>|Y=<0 or 1>

When finished, reply exactly:
{"ate": <finite number>}
Write nothing else.

The message contains:
- SOURCE DATA: the source dataset.
- TARGET DATA: the target dataset.

Estimate the average treatment effect (ATE) for the target dataset.

=== SOURCE DATA: <N_SOURCE> rows ===
<SOURCE_ROWS>
=== END SOURCE DATA ===

=== TARGET DATA: <N_TARGET> rows ===
<TARGET_ROWS>
=== END TARGET DATA ===
```

E Additional Experimental Results

In this appendix, we first analyze the baseline LLM agents on TRACE-Bench (Appendix E.1) and evaluate the statistical ML methods with text embeddings instead of bag-of-words covariates (Appendix E.2), then report two experiments evaluating TRACE-NN beyond TRACE-Bench: a controlled simulation with tunable covariate dimension p and target sample size n^Q (Appendix E.3), and the semi-synthetic *News* benchmark of Johansson et al. (2016) with bag-of-words text covariates and designed ground truth (Appendix E.4). The latter two experiments assess the statistical foundation of our method under controlled conditions and use statistical ML baselines.

E.1 Baseline agent analysis

Direct estimation and key information retrieval. The difference-in-means (DiM) estimator subtracts the mean outcome of target-domain control units from that of treated units, $\bar{Y}_{T=1} - \bar{Y}_{T=0}$. It achieves an RMSE of 0.0375 without covariate information or confounding adjustment, lower than the reported RMSE of 16 of the 18 baseline agent configurations in Table 1. Both direct estimation from observed records and one-shot estimation from retrieved information can therefore struggle to produce accurate population-level effect estimates. For Gemma-4-12B with key information retrieval, all three runs on task 3, 11, 15, 17 and 21 returned no final answer containing a parsable, finite ATE estimate. We therefore report RMSE over the remaining 16 tasks, taking the median of the available successful runs within each task (31 successful runs in total).

Plan-and-execute estimator choices. Table 4 reports the estimation frameworks selected by Plan-and-execute across its 63 runs per foundation model. AIPW is the most frequent choice for DeepSeek-V4-Flash, GPT-OSS-120B, Qwen3.6-27B, and Gemma-4-12B, while GPT-OSS-20B and

Table 4: Estimation frameworks chosen by Plan-and-execute, as a percentage of its 63 runs per foundation model (21 tasks × 3 runs).

Framework	DeepSeek-V4-Flash	GPT-OSS-120B	Qwen3.6-27B	Gemma-4-12B	GPT-OSS-20B	Qwen3-4B
AIPW (doubly robust)	96.8	74.6	100.0	60.3	36.5	0.0
IPW	0.0	23.8	0.0	9.5	61.9	58.7
Propensity-score matching	0.0	0.0	0.0	0.0	0.0	25.4
Outcome regression	3.2	1.6	0.0	28.6	0.0	0.0
Other / unreported	0.0	0.0	0.0	1.6	1.6	15.9

Qwen3-4B most often use IPW. Despite access to statistical tools and flexible analysis pipelines, Plan-and-execute underperforms TRACE-AGENT on every model. These results suggest that selecting a standard ATE estimator alone does not resolve the challenges of downstream model selection and training. TRACE-AGENT uses AIPW throughout and learns propensity and outcome prediction contexts through reflective prompt optimization, selective source reuse, and target-specific adaptation.

E.2 Ablation study on text embedding

We evaluate TRACE-NN and the statistical ML baselines using LLM-derived embeddings in place of bag-of-words covariates. We apply late chunking (Günther et al., 2024) with Qwen3-Embedding-8B to preserve information from long posts. Each post is encoded before mean pooling within sentence-based chunks, with causal attention retaining preceding context across chunk boundaries. We concatenate up to ten chunk vectors and zero-pad to 40,960 dimensions. Source and target populations, hyperparameters, and the evaluation protocol remain unchanged.

Table 5: Target-domain ATE estimation RMSE of statistical ML methods on TRACE-Bench with bag-of-words and late-chunk embedding covariates (lower is better). Results use per-task median ATE estimates from three runs, with task-level bootstrap standard errors in parentheses.

Method	Bag-of-words ($p = 67{,}398$)	Late-chunk embedding ($p = 40{,}960$)
ℓ_1-TCL (DragonNet)	0.0625 (0.0089)	0.0469 (0.0072)
ℓ_1-TCL (TARNet)	0.0342 (0.0041)	0.1723† (0.0209)
Finetune LASSO	0.0440 (0.0048)	0.0365 (0.0042)
Finetune Vanilla-NN	0.0863 (0.0152)	0.0483 (0.0059)
FIDDLE (source-only)	0.0457 (0.0056)	0.0438 (0.0049)
FIDDLE (target-only)	0.0368 (0.0042)	0.0369 (0.0052)
TRACE-NN (ours)	**0.0327** (0.0036)	**0.0348** (0.0047)

†18/21 tasks; no valid ATE was produced for the remaining three tasks (see text).

Table 5 shows that TRACE-NN maintains similar accuracy with bag-of-words and LLM-derived embeddings, achieving RMSEs of 0.0327 and 0.0348, respectively. It has the lowest reported RMSE in both settings, supporting its robustness to text representation. The embedding result for ℓ_1-TCL (TARNet) covers only 18 tasks because extreme propensity estimates leave no observations after trimming on three tasks.

E.3 Simulation study

Data generating process. Let covariates follow $\boldsymbol{x}^* = \boldsymbol{B}^* \boldsymbol{f}^* + \boldsymbol{u}^*$ for both $* \in \{P, Q\}$ with $r = 4$ factors. We draw factors $\boldsymbol{f}^*$ and idiosyncratic components $\boldsymbol{u}^*$ for both source and target from Unif$[-1, 1]$, and source loadings $\boldsymbol{B}^P = (b^P_{ij})_{p\times r}$ from $b^P_{ij} \overset{i.i.d.}{\sim}$ Unif$[-\sqrt{3}, \sqrt{3}]$. Target loadings are obtained via perturbation $b^Q_{ij} = b^P_{ij} + \epsilon^B_{ij}$ with $\epsilon^B_{ij} \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.01)$. Given covariates, treatments are drawn as $T^*_i \sim \text{Bernoulli}(\pi^*(\boldsymbol{f}^*_i, \boldsymbol{u}^*_i))$ and outcomes follow $Y^*_i =$

$\mu^*(\boldsymbol{f}_i^*, \boldsymbol{u}_i^*) + T_i^* \Delta^*(\boldsymbol{f}_i^*, \boldsymbol{u}_i^*) + \varepsilon_i^*$ with $\varepsilon_i^* \overset{i.i.d.}{\sim} \mathcal{N}(0, 1/4)$, where π^*, μ^*, and Δ^* denote the propensity, baseline outcome, and conditional treatment effect functions given the latent covariates, respectively. The outcome regressions given the observed numerical covariates are

$$\mu_t^*(\boldsymbol{x}) = \mathbb{E}_*[\mu^*(\boldsymbol{f}^*, \boldsymbol{u}^*) + t\Delta^*(\boldsymbol{f}^*, \boldsymbol{u}^*) \,|\, \boldsymbol{X}^* = \boldsymbol{x}], \qquad t \in \{0, 1\}.$$

We now specify the latent-covariate functions.

Source functions are additive over the factors and $|\mathcal{J}^P| = 5$ idiosyncratic components. The propensity is $\pi^P = \rho \circ \sigma \circ \eta^P$, where $\eta^P(\boldsymbol{f}^P, \boldsymbol{u}^P) = \boldsymbol{\alpha}^\top \boldsymbol{f}^P + \boldsymbol{\phi}^\top \boldsymbol{u}^P_{\mathcal{J}^P}$, $\sigma(x) = (1+e^{-x})^{-1}$ is the sigmoid, and $\rho(a) = 0.8a + 0.1$ projects to $[0.1, 0.9]$ to ensure overlap. The outcome and treatment effect functions are $\mu^P(\boldsymbol{f}^P, \boldsymbol{u}^P) = \boldsymbol{\beta}^\top \boldsymbol{m}(\boldsymbol{f}^P) + \boldsymbol{\psi}^\top \boldsymbol{u}^P_{\mathcal{J}^P}$ and $\Delta^P(\boldsymbol{f}^P, \boldsymbol{u}^P) = \tau_0 + \boldsymbol{\gamma}^\top \boldsymbol{f}^P + \boldsymbol{\omega}^\top \boldsymbol{u}^P_{\mathcal{J}^P}$, with element-wise nonlinearities $\boldsymbol{m}(\cdot)$ in μ^P.

Target functions are additive perturbations of their source counterparts, with one additional idiosyncratic dependence ($\mathcal{J}^Q = \mathcal{J}^P \cup \{6\}$): $\eta^Q(\boldsymbol{f}^Q, \boldsymbol{u}^Q) = \eta^P(\boldsymbol{f}^Q, \boldsymbol{u}^Q) + \tilde{\boldsymbol{\alpha}}^\top \boldsymbol{f}^Q$, $\mu^Q(\boldsymbol{f}^Q, \boldsymbol{u}^Q) = \mu^P(\boldsymbol{f}^Q, \boldsymbol{u}^Q) + \tilde{\boldsymbol{\beta}}^\top \boldsymbol{f}^Q + \tilde{\psi}_1 u_1^Q$, $\Delta^Q(\boldsymbol{f}^Q, \boldsymbol{u}^Q) = \Delta^P(\boldsymbol{f}^Q, \boldsymbol{u}^Q) + (\tilde{\tau}_0 - \tau_0) + \tilde{\boldsymbol{\gamma}}^\top \boldsymbol{f}^Q + \tilde{\omega}_6 u_6^Q$. We set $\tau_0 = 3$ and $\tilde{\tau}_0 = 4$, giving ground-truth target ATE $\tau^Q = \mathbb{E}_Q[\Delta^Q(\boldsymbol{f}^Q, \boldsymbol{u}^Q)] = 4$. We vary $p \in \{1000, 2500, 5000, 10000\}$ and $n^Q \in \{500, 1000, \ldots, 5000\}$ with $n^P = 10{,}000$ fixed, running 100 replications per configuration. We report RMSE with 95% confidence bands in Figure 5.

Simulation parameter values. The structural parameter values used in the data generating process above are as follows. The nonlinear factor map $\boldsymbol{m}$ in the source outcome model is drawn once from the candidate set $\{\mathrm{id}, \sin, \arctan, 2|\cdot| - 1\}$, and fixed as

$$\boldsymbol{m}(\boldsymbol{f}^P) = (\arctan(f_1^P),\ 2|f_2^P| - 1,\ 2|f_3^P| - 1,\ f_4^P)^\top.$$

The propensity coefficients are drawn i.i.d. from $\mathrm{Unif}[-0.3, 0.3]$, yielding

$$\boldsymbol{\alpha} = (-0.075,\ 0.270,\ 0.139,\ 0.059)^\top, \qquad \boldsymbol{\phi} = (-0.206,\ -0.206,\ -0.265,\ 0.220,\ 0.061)^\top.$$

The outcome and treatment effect coefficients are drawn i.i.d. from $\mathrm{Unif}[-1, 1]$, yielding

$$\boldsymbol{\beta} = (0.416,\ -0.959,\ 0.940,\ 0.665)^\top, \qquad \boldsymbol{\psi} = (-0.575,\ -0.636,\ -0.633,\ -0.392,\ 0.050)^\top,$$
$$\boldsymbol{\gamma} = (-0.136,\ -0.418,\ 0.224,\ -0.721)^\top, \quad \boldsymbol{\omega} = (-0.416,\ -0.267,\ -0.088,\ 0.570,\ -0.601)^\top.$$

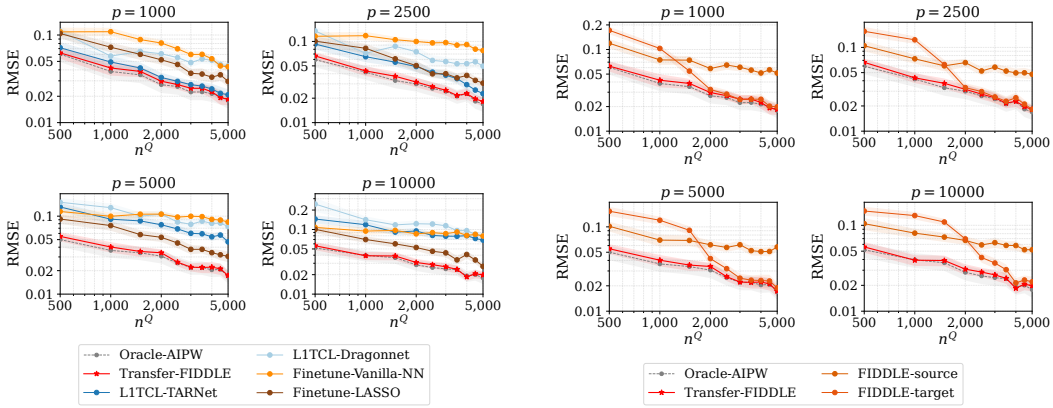
For the target-domain perturbation, we set

$$\tilde{\boldsymbol{\alpha}} = (0.1,\ -0.1,\ 0,\ 0)^\top, \qquad \tilde{\boldsymbol{\beta}} = (0.5,\ -0.5,\ 0,\ 0)^\top, \qquad \tilde{\boldsymbol{\gamma}} = (0.3,\ 0,\ 0,\ 0)^\top,$$

with $\tilde{\psi}_1 = 0.2$ and $\tilde{\omega}_6 = -0.1$; the intercepts $\tau_0 = 3$ and $\tilde{\tau}_0 = 4$ are given above. All structural coefficients are drawn once and held fixed across replications; only the sampled covariates, treatments, and outcome noise vary.

Results. As reported in Figure 5a, TRACE-NN consistently achieves near-oracle performance across all (p, n^Q) configurations, and the performance gap between TRACE-NN and other transfer baselines widens as the ambient dimension p grows. We attribute this advantage to two design choices. First, the diversified projection matrix extracts factor surrogates from the sample covariance without learnable parameters, yielding stable low-dimensional representations even at large p. Second, the two-stage pipeline passes the source model's predictions as an additional input to the target model, serving as implicit dimension reduction: when distribution shift is moderate, the effective number of idiosyncratic components in the finetuning stage is smaller than in the pretraining stage. This avoids imposing explicit constraints on target model weights, so that TRACE-NN's tolerance to distribution and posterior shift is less dependent on hyperparameter tuning.

The ablation in Figure 5b supports the pretraining-finetuning architectural design of TRACE-NN. Without target finetuning, FIDDLE (source-only) cannot account for distribution and posterior shift between domains, leading to persistent bias that does not diminish with increasing n^Q. Without source pretraining, FIDDLE (target-only) must learn all functional structures from the small target dataset alone, resulting in high variance in the large-p, small-n^Q regime. TRACE-NN combines both stages to mitigate each failure mode.



(a) TRACE-NN vs. transfer learning baselines. (b) Ablation study with single-domain baselines

Figure 5: ATE estimation RMSE (log–log scale) across $p \in \{1000, 2500, 5000, 10000\}$ and $n^Q \in [500, 5000]$. **(a)** TRACE-NN (red) closely tracks Oracle-AIPW (gray, dashed) and uniformly dominates all transfer baselines; the gap widens with p. **(b)** Ablation isolating individual components: FIDDLE (source-only, no target finetuning) and FIDDLE (target-only, no source pretraining). TRACE-NN achieves the best data efficiency, with the largest gains at small n^Q and large p.

Algorithm 1 Prompt optimization and target context finetuning

Loss definitions: For nuisance predictions $q_i = (\pi_i, \mu_{0,i}, \mu_{1,i})$,
$\ell(p, y) = -y \log p - (1 - y) \log(1 - p), \qquad \ell_{\pi,i}(q_i) = \ell(\pi_i, T_i), \quad \ell_{\mu,i}(q_i) = \ell(\mu_{T_i,i}, Y_i).$
$\mathcal{L}_a(q; D) = |D|^{-1} \sum_{i \in D} \ell_{a,i}(q_i).$

Helpers: $\textsc{Eval}(a, c, D, \widehat{s})$ returns per-example losses and their mean using context c; $\widehat{s} = \varnothing$ for source learning.

$w_{\mathcal{P}}(c) = \sum_{i \in D_V} \mathbf{1}\left\{ \ell_i(c) = \min_{u \in \mathcal{P}} \ell_i(u) \right\}.$ ▷ Validation wins; ties count

procedure $\textsc{Evolve}(a, c_0, D_L, D_V, \widehat{s})$
  $(\boldsymbol{\ell}(c_0), L(c_0)) \leftarrow \textsc{Eval}(a, c_0, D_V, \widehat{s})$
  $\mathcal{P} \leftarrow \{c_0\}$; $r_\star \leftarrow 0$ ▷ $r_\star$: Last round improving the best mean loss
  for $r = 1, \ldots, 4$ **do**
    $b \leftarrow \min_{u \in \mathcal{P}} L(u)$
    $c \sim p(c) = \dfrac{1 + w_{\mathcal{P}}(c)}{\sum_{u \in \mathcal{P}} (1 + w_{\mathcal{P}}(u))}$
    $B \sim \mathrm{UniformSubset}(D_L, \min(64, |D_L|))$ ▷ Without replacement
    $(\boldsymbol{\ell}_B, \cdot) \leftarrow \textsc{Eval}(a, c, B, \widehat{s})$
    $E \leftarrow \mathrm{TopLossExamples}_8(B, \boldsymbol{\ell}_B)$
    $c' \leftarrow \textsc{Reflect}(a, c, E, D_L, \widehat{s})$ ▷ Tool access to all D_L
    if $\mathrm{Valid}(c')$ **and** $c' \neq c$ **then**
      $(\boldsymbol{\ell}(c'), L(c')) \leftarrow \textsc{Eval}(a, c', D_V, \widehat{s})$
      if $L(c') < b$ **then** $r_\star \leftarrow r$
      if $L(c') < b$ **or** $\exists i \in D_V : \ell_i(c') < \min_{u \in \mathcal{P}} \ell_i(u)$ **then**
        $\mathcal{P} \leftarrow \mathcal{P} \cup \{c'\}$
        $\mathcal{P} \leftarrow \{u \in \mathcal{P} : w_{\mathcal{P}}(u) > 0\} \cup \left\{ \arg\min_{u \in \mathcal{P}} L(u) \right\}$
      end if
    end if
    if $r - r_\star \geq 2$ **then break**
  end for
  return $\arg\min_{c \in \mathcal{P}} L(c)$ ▷ Mean-loss ties favor the earlier candidate
end procedure

procedure $\textsc{TargetFinetune}(D_Q, \widehat{s}, K, \delta)$ ▷ K: number of folds; δ: gate margin
  $(I_1, \ldots, I_K) \leftarrow \mathrm{RandomPartition}(D_Q, K)$
  for $k = 1, \ldots, K$ **do**
    $D_{-k} \leftarrow D_Q \setminus I_k$; $N \leftarrow |D_{-k}|$
    $\bar{\pi} \leftarrow \dfrac{1 + \sum_{i \in D_{-k}} T_i}{N + 2}$
    $\bar{\mu}_t \leftarrow \dfrac{1 + \sum_{i \in D_{-k}} \mathbf{1}\{T_i = t\} Y_i}{2 + \sum_{i \in D_{-k}} \mathbf{1}\{T_i = t\}}, \quad t \in \{0, 1\}$
    $\bar{q} \leftarrow (\bar{\pi}, \bar{\mu}_0, \bar{\mu}_1)$ ▷ Laplace-smoothed constants
    for $a \in \{\pi, \mu\}$ **do**
      $L_{\mathrm{src}} \leftarrow \mathcal{L}_a(\widehat{s}; D_{-k})$; $L_{\mathrm{base}} \leftarrow \mathcal{L}_a(\bar{q}; D_{-k})$
      if $L_{\mathrm{src}} \leq L_{\mathrm{base}} - \delta$ **then**
        $\widehat{q}^a_{I_k} \leftarrow \widehat{s}^a_{I_k}$
      else
        $n_V \leftarrow \min(48, \lfloor 0.2N \rfloor)$
        $(D_L, D_V) \leftarrow \mathrm{StratifiedSplit}(D_{-k}, a, n_V)$
        $c_0 \leftarrow \mathrm{TargetSeed}(a) \oplus \mathrm{GateDiagnostics}(D_{-k}, \widehat{s}, \delta)$
        $c^\star \leftarrow \textsc{Evolve}(a, c_0, D_L, D_V, \widehat{s})$
        $\widehat{q}^a_{I_k} \leftarrow \textsc{Predict}(a, c^\star, I_k, \widehat{s})$
      end if
    end for
  end for
  return $\widehat{q} = (\widehat{\pi}, \widehat{\mu}_0, \widehat{\mu}_1)$
end procedure

Table 6: Target-domain ATE estimation RMSE on the News Device Preference benchmark across 50 datasets ($n = 5{,}000$, $p = 3{,}477$; lower is better). Standard errors are in parentheses.

ℓ_1-TCL (DragonNet)	ℓ_1-TCL (TARNet)	Finetune LASSO	Finetune Vanilla-NN	FIDDLE (source-only)	FIDDLE (target-only)	**TRACE-NN (ours)**
0.6075 (0.0751)	0.3518 (0.0424)	0.2381 (0.0205)	0.3107 (0.0348)	0.8785 (0.1458)	0.3030 (0.0315)	**0.2278** (0.0218)

E.4 News device preference experiment

We use the semi-synthetic *News* benchmark (Johansson et al., 2016) to validate TRACE-NN on another real-world numerical transfer ATE estimation setting. The benchmark provides 50 independent datasets, each containing $5{,}000$ news items from the NY Times corpus, where each news item is a $3{,}477$-dimensional bag-of-words count vector. The treatment $T \in \{0, 1\}$ indicates the viewing device (desktop or mobile), and the outcome $Y \in \mathbb{R}$ measures reader experience. Both treatment assignment and outcome generation depend on a 50-dimensional LDA topic distribution $z(x)$: treatment follows a softmax over the similarity between $z(x)$ and device-specific topic centroids, and the outcome is a linear function of these similarities plus Gaussian noise. This couples confounding to article content. Each dataset is split by total word count: the $4{,}750$ items with the lowest counts form the source domain, and the 250 items with the highest counts (the upper 5% of the word-count distribution) form the target domain.

Results. Table 6 summarizes performance on the *News* benchmark, averaged over the 50 datasets. TRACE-NN achieves the lowest RMSE, outperforming all transfer baselines by a clear margin. FIDDLE (target-only) and FIDDLE (source-only) each perform substantially worse, confirming that source pretraining combined with target finetuning is essential for the transfer gain.