Calibrating LLM Confidence by Probing Perturbed Representation Stability

Anonymous ACL submission

Abstract

Miscalibration in Large Language Models (LLMs) undermines their reliability, highlighting the need for accurate confidence estimation. We introduce CCPS (Calibrating LLM Confidence by Probing Perturbed Representation Stability), a novel method analyzing internal representational stability in LLMs. CCPS applies targeted adversarial perturbations to final hidden states, extracts features reflecting the model's response to these perturbations, and uses a lightweight classifier to predict answer correctness. CCPS was evaluated on LLMs from 8B to 32B parameters (covering Llama, Qwen, and Mistral architectures) using MMLU and MMLU-Pro benchmarks in both multiple-choice and open-ended formats. Our results show that CCPS significantly outperforms current approaches. Across four LLMs and three MMLU variants, CCPS reduces Expected Calibration Error by approximately 55% and Brier score by 21%, while increasing accuracy by 5 percentage points, Area Under the Precision-Recall Curve by 4 percentage points, and Area Under the Receiver Operating Characteristic Curve by 6 percentage points, all relative to the strongest prior method. CCPS delivers an efficient, broadly applicable, and more accurate solution for estimating LLM confidence, thereby improving their trustworthiness.

1 Introduction

Despite their impressive performance, large language models (LLMs) consistently struggle with confidence calibration (Guo et al., 2017; Geng et al., 2024). Their confidence—the model's internally estimated probability that a given response is correct—frequently misaligns with actual outcomes: LLMs often assign high confidence to wrong answers and low confidence to right ones. This unreliability is particularly acute in high-stakes domains like medicine, finance, and law. For example, in a critical medical task like symptom extraction for cancer toxicity assessment, even if an LLM often produces correct information, it might do so with inappropriately low confidence, or conversely, express high confidence for incorrect outputs. If such confidence scores are not dependable guides to actual correctness, human experts may be forced to meticulously review every LLM-generated instance, significantly diminishing the practical benefits of automation. Accurate confidence estimation for each specific response is therefore essential, as it provides a vital mechanism for managing risk, enabling users to prioritize human oversight, selectively rely on LLM outputs, and ultimately foster more responsible and effective LLM integration.

Existing approaches to LLM confidence estimation include direct self-evaluation (Kadavath et al., 2022), post-hoc adjustments (Jiang et al., 2021), internal state probing with lightweight classifiers (Azaria and Mitchell, 2023; Liu et al., 2024), and model fine-tuning (Kapoor et al., 2024b). These methods often struggle to consistently deliver on multiple desirable properties simultaneously, namely achieving strong calibration (e.g., low Expected Calibration Error (ECE)) and high discriminative power (e.g., high Area Under the Precision-Recall Curve (AUCPR) or Area Under the Receiver Operating Characteristic Curve (AUROC)) while maintaining computational efficiency and robust generalizability across the diverse set of LLM architectures and families. Many methods excel in some of these desirable properties but make trade-offs in others; for instance, fine-tuning approaches like Calibration-Tuning (CT) (Kapoor et al., 2024b) often achieve strong calibration in ECE but may not consistently lead in discriminative metrics like AUROC, while lightweight methods such as LitCab (Liu et al., 2024) can demonstrate strong AUROC but sometimes show variable ECE performance across different LLM families. This leaves a need for more holistically effective solutions.

In this work, we introduce **CCPS** (**C**alibrating LLM **C**onfidence by Probing **P**erturbed Representation **S**tability), a novel method that addresses these challenges by assessing LLM confidence through the stability of its internal representations. CCPS operates on frozen base LLMs, applying targeted adversarial perturbations to the final hidden states that generate an answer's tokens. From the LLM's response to these perturbations, we extract a rich feature set and train a lightweight classifier to predict answer correctness. This model-agnostic probing offers an efficient confidence proxy without modifying the base LLM.

Comprehensive evaluations demonstrate CCPS's significant advantages over existing confidence estimation approaches. Tested across four modern LLMs (8B to 32B parameters, spanning three architectural families) on MMLU and MMLU-Pro benchmarks in both multiple-choice and open-ended formats, CCPS consistently achieves superior performance across key calibration (e.g., ECE, Brier score) and discrimination metrics (e.g., ACC, AUCPR, AUROC). Our findings reveal that by quantifying LLM representational stability through targeted internal perturbations, CCPS achieves substantial improvements over other state-of-the-art confidence estimation methods; for instance, CCPS reduces average ECE by approximately 55% (up to 88%) and Brier score by 21% (up to 45%), while also increasing average Accuracy (ACC) by 5 percentage points (pp) (up to +14 pp), AUCPR by 4 pp (up to +13 pp), and AUROC by 6 pp (up to +17 pp), relative to the best performing baseline. The key contributions of this work include:

- A novel, model-agnostic, parameter-efficient, and scalable framework (CCPS) offering a fresh perspective on LLM confidence estimation by quantifying it through the stability of internal representations under targeted perturbations.
- Demonstration of CCPS's substantial improvements in both key calibration (ECE, Brier score) and discrimination (ACC, AUCPR, AUROC) metrics.
- Evidence of CCPS's generalizability across diverse LLM architectures (Llama, Qwen, Mistral; 8B to 32B).
- Extensive benchmarking of confidence estimation methods on MMLU and MMLU-Pro in both multiple-choice and open-ended formats.

These contributions establish CCPS as an effective method for improving LLM confidence estimation, helping to make LLM applications more trustworthy, especially in critical domains where reliability is crucial.

2 Related Work

Calibration in LLMs A model is considered well-calibrated when its expressed confidence in a prediction aligns with the empirical likelihood of that prediction being correct. In the context of LLMs, calibration efforts broadly diverge into two streams. The first targets calibration of next-token predictions and responses to reduce hallucinations. This direction is exemplified by the work of Zhou et al. (2025), which focuses on hallucination mitigation through comprehensive model calibration. The second stream, more aligned with the present work, focuses on developing and calibrating explicit confidence estimation mechanisms that assess the correctness of statements generated by LLMs.

Confidence Estimation in LLMs Several approaches have been proposed for estimating an LLM's confidence in its assertions. One vein of research explores probing the internal states of LLMs. For instance, Azaria and Mitchell (2023) train an auxiliary linear classifier on hidden layer activations from an LLM to predict the truthfulness of statements. While this can reveal internal knowledge, its efficacy depends on identifying the optimal representational layer and may vary across evaluation metrics. Another approach involves eliciting the model's inherent self-assessment. Kadavath et al. (2022) introduced concepts like P(True), the probability an LLM assigns to its generated answer being correct (often derived from probabilities of "True" or "False" tokens when prompted to evaluate its own previous answer), and P(IK), the probability the model assigns to its own ability to answer a given question correctly, estimated before attempting to generate the answer. These methods assess the model's intrinsic confidence without external classifiers but rely on the LLM's inherent, and often uncalibrated, self-evaluation capabilities.

Improving Confidence Calibration in LLMs Other research adapts the LLM or its outputs to produce more reliable confidence scores. Logit Temperature Scaling (LTS) (Jiang et al., 2021) is a post-hoc method that adjusts output logits using a learned temperature parameter; however, its performance can degrade under distributional

shifts between calibration and test data (Kapoor et al., 2024b). More intensive methods involve fine-tuning. CT (Kapoor et al., 2024a,b) builds on the P(True) concept, prompting the LLM to assess its own answers and then fine-tuning it on this self-evaluation using methods like LoRA. This can achieve strong ECE but may face challenges in efficient class discrimination (e.g., AUROC) and can be computationally demanding. In contrast, LitCab (Liu et al., 2024) offers a lightweight approach by training a single linear layer to predict a bias term added to the LLM's output logits. While LitCab shows strong discrimination, our experiments reveal variable ECE across LLM families. These diverse strategies highlight an ongoing trade-off in achieving robust calibration, discriminative power, computational efficiency, and generalization in LLM confidence estimation.

3 Method

Our approach to LLM confidence estimation is centered on evaluating the internal stability of the model's representations when its generated answer is produced. We hypothesize that an LLM's confidence correlates with the robustness of its internal states; specifically, the final hidden states that lead to the tokens of a high-confidence answer should exhibit greater stability when subjected to targeted perturbations. This internal probing of representational stability offers an efficient alternative to methods relying on multiple generation passes for consistency checking. Notably, output consistency has been identified as a strong indicator of LLM reliability (Zhou et al., 2025), but external checks involve significant computational overhead, which our internal analysis aims to mitigate while leveraging a similar underlying principle of stability.

The methodology involves three primary stages, applied while the base LLM (whose confidence is being estimated) remains frozen: (1) token-level adversarial perturbation of the LLM's final hidden states along a defined trajectory, (2) extraction of features that quantify the impact of these perturbations, and (3) a classification architecture that maps these features to a confidence score, representing the answer's probability of correctness. These three stages are illustrated in Figure 1.

3.1 Probing Internal Stability

For a given input prompt P (which includes few-shot exemplars and the target question) and an answer $A = (t_1, t_2, \ldots, t_L)$ generated by the base LLM, where t_i is the i-th token, we analyze each token individually:

Original State Identification For each token t_i in A, we first identify the original final hidden state $H_0^{(i)} \in \mathbb{R}^{d_h}$ from the LLM's last transformer layer that immediately led to the generation of t_i. This is obtained by feeding P and any preceding generated tokens $t_{<i}$ into the LLM. The corresponding original logits are $Z_0^{(i)} = \text{LM\_Head}(H_0^{(i)})$.

Adversarial Perturbation Trajectory Direction To define a systematic perturbation trajectory that challenges the LLM's generation of the *observed token* t_i, we utilize the gradient of the loss associated with t_i with respect to its generating hidden state $H_0^{(i)}$. Let $P(t_i|H_0^{(i)})$ be the probability of the token t_i given $H_0^{(i)}$. We define the loss as the negative log-likelihood: $\mathcal{L}^{(i)} = -\log P(t_i|H_0^{(i)})$. The Jacobian vector $J^{(i)} = \nabla_{H_0^{(i)}} \mathcal{L}^{(i)}$ then indicates the direction in the hidden state space where this loss $\mathcal{L}^{(i)}$ increases most rapidly; equivalently, this is the direction where the probability of token t_i decreases most steeply. We normalize this vector to obtain the unit direction $d^{(i)} = J^{(i)}/||J^{(i)}||_2$. If $J^{(i)}$ is a zero vector, $d^{(i)}$ is also set to zero. Perturbing along this direction $d^{(i)}$ is an adversarial act[1] aimed at making the original token t_i less likely. This contrasts with standard LLM training where one steps in the negative gradient direction (e.g., $-\nabla\mathcal{L}$) to *reduce* loss for a target token. Here, by moving along the positive gradient of $\mathcal{L}^{(i)}$, we are adversarially probing the stability of the LLM's initial choice t_i by actively trying to dislodge it.

Iterative Adversarial Perturbation We then explore the stability of $H_0^{(i)}$ by applying S discrete adversarial perturbations along the direction $d^{(i)}$. The maximum extent of this exploration is defined by a radius ϵ_{max} (in our experiments, $\epsilon_{\text{max}} = 20.0$ and $S = 5$). The s-th perturbation magnitude is $\epsilon_s = s \cdot (\epsilon_{\text{max}}/S)$, for $s \in \{1, \ldots, S\}$. The s-th perturbed hidden state is:

$$H_s^{(i)} = H_0^{(i)} + \epsilon_s \cdot d^{(i)}$$

For each $H_s^{(i)}$, we compute the corresponding per-

[1] Our use of "adversarial" here refers to targeted, gradient-informed perturbations designed to systematically probe representational stability by challenging the generation of token t_i. This is distinct from adversarial attacks aimed at finding minimal input perturbations to cause misclassification or from adversarial training regimens.

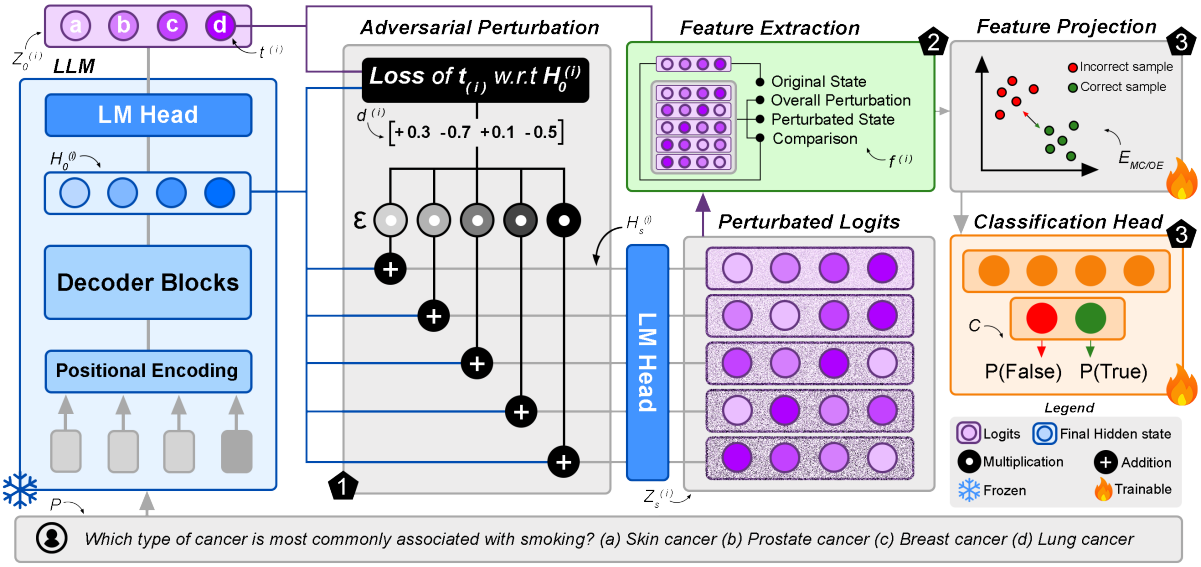

Figure 1: Overview of the CCPS method, illustrating its three primary stages. **(1)** For each token t_i (e.g., 'd' in the example) from a frozen LLM's response to an input prompt P, its original final hidden state $H_0^{(i)}$ is systematically perturbed (details in §3.1). This involves moving $H_0^{(i)}$ along a derived adversarial direction $d^{(i)}$ with varying magnitudes ϵ_s (visually represented by lighter to darker shades for increasing ϵ_s) to yield a trajectory of perturbed hidden states $H_s^{(i)}$ and their corresponding logits $Z_s^{(i)}$ via the LM Head. **(2)** A D_f-dimensional feature vector $f^{(i)}$ is then engineered (§3.2), encompassing original state characteristics, perturbation stability indicators, and trajectory divergence statistics derived from the original and perturbed representational data. **(3)** This feature vector $f^{(i)}$ is subsequently processed by a trainable feature projection network ($E_{\text{MC/OE}}$) and a classification head (C) (§3.3) to output the final confidence score, $P(\text{True})$, indicating the likelihood of the LLM's answer being correct.

turbed logits $Z_s^{(i)} = \text{LM\_Head}(H_s^{(i)})$. This creates a trajectory of hidden states and their resulting output distributions under these adversarial nudges.

3.2 Quantifying Perturbation Impact

From the original hidden state $H_0^{(i)}$ and its corresponding logits $Z_0^{(i)}$, along with the trajectory of S perturbed hidden states $\{H_s^{(i)}\}_{s=1}^{S}$ and their respective logits $\{Z_s^{(i)}\}_{s=1}^{S}$, we extract a D_f-dimensional feature vector $f^{(i)}$ for each token t_i. These features are designed to capture the LLM's initial output characteristics for t_i and how these characteristics evolve under systematic adversarial perturbation. Detailed definitions of all features are provided in Appendix B. The primary categories are:

Original State Features This set quantifies the LLM's baseline predictive characteristics for token t_i prior to any perturbation, including measures of output probabilities, logits, distribution entropy, and prediction margins.

Overall Perturbation Features This category comprises scalar metrics reflecting key aspects of the perturbation process itself or its integrated effects, such as the L2 norm of the Jacobian vector $J^{(i)}$, the perturbation magnitude required to change the LLM's top predicted token from t_i (epsilon-to-flip), and the Perturbation Energy Integral (PEI) value which summarizes the impact of perturbations on the log-probability of t_i.

Perturbed State Features These features consist of statistical summaries (e.g., mean, standard deviation, min, max, across the S perturbation steps) of the LLM's output characteristics (such as token probabilities and distribution entropy) evaluated *after* its hidden states have been perturbed.

Comparison Features This group includes statistical summaries of metrics that quantify the differences or relationships (e.g., distributional divergences like Kullback–Leibler and Jensen–Shannon, cosine similarities) between the LLM's original state (hidden states, logits, probability distributions) and its perturbed states across the trajectory.

3.3 Confidence Classification Architecture

The per-token feature vectors serve as input to a neural network designed to predict the correctness of the entire answer A. This architecture comprises a feature projection network and a classification

head.

Feature Projection Network The network structure adapts to the answer format. For Multiple-Choice (MC) answers, which are typically single-token responses, the feature vector $f^{(1)}$ is processed by a Multi-Layer Perceptron (MLP), denoted as E_{MC}, to yield an embedding $e = E_{\text{MC}}(f^{(1)})$. In contrast, for Open-Ended (OE) answers consisting of L tokens, the sequence of feature vectors $(f^{(1)}, \ldots, f^{(L)})$ is passed through an encoder E_{OE} composed of 1D convolutional layers and adaptive pooling, resulting in a sequence embedding $e = E_{\text{OE}}(f^{(1)}, \ldots, f^{(L)})$.

Both E_{MC} and E_{OE} are pre-trained using a Max-Margin contrastive loss. This choice of loss is aimed at learning discriminative embeddings, a strategy also found effective in other confidence estimation works such as Liu et al. (2024), though our feature generation process and overall architecture are distinct. The objective of this pre-training is to map features from correctly answered questions to regions in the embedding space that are separable from those associated with incorrect answers, supervised by the ground truth correctness of A.

Classification Head The embedding e is then passed to an MLP classification head, C. This head outputs a 2-dimensional logit vector, $Z_{\text{conf}} = C(e)$. This architectural choice for binary correctness prediction (incorrect vs. correct) is similar to that used by Kapoor et al. (2024b). The final confidence score, $P(\text{correct}|A)$, is obtained via a softmax function applied to Z_{conf}.

Training Procedure Following the contrastive pre-training of the projection network, the projection network (E_{MC} or E_{OE}) and the classification head C are jointly fine-tuned. This stage employs a standard cross-entropy loss, again supervised by the ground truth correctness of answer A.

4 Experimental Setup

We empirically evaluate the effectiveness of our proposed confidence estimation method.

Languge Models Our experiments utilize a range of contemporary decoder-only LLMs, with parameter sizes from 8B to 32B, encompassing three distinct model families to assess broader applicability. Specifically, we include: `Meta-Llama-3.1-8B-Instruct` (Grattafiori et al., 2024) (governed by the Llama 3.1 Community License Agreement), `Qwen2.5-14B-Instruct` (Qwen et al., 2025) (Apache License 2.0), `Mistral-Small-24B-Instruct-2501` (Apache License 2.0), and `Qwen2.5-32B-Instruct` (Qwen et al., 2025) (Apache License 2.0). We use the `unsloth` (Daniel Han and team, 2023) versions of these models to leverage their efficient, non-quantized model handling capabilities. All experiments, including hidden state extraction and model training, were conducted using `bfloat16` precision to ensure manageability across the diverse models and computational tasks. Our implementations rely on the HuggingFace Transformers (Wolf et al., 2020) and PyTorch (Paszke et al., 2019) libraries.

Datasets For training and validating our confidence estimation models, we utilize the `CT-CHOICE` and `CT-OE` datasets for multiple-choice and open-ended question-answering formats, respectively. These datasets, generated following the exact methodology detailed by Kapoor et al. (2024b) (Apache License 2.0), comprise a large collection of commonly used question-answering datasets from the literature. To assess generalization and performance, we evaluate on tasks from the Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2021) (MIT License). We created multiple-choice and open-ended versions of these tasks, namely `MMLU-CHOICE` and `MMLU-OE`, using the same data processing approach as Kapoor et al. (2024b) to ensure consistency. Additionally, we employ `MMLU-PRO-CHOICE` (Apache License 2.0), a multiple-choice version of the MMLU-Pro dataset (Wang et al., 2024), for further rigorous testing. All dataset instances across training, validation, and testing incorporate 5-shot exemplars within the input prompt P to contextualize the LLMs. Additional details on the dataset characteristics, response generation process, and labeling procedures are provided in Appendix C.

Training Details To ensure fair comparisons, training configurations were kept consistent across all methods, including baselines. The main classification/fine-tuning stage for all models involved a total of 10,000 training steps. For our proposed method, the contrastive feature projection network (E_{MC} or E_{OE}) was pre-trained for 5,000 steps. Subsequently, the confidence classification model was trained for an additional 5,000 steps. Key hyperparameters for the AdamW optimizer (Loshchilov and Hutter, 2019), such as a learning rate of 1×10^{-4}, were aligned with those reported by Kapoor et al. (2024b). Training was

conducted with a batch size of 32. A weight decay of 0.1 was uniformly applied across all training stages and methods.

Baselines We compare our method (CCPS) against a comprehensive set of established confidence estimation techniques. These include P(True) and P(IK) (Kadavath et al., 2022), Logit Temperature Scaling (LTS) (Jiang et al., 2021), Instruction Tuning (IT) (Wei et al., 2022) on the uncertainty query, SAPLMA (Azaria and Mitchell, 2023) (with variants SAPLMA-F, SAPLMA-M, and SAPLMA-UM corresponding to different layer inputs), Calibration Tuning (CT) (Kapoor et al., 2024b), and LitCab (Liu et al., 2024). Detailed descriptions of these baseline methods are provided in Appendix D, and the architectural specifics of our CCPS model, including details of the hyperparameter search process that determined these architectures, are detailed in Appendix E. Information regarding the computational setup and resources utilized for all methods is available in Appendix F. Furthermore, a comparative analysis of the additional trainable parameters introduced by each method is presented in Appendix G, underscoring the parameter efficiency of our CCPS approach.

Evaluation Metrics We evaluate our confidence estimation method using established metrics. For assessing calibration, we employ the ECE and the Brier Score. We also measure our confidence scores' discriminative performance using ACC, AUCPR, and AUROC. Detailed definitions and formulations for these metrics are provided in Appendix H.

Scientific Artifacts A detailed discussion regarding the scientific artifacts utilized and developed in this study, including our adherence to their intended use and the intended applications of our created artifacts, can be found in Appendix A.

5 Results

The performance of CCPS compared to baseline methods across different LLMs and MMLU benchmark variants is presented in Table 1. Our method, CCPS, consistently demonstrates notable improvements in both calibration and discriminative power.

On the standard multiple-choice benchmark, `MMLU-CHOICE`, CCPS consistently achieves superior performance across all four base LLMs. For instance, ECE scores for CCPS are typically in the range of 5.8-6.5%, representing substantial reductions compared to both LitCab and CT, which often exhibit much higher ECEs (e.g., LitCab's ECE of 45.6% and CT's 45.2% on `Qwen2.5-14B` and `Qwen2.5-32B` respectively, against CCPS's 6.3% on both). CCPS shows similar gains in Brier

Method	ECE ↓	BRIER ↓	ACC ↑	AUCPR ↑	AUROC ↑
MMLU-CHOICE					
Meta-Llama-3.1-8B-Instruct					
LitCab	10.9	18.1	73.2	84.0	**77.1**
CT	10.7	21.1	67.8	74.2	62.8
CCPS	**6.5**	**17.1**	**73.4**	**84.1**	**77.1**
Qwen2.5-14B-Instruct					
LitCab	45.6	20.0	78.3	83.7	65.3
CT	12.1	17.0	78.6	84.7	64.8
CCPS	**6.3**	**13.1**	**80.2**	**92.1**	**81.6**
Mistral-Small-24B-Instruct-2501					
LitCab	13.5	15.1	79.5	91.5	78.2
CT	8.2	15.5	79.6	83.3	56.5
CCPS	**5.8**	**11.5**	**83.0**	**93.1**	**83.3**
Qwen2.5-32B-Instruct					
LitCab	43.2	15.9	82.6	87.9	67.2
CT	45.2	46.9	37.2	84.3	51.6
CCPS	**6.3**	**10.8**	**84.1**	**94.1**	**82.8**
MMLU-PRO-CHOICE					
Meta-Llama-3.1-8B-Instruct					
LitCab	16.6	24.7	66.1	51.7	63.6
CT	21.5	29.8	50.4	43.7	57.3
CCPS	**4.5**	**20.0**	**70.4**	**55.2**	**67.9**
Qwen2.5-14B-Instruct					
LitCab	49.7	38.3	55.3	66.2	68.0
CT	20.4	28.7	55.6	59.4	56.6
CCPS	**4.2**	**20.1**	**69.2**	**75.8**	**74.0**
Mistral-Small-24B-Instruct-2501					
LitCab	32.2	34.6	57.0	66.2	60.1
CT	17.8	27.4	58.2	60.1	54.3
CCPS	**4.5**	**18.6**	**71.3**	**79.5**	**77.2**
Qwen2.5-32B-Instruct					
LitCab	48.4	33.7	60.8	72.7	70.3
CT	38.0	41.6	44.8	60.5	49.9
CCPS	**4.6**	**18.5**	**71.8**	**82.4**	**77.8**
MMLU-OE					
Meta-Llama-3.1-8B-Instruct					
LitCab	8.8	22.5	65.3	46.2	66.0
CT	8.8	21.1	65.3	48.9	**70.9**
CCPS	**8.0**	**20.2**	**69.5**	**49.4**	69.3
Qwen2.5-14B-Instruct					
LitCab	34.4	37.0	49.4	56.8	62.5
CT	9.4	22.6	63.4	**61.7**	**69.3**
CCPS	**6.7**	**22.5**	**63.6**	59.0	66.6
Mistral-Small-24B-Instruct-2501					
LitCab	11.2	24.6	60.2	60.5	66.4
CT	10.8	22.8	62.2	60.7	68.2
CCPS	**6.8**	**20.8**	**67.6**	**64.7**	**71.4**
Qwen2.5-32B-Instruct					
LitCab	28.4	33.2	52.7	60.2	62.3
CT	22.9	31.1	57.1	52.9	56.3
CCPS	**8.7**	**23.3**	**62.6**	**62.0**	**66.4**

Table 1: Average performance on MMLU variants across tasks per LLM. Arrows indicate whether lower (↓) or higher (↑) values are better. All values are percentages. Best values per method-block are bolded.

score and discriminative metrics like AUCPR and AUROC, often matching or outperforming baselines.

When evaluated on the more challenging MMLU-PRO-CHOICE dataset, CCPS further extends its performance advantages, particularly in calibration. CCPS consistently achieves ECE values around 4.5% across all tested LLMs, a significant improvement over LitCab (ECEs ranging from 16.6% to 49.7%) and CT (ECEs from 17.8% to 38.0%). This strong calibration is paired with top scores in Brier, ACC, AUCPR, and AUROC, showing CCPS's robustness on more difficult questions. For example, with Mistral-24B, CCPS records an ECE of 4.5% and an AUROC of 77.2%, compared to LitCab's 32.2% ECE and 60.1% AUROC, and CT's 17.8% ECE and 54.3% AUROC.

In the open-ended generation setting (MMLU-OE), CCPS generally maintains strong calibration, consistently achieving the best ECE and Brier scores, especially with larger models like Mistral-24B and Qwen2.5-32B where it leads across all metrics. For smaller models on MMLU-OE, while CCPS leads in calibration, CT demonstrates competitive discriminative performance in AUCPR and AUROC (e.g., for Llama-3.1-8B, CT's AUROC is 70.9% vs. CCPS's 69.3%; for Qwen2.5-14B, CT leads in AUCPR and AUROC). However, CCPS's calibration advantage remains evident, for example, achieving an ECE of 6.7% with Qwen2.5-14B compared to CT's 9.4%.

In summary, CCPS consistently delivers substantial improvements in confidence estimation, excelling in both calibration and the ability to discriminate between correct and incorrect responses across diverse LLMs and task formats, particularly on challenging multiple-choice benchmarks. The findings in Table 1 are further detailed in Appendix I, which includes comprehensive results for all baselines (mean and standard deviation scores, comparative bar charts, per-task breakdowns, and feature importance analyses).

6 Discussion

CCPS Excels in Both Calibration and Discrimination. A significant finding is the ability of CCPS to simultaneously achieve strong calibration (low ECE and Brier scores) and high discriminative power (high AUCPR and AUROC), as evidenced in Table 1. This contrasts with observations for some baselines; for instance, while LitCab often demonstrates good discrimination, its ECE can be variable, particularly with certain LLM families (e.g., Qwen models). Conversely, Calibration Tuning (CT) generally achieves good ECE but can lag in discriminative metrics compared to CCPS. Our method's dual strength suggests that the features extracted from internal perturbation trajectories effectively capture signals relevant to both the reliability and the correctness of an LLM's answer.

The CCPS Framework Provides an Efficient and Scalable Approach to Confidence Estimation. CCPS is designed to be lightweight. Once features are extracted, the confidence estimation model itself consists of relatively small MLPs or CNNs (as detailed in Appendix E), making its training and inference efficient. This contrasts with methods like CT which, despite using LoRA, require fine-tuning larger portions of the base LLM and can be resource-intensive (e.g., CT reportedly takes $\sim$4 GPU days on an NVIDIA V100). Furthermore, CCPS avoids some scalability concerns present in other methods. For example, LitCab's projection layer size (hidden_dim $\times$ vocabulary_size) can become very large for LLMs with extensive vocabularies, and its reliance on multiple negative samples per question for its contrastive learning imposes specific data curation requirements. CCPS, on the other hand, uses more compact projection networks and only requires labels of correctness for the LLM's generated answers.

Probing Internal Representational Stability Forms the Core of CCPS's Mechanism. The methodological foundation of CCPS lies in quantifying internal consistency. While prior work has highlighted output consistency (e.g., through multiple sampling of responses) as a strong indicator of LLM reliability (Zhou et al., 2025), such approaches can be computationally expensive. CCPS internalizes this concept by perturbing hidden state representations. The premise is that if an LLM is truly confident, its internal decision-making process for a token should be stable against relevant adversarial nudges. Our results suggest that features derived from this internal stability analysis serve as effective proxies for confidence.

Perturbation-Derived Features Offer Key Insights into LLM Confidence Signals. The SHAP value analyses (Appendix I.5) provide insights into which features derived from our perturbation process are most influential. Consistently

across different LLMs and datasets, the *original entropy* of the LLM's output distribution for a token emerges as an important feature. As expected, higher original entropy typically shows a negative correlation with the prediction of correctness (meaning higher entropy contributes to predicting the answer as incorrect), signifying that greater initial uncertainty in the LLM's choice is indicative of a potentially incorrect answer. More revealingly, many of the top-ranking features are those derived from the *perturbed states*. For instance, the *margin between the logits of the top-ranked and second-ranked tokens after perturbation* often shows a positive correlation with correctness; a larger margin, even under adversarial stress, indicates a more decisive and less ambiguous output from the LLM, which CCPS learns as a sign of confidence. Similarly, a higher *epsilon-to-flip* value, indicating that a larger perturbation magnitude is needed to make the LLM change its predicted token, consistently contributes positively to the confidence score. These findings affirm that the dynamic response to perturbation, not just the initial state, provides critical signals for confidence estimation.

CCPS Demonstrates Consistent Efficacy Across Diverse LLM Architectures. The strong performance of CCPS is not confined to a specific model architecture or size, as it demonstrates effectiveness across `Llama`, `Qwen`, and `Mistral` families (8B to 32B parameters). This consistency, particularly when compared to methods like LitCab which showed variable ECE performance across LLM families in our experiments (Table 1), suggests that the feature set derived from our internal perturbation methodology captures fundamental aspects of LLM decision-making relevant to confidence, regardless of the specific base model.

7 Conclusion

In this work, we introduced CCPS, a novel method for estimating LLM confidence by evaluating the stability of their internal representations when subjected to targeted adversarial perturbations, using features derived from this process with a lightweight classifier. Our approach demonstrated significant improvements over existing methods, consistently achieving superior calibration (measured by ECE and Brier scores) and discriminative ability (evidenced by strong AUCPR and AUROC results). This effectiveness was observed across a diverse range of LLMs, various MMLU and MMLU-Pro task formats (including multiple-choice and open-ended question answering), and differing levels of difficulty. The features derived from the LLM's response to adversarial nudges proved highly indicative of confidence. CCPS offers an effective and lightweight way to assess LLM reliability, requiring no changes to generation or extensive fine-tuning, and marks a promising step toward more trustworthy, interpretable systems.

Limitations

Despite its strong performance, CCPS has limitations. Firstly, the pre-processing stage of quantifying features from perturbation impacts incurs a computational cost. For each token in an answer, this cost includes an initial Jacobian calculation and subsequently, for each of the S perturbations, processing the perturbed hidden state through the LLM's head to obtain perturbed logits. Access to model internals is also a prerequisite for this feature extraction phase. Secondly, feature effectiveness depends on perturbation hyperparameters (e.g., ϵ_{max}, S), which, though optimized in our experiments, may need retuning for different models or tasks. Lastly, the quality of extracted features inherently relies on the meaningfulness of the base LLM's internal representations; if an LLM's hidden states do not systematically encode information related to its certainty, the efficacy of any method probing these states might be constrained.

These limitations also highlight opportunities for improvement. One avenue is using the learned stability signals not just for post-hoc estimation but to directly inform and calibrate the generation process, potentially reducing hallucinations. Additionally, while this work perturbs only the final hidden state, exploring perturbations across different transformer layers may yield richer or complementary indicators of confidence.

Ethical Considerations

While CCPS is developed with the aim of enhancing the reliability and trustworthiness of LLMs, several ethical considerations are relevant to its application and interpretation. A primary concern is the potential for over-reliance on the confidence scores produced. Although CCPS demonstrates improved calibration and discrimination, it is crucial to recognize that no confidence estimation method is perfect. In high-stakes domains, such as medicine, finance, or law, an uncritical acceptance

of automated confidence scores without appropriate human judgment and oversight could lead to adverse outcomes if the underlying LLM makes an error that is not perfectly flagged by the confidence score.

Secondly, the fairness of CCPS across diverse demographic groups and data distributions warrants careful attention during deployment. If the base LLMs, from which internal representations are extracted, contain inherent biases or exhibit differential performance characteristics for certain populations, CCPS's confidence assessments could potentially reflect or even inadvertently amplify these disparities. This could result in confidence scores that are less reliable for some groups than for others, potentially leading to inequitable or unfair consequences. Therefore, the deployment of any confidence estimation method, including CCPS, especially in sensitive applications, should be accompanied by rigorous testing for fairness, ongoing monitoring of its performance across relevant subgroups, and a clear framework emphasizing its role as an assistive tool to augment, not replace, human expertise and critical decision-making.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Amos Azaria and Tom Mitchell. 2023. The internal state of an LLM knows when it's lying. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 967–976, Singapore. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. Piqa: Reasoning about physical commonsense in natural language. *Preprint*, arXiv:1911.11641.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *Preprint*, arXiv:1803.05457.

Michael Han Daniel Han and Unsloth team. 2023. Unsloth.

Marie-Catherine de Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. *Proceedings of Sinn und Bedeutung*, 23(2):107–124.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2024. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595, Mexico City, Mexico. Association for Computational Linguistics.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos QA: Machine reading comprehension with contextual commonsense reasoning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2391–2401, Hong Kong, China. Association for Computational Linguistics.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, and 17 others. 2022. Language models (mostly) know what they know. *Preprint*, arXiv:2207.05221.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Katherine Collins, Arka Pal, Umang Bhatt, Adrian Weller, Samuel Dooley, Micah Goldblum, and Andrew Gordon Wilson. 2024a. Large language models must be taught to know what they don't know. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Sanyam Kapoor, Nate Gruver, Manley Roberts, Arka Pal, Samuel Dooley, Micah Goldblum, and Andrew Wilson. 2024b. Calibration-tuning: Teaching large language models to know what they don't know. In *Proceedings of the 1st Workshop on Uncertainty-Aware NLP (UncertaiNLP 2024)*, pages 1–14, St Julians, Malta. Association for Computational Linguistics.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Xin Liu, Muhammad Khalifa, and Lu Wang. 2024. Litcab: Lightweight language model calibration over short- and long-form responses. In *The twelfth International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *Preprint*, arXiv:1711.05101.

Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 4768–4777, Red Hook, NY, USA. Curran Associates Inc.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Preprint*, arXiv:2406.01574.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. *Preprint*, arXiv:2109.01652.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Xiaoling Zhou, Mingjie Zhang, Zhemg Lee, Wei Ye, and Shikun Zhang. 2025. Hademif: Hallucination detection and mitigation in large language models. In *The Thirteenth International Conference on Learning Representations*.

Appendix

Table of Contents

A Artifact Usage and Creation

Consistency with Intended Use of Existing Artifacts: All existing scientific artifacts employed in this research, including pre-trained LLMs, benchmark datasets (`MMLU`, `MMLU-Pro`, and the constituent datasets of `CT-CHOICE/CT-OE`), and software libraries, were used in a manner consistent with their specified intended uses, primarily for academic research, evaluation, and the development of new methodologies within the field of Natural Language Processing. The use of proprietary models like `GPT-4o-mini` for data labeling was conducted in accordance with its API terms of service for research applications.

Intended Use of Created Artifacts: The scientific artifacts created as part of this work—including the source code for the CCPS method, our trained confidence estimation models, and the derived feature sets—are primarily intended to support academic research. Their release aims to ensure the reproducibility of our findings, encourage further investigation into LLM confidence estimation techniques, and allow the community to build upon our contributions. The use and distribution of any created artifacts that are derivative of existing datasets or models will be governed by terms compatible with the original access conditions and licenses of those foundational resources, particularly ensuring that derivatives of artifacts intended for research remain within research contexts where applicable.

B Feature Set Description

This appendix details the features extracted for analyzing the language model's token-level generative behavior. These features, defined in Table 2, are derived from the model's internal states and its responses to systematic perturbations. The perturbations are applied to a token's hidden state by

incrementally moving it along the normalized Jacobian vector of the actual observed token's log-probability, across a predefined number of steps and range of magnitudes.

B.1 Original State Features

This feature set quantifies the model's baseline predictive characteristics for each token prior to experimental perturbation. These include measures of output probabilities, logits, distribution entropy, prediction margins, and vector norms of internal representations. These features establish a reference for evaluating perturbation effects.

B.2 Overall Perturbation Features

This group comprises scalar Features quantifying specific properties related to the perturbation mechanism itself or its direct consequences. These include the L2 norm of the Jacobian vector, the perturbation magnitude required to alter the model's top-1 predicted token (`epsilon_to_flip_token`), and the integrated effect of perturbations on the log-probability of the token guiding the perturbation direction (PEI value).

B.3 Perturbed State Features

These features describe the model's output characteristics (e.g., token probabilities, distribution entropy, decision margins, as listed in Table 2) evaluated after its hidden states are perturbed. The base metrics are calculated at each discrete perturbation step. Statistical summaries (minimum, maximum, mean, standard deviation) of these per-step metrics are then computed across all applied perturbation magnitudes for a given token. This process summarizes the model's output behavior under varying degrees of targeted hidden state modification.

B.4 Comparison Features (Original vs. Perturbed)

This feature set quantifies the differences between the model's original state (hidden states, logits, probability distributions) and its state after each perturbation step. Base comparison metrics are detailed in Table 2. These metrics, such as changes in log-probabilities, distributional divergences (KL, JS), and vector similarities/distances, are statistically summarized (minimum, maximum, mean, standard deviation) across all perturbation magnitudes. The summaries indicate the extent of change in model representations and outputs due to the applied perturbations.

A total of $D_f = 75$ such features are extracted per token.

C Datasets

This section provides further details on the datasets used for training, validation, and evaluation of our confidence estimation models. All datasets employed in this study are in English. For comprehensive information regarding the original construction, specific domain coverage, linguistic characteristics, and any available demographic details for the underlying public benchmarks (such as MMLU, MMLU-Pro, and the constituent datasets of `CT-CHOICE` and `CT-OE`), we refer readers to their respective original publications, which are cited upon their introduction in the subsequent subsections.

C.1 Training and Validation Datasets

For training and validating our confidence estimation models, we utilize the `CT-CHOICE` and `CT-OE` datasets, designed for multiple-choice and open-ended question-answering formats, respectively. These datasets, generated following the methodology of Kapoor et al. (2024b), aggregate a diverse collection of commonly used public question-answering datasets. Instances from these datasets were formatted to ensure a maximum input sequence length of 1,600 tokens during our training process. The underlying datasets include:

- AI2 Reasoning Challenge (ARC) (Clark et al., 2018)
- Boolean Questions (BoolQ) (Clark et al., 2019)
- CommonsenseQA (Talmor et al., 2019)
- CosmosQA (Huang et al., 2019)
- HellaSwag (Zellers et al., 2019)
- MathQA (Amini et al., 2019)
- Recognizing Textual Entailment (RTE/SNLI) (Bowman et al., 2015)
- Adversarial NLI (Nie et al., 2020)
- OpenBookQA (Mihaylov et al., 2018)
- PIQA (Bisk et al., 2019)
- SciQ (Welbl et al., 2017)

- The CommitmentBank (CB) (de Marneffe et al., 2019)
- Multi-Sentence Reading Comprehension (MultiRC) (Khashabi et al., 2018)
- Choice of Plausible Alternatives (CoPA) (Gordon et al., 2012)
- TREC (Li and Roth, 2002)
- Adversarial Winograd (Winogrande) (Sakaguchi et al., 2021)

C.2 Evaluation Datasets

Our evaluation suite consists of variants of the Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) and MMLU-Pro (Wang et al., 2024) benchmarks, formatted for both multiple-choice and open-ended evaluation.

MMLU-CHOICE and MMLU-OE: These datasets are derived from the standard MMLU benchmark, which covers 57 diverse tasks spanning STEM, humanities, social sciences, and other areas. We created multiple-choice (`MMLU-CHOICE`) and open-ended (`MMLU-OE`) versions following the data processing approach of Kapoor et al. (2024b). The constituent tasks and their respective sample sizes for MMLU are listed in Table 3.

MMLU-PRO-CHOICE: This dataset is the multiple-choice version of MMLU-Pro (Wang et al., 2024), which includes 14 tasks designed with more challenging questions that often require deeper domain knowledge. Unlike the standard MMLU, the structure of MMLU-Pro questions often makes the provided choices an indispensable part of the question's context, meaning it could not be meaningfully converted to an open-ended format without fundamentally altering the nature of the problems. Furthermore, the answer options in MMLU-Pro frequently extend beyond the typical A-D choices, sometimes including E, F, or more. The tasks and their sample sizes for MMLU-Pro are detailed in Table 4.

C.3 Response Generation and Labeling

For all datasets described above, responses from the base LLMs were first generated to create the instances for our confidence estimation task. The user prompt, which includes the question and any contextual information (such as few-shot exemplars), was constructed following the methodology of Kapoor et al. (2024b), to which we refer the reader for further details. We employed specific system prompts for guiding the base LLMs during response generation, as detailed in Table 5. These prompts are similar to those used by Kapoor et al. (2024b) but were slightly refined for improved clarity to the LLMs. In line with their approach, for multiple-choice question-answering datasets (`CT-CHOICE`, `MMLU-CHOICE`, `MMLU-PRO-CHOICE`), answers were generated with a maximum token limit of 1, corresponding to the chosen option letter. For open-ended datasets (`CT-OE`, `MMLU-OE`), responses were generated using greedy decoding with a maximum length of 30 tokens.

Each generated response was subsequently labeled as correct or incorrect. For multiple-choice questions, correctness was determined by a straightforward string match between the LLM's generated option letter and the ground truth option. For open-ended responses, assessing semantic equivalence requires a more nuanced approach. Kapoor et al. (2024b) conducted a comparative analysis of different grading techniques against human evaluations. Their study found that `GPT-4` assessments exhibited a low average absolute difference of 4.5% in accuracy estimation compared to human annotators, while `GPT-3.5 Turbo` also demonstrated strong agreement, with an average difference of 8.7%. Despite the superior performance of `GPT-4`, they ultimately employed `GPT-3.5 Turbo` for their labeling due to expediency. Leveraging their validation of advanced LLMs for this task, and given the availability of even more capable models since their study, we utilized the more recent `GPT-4o-mini` model via its API for assessing the correctness of open-ended LLM responses. This choice was made to ensure the highest quality semantic equivalence judgments. The prompts used for this grading task are detailed in Table 6.

The distribution of these correct and incorrect LLM responses across all datasets, for each base model used in our experiments, is detailed in Table 7.

D Baseline Method Details

This section details the baseline methods implemented for comparison against our proposed CCPS method. Our selection of baselines was guided by the aim to provide a comprehensive benchmark against prominent, recent, and state-of-the-art techniques in LLM confidence estimation, many of which are established through peer-reviewed publi-

cations in highly regarded scientific venues. While the work introducing CT by Kapoor et al. (2024b) provided a valuable starting point by evaluating methods such as P(True), Instruction Tuning (IT), Logit Temperature Scaling (LTS), and a specific variant of SAPLMA (SAPLMA-F), our study expands significantly on this comparison. We include P(IK), which was not part of their direct comparison, and additional SAPLMA variants (SAPLMA-M, SAPLMA-UM) to explore signals from different representational depths. Furthermore, our evaluation framework encompasses a broader range of test conditions, including comprehensive training and testing on both multiple-choice and open-ended formats, and performance on datasets like `MMLU-PRO-CHOICE`, aspects not exhaustively covered for all these prior methods in the context of confidence estimation by Kapoor et al. (2024b). We also incorporate LitCab (Liu et al., 2024), another significant and well-regarded recent contribution in lightweight white-box confidence estimation also originating from a top-tier conference, which provides an important additional point of comparison. For all established baseline methods, we adhered to the architectural descriptions and training configurations reported in their original publications. Common training hyperparameters, such as total steps and optimizer settings, are described in Section 4 (Training Details).

D.1 P(True)

Introduced by Kadavath et al. (2022), P(True) assesses an LLM's self-evaluation of a generated answer. After an LLM generates an answer to an input prompt P, it is presented with the question, "Is the proposed answer correct? a) no b) yes" (referred to as the uncertainty query). The probabilities assigned by the original, frozen LLM to options 'a' and 'b' are then normalized (e.g., via softmax) to derive the confidence score, representing the probability of correctness. This method requires no additional training.

D.2 P(IK)

Also from Kadavath et al. (2022), P(IK) (short for "I Know") estimates the LLM's probability of correctly answering a given question *before* it generates a specific response. This typically involves training a lightweight classifier head on a hidden state representation from the LLM (e.g., the final hidden state after processing the input prompt P) to predict correctness. The output probabilities from this classifier serve as the confidence score.

D.3 Logit Temperature Scaling (LTS)

As described by Jiang et al. (2021), LTS is a post-hoc calibration technique that adjusts a model's output probabilities. It introduces a scalar temperature parameter $\tau > 0$ which is applied to the logits before the LLM's final softmax function. In our application, after the LLM responds to the uncertainty query, the temperature τ is applied to the logits corresponding to the 'a' and 'b' options. The calibrated probability is then $\text{softmax}(\text{logits}_{\text{uncertainty query}}/\tau)$. The temperature τ is optimized on a held-out development set. LTS is computationally very light as it involves learning only a single parameter.

D.4 Instruction Tuning (IT)

Instruction tuning, as introduced by (Wei et al., 2022), involves fine-tuning language models on a collection of tasks framed as natural language instructions. In our setting, this baseline involves fine-tuning the base LLM to respond to the uncertainty query more accurately. The model is trained using Low-Rank Adaptation (LoRA) (Hu et al., 2021), a parameter-efficient fine-tuning technique, to predict the correct option ('a' or 'b') for the uncertainty query, based on ground-truth labels derived from the answer grading phase. While LoRA makes this more efficient than full fine-tuning of all parameters, it remains more resource-intensive than non-fine-tuning methods.

D.5 SAPLMA

SAPLMA (Statement Accuracy Prediction based on Language Model Activations) (Azaria and Mitchell, 2023) trains a lightweight feedforward classifier on LLM hidden state activations to predict statement truthfulness, while the LLM itself remains frozen. SAPLMA's classifier employs a feedforward neural network featuring three hidden layers with decreasing numbers of hidden units (256, 128, 64), each followed by a ReLU activation. Their studies suggest that signals related to an LLM's internal assessment of truthfulness or confidence can manifest at different network depths depending on the model architecture and task. Therefore, while a common approach is to use final hidden states (**SAPLMA-F**), we also implemented variants using activations from the middle layer (**SAPLMA-M**) and an upper-middle layer

(**SAPLMA-UM**) of the LLM to explore these potentially richer representational layers. The output probabilities from these classifiers are used as confidence scores.

D.6 Calibration-Tuning (CT)

Proposed by Kapoor et al. (2024b), CT fine-tunes an LLM (using LoRA) to explicitly predict its answer's correctness in response to the uncertainty query. It uses a classification loss combined with a divergence-based regularizer (such as Jensen-Shannon or KL Divergence) to help maintain the LLM's original generation capabilities. While LoRA reduces the training burden compared to full fine-tuning, CT can still be resource-intensive, reportedly taking about 4 GPU days on an NVIDIA V100 for their experiments. The divergence term, particularly with longer sequences in open-ended tasks, can also be memory-demanding.

D.7 LitCab

This lightweight calibration method by Liu et al. (2024) involves a trainable linear layer of size (hidden_dim $\times$ vocabulary_size) that is attached to the LLM's final hidden states. This layer predicts a bias term which is added to the original output logits of the LLM. LitCab is trained using a contrastive max-margin loss, which typically requires multiple incorrect answer examples per question. The confidence score is then derived from the geometric mean of the adjusted probabilities of the response tokens.

E CCPS Architecture Details

Our CCPS approach employs a feature projection network (E_{MC} for multiple-choice, E_{OE} for open-ended) followed by a classifier head (C). The specific architectures for these components were determined through a systematic hyperparameter search for both MC and OE formats, aimed at optimizing for the loss on validation data. Key training hyperparameters such as learning rate (1×10^{-4}), weight decay (0.1), batch size (32), and training steps were kept consistent during this search, aligned with those detailed in Section 4 (Training Details). The finalized best-performing architectures are detailed below.

E.1 Multiple-Choice Question Answering

For the Multiple-Choice (MC) CCPS model, the hyperparameter search explored various configurations for the contrastive encoder (E_{MC}), including different embedding dimensions, the number and size of hidden layers, and a range of activation functions (ReLU, GeLU, SiLU, ELU, Leaky ReLU). Similarly, various hidden layer structures and activation functions were evaluated for the MLP-based classifier head (C). The selected architecture, which yielded the optimal balance of performance metrics, is as follows: the contrastive encoder (E_{MC}) is an MLP that processes the D_f-dimensional feature vector. It consists of a sequence of linear layers with output dimensions 64, 32, 16, and a final 8-dimensional embedding layer. ELU activation is applied after each layer except the output embedding layer. The subsequent classifier head receives the 8-dimensional embedding and passes it through an MLP with layers having output dimensions 48, 24, 12, each followed by ELU activation, and concludes with a final linear layer producing 2 output logits for classification.

E.2 Open-Ended Question Answering

For the Open-Ended (OE) CCPS model, the hyperparameter search for the contrastive encoder (E_{OE}) covered different embedding dimensions, the number and size of hidden channels for its 1D convolutional layers, various kernel sizes for these convolutional layers, and a range of activation functions (ReLU, GeLU, SiLU, ELU, Leaky ReLU). The MLP-based classifier head (C) also underwent a search over its hidden layer structures and activation functions. The best-performing configuration found is detailed here: the contrastive encoder (E_{OE}) processes sequences of D_f-dimensional token features. It employs two 1D convolutional layers; the first maps the input features to 64 channels (kernel size 3), and the second maps from 64 to 32 channels (kernel size 3). ReLU activation is applied after each convolutional layer. An adaptive max-pooling layer then reduces the sequence to a fixed-size representation, which is projected by a linear layer to a 16-dimensional embedding. The classifier head takes this 16-dimensional embedding, passes it through a linear layer to a 32-dimensional representation with ReLU activation, and finally to an output linear layer producing 2 logits for classification.

F Computational Setup and Resources

All computational experiments were conducted on a GPU cluster equipped with NVIDIA A100-SXM (48GB VRAM) and NVIDIA H200 (141GB

VRAM) GPUs. The allocation of GPU resources and specific setup details for the different confidence estimation methods are outlined below.

F.1 P(True):

This method involves no training. Inference to obtain responses to the uncertainty query was performed using a single NVIDIA A100 GPU for the `Meta-Llama-3.1-8B-Instruct` and `Qwen2.5-14B-Instruct` models, and a single NVIDIA H200 GPU for the `Mistral-Small-24B-Instruct-2501` and `Qwen2.5-32B-Instruct` models.

F.2 P(IK), SAPLMA, LitCab, and CCPS:

Hidden State / Feature Extraction: For these methods, the initial stage of extracting hidden states or features (including perturbation processes for CCPS) from the base LLMs was performed using a single NVIDIA A100 GPU for the `Meta-Llama-3.1-8B-Instruct` and `Qwen2.5-14B-Instruct` models. Due to their larger size, a single NVIDIA H200 GPU was used for the `Mistral-Small-24B-Instruct-2501` and `Qwen 2.5-32B-Instruct` models. This allocation ensured that each base LLM could be loaded onto an appropriate GPU.

Training of Confidence Modules: The subsequent training of the lightweight confidence modules for P(IK), SAPLMA variants, LitCab, and our CCPS classifiers (which typically comprise fewer than 1 million trainable parameters) was conducted on a single NVIDIA A100 GPU for all base LLMs.

F.3 IT and LTS:

For IT, the LoRA-based fine-tuning of the base LLMs on the uncertainty query, and for LTS, the optimization of the temperature parameter, were performed on a single NVIDIA A100 GPU for `Meta-Llama-3.1-8B-Instruct` and `Qwen2.5-14B-Instruct`. For the larger `Mistral-Small-24B-Instruct-2501` and `Qwen 2.5-32B-Instruct` models, these processes utilized a single NVIDIA H200 GPU.

F.4 CT

The LoRA-based fine-tuning process for CT was conducted using 4 NVIDIA A100 GPUs operating in parallel for each combination of base LLM and dataset. This multi-GPU setup, managed with libraries such as Hugging Face Accelerate (Gugger et al., 2022) and DeepSpeed (Rasley et al., 2020) (Zero Redundancy Optimizer Stage 2), was implemented in accordance with the original CT methodology to handle its more intensive training requirements.

G Analysis of Additional Trainable Parameters

This appendix quantifies and compares the *additional* learnable parameters introduced by each evaluated confidence estimation method, including our proposed CCPS, when applied to a base LLM. We first detail the architectural parameters of the base LLMs used, then provide the formulas for calculating additional trainable parameters for each confidence estimation method, followed by the exact parameter counts for the specific LLMs analyzed in our experiments. This analysis supports our claim regarding the lightweight nature of CCPS. All parameter counts include biases unless otherwise specified for asymptotic estimates.

G.1 Base LLM Architectural Parameters

The key architectural dimensions of the base Large Language Models (LLMs) utilized in this study, which influence the number of trainable parameters for certain confidence estimation methods, are provided in Table 8. These include the hidden size (d_h), tokenizer vocabulary size (V), and the number of decoder layers (L).

G.2 Formulation of Additional Trainable Parameters

The number of additional trainable parameters for each confidence estimation method is determined as follows (Table 9). We define $D_f = 75$ as the input feature dimension for CCPS, and $r = 8$ as the rank for LoRA implementations.

For CCPS (MC), the encoder E_{MC} layers are $(D_f, 64)$, $(64, 32)$, $(32, 16)$, $(16, 8)$, and classifier C_{MC} layers are $(8, 48)$, $(48, 24)$, $(24, 12)$, $(12, 2)$. The sum of $h_i h_{i+1}$ (weights) and h_{i+1} (biases) for E_{MC}, and $g_j g_{j+1}$ (weights) and g_{j+1} (biases) for C_{MC} yields the total. For CCPS (OE), the encoder E_{OE} consists of two 1D convolutional layers (first: D_f to 64 channels, kernel 3; second: 64 to 32 channels, kernel 3) and a linear projection layer (32 to 16 dimensions). The classifier head C_{OE} is an MLP (16 to 32 dimensions, then 32 to 2 outputs). The exact calculation for CCPS (OE), including convolutional layer parameters (weights and biases) and MLP parameters, results in approximately 22,000 parameters, as detailed in Appendix E.

G.3 Exact Additional Trainable Parameter Counts

Based on the formulations above and the LLM dimensions in Table 8, the exact number of additional trainable parameters introduced by each method when applied to the different base LLMs is presented in Table 10. For methods like IT and CT, LoRA with rank $r = 8$ is applied to the Query (Q) and Value (V) matrices within each of the L attention blocks of the base LLMs.

G.4 Discussion of Parameter Efficiency

The results presented in Table 10 highlight the parameter efficiency of CCPS. Irrespective of the base LLM's size, our CCPS (MC) method introduces only 9,542 trainable parameters, and the CCPS (OE) variant introduces approximately 22,000 parameters. This contrasts sharply with other methods. For instance, LitCab requires hundreds of millions of parameters (e.g., over 525 million for `Meta-Llama-3.1-8B-Instruct`) due to its vocabulary-sized projection. LoRA-based fine-tuning (IT/CT with $r = 8$) adds several million parameters (e.g., 4.2 million to 10.5 million). SAPLMA, with its MLP architecture, introduces a moderate number of parameters (e.g., approximately 1.1 million for `Meta-Llama-3.1-8B-Instruct`), while simpler probes like P(IK) remain very light (e.g., 8,194 for the same LLM). CCPS remains significantly more parameter-efficient than SAPLMA, LoRA-based methods, and LitCab.

To further illustrate this, Table 11 shows the relative parameter budgets compared to CCPS (MC). CCPS (MC) is approximately 440 to 1,100 times smaller than LoRA-based IT/CT, and 55,000 to 81,000 times smaller than LitCab for the LLMs tested. This extreme parameter efficiency, combined with CCPS's strong performance demonstrated in the main paper, underscores its suitability as a highly scalable solution for confidence estimation on large, frozen LLMs.

H Evaluation Metrics

We assess the performance of our confidence estimation method using a suite of standard metrics. This comprehensive set allows for a nuanced understanding beyond ECE and ACC, which can be less informative for imbalanced datasets often encountered in correctness prediction.

H.1 Expected Calibration Error (ECE)

A model's uncertainties are well-calibrated if they align with empirical probabilities—i.e., an event assigned probability p occurs at rate p in reality. Following Kapoor et al. (2024b), we estimate ECE by binning the predicted confidence score (probability of correctness) for each of n samples into b equally-spaced bins $B = \{B_j\}_{j=1}^{b}$. The empirical ECE estimator is given by:

$$\text{ECE} = \sum_{j=1}^{b} \frac{|B_j|}{n} |\text{conf}(B_j) - \text{acc}(B_j)|$$

where $\text{conf}(B_j)$ is the average predicted confidence of samples in bin B_j and $\text{acc}(B_j)$ is the corresponding ACC (fraction of correct LLM answers) within that bin. Consistent with common practice, we use $b = 10$ bins. An ECE of 0 signifies perfect calibration.

H.2 Brier Score

This measures the mean squared difference between the predicted probability of correctness p_k for sample k and its actual binary outcome o_k (1 if correct, 0 if incorrect), summed over all N samples:

$$\text{Brier Score} = \frac{1}{N} \sum_{k=1}^{N} (p_k - o_k)^2$$

It provides a measure of both calibration and refinement, with lower scores being better.

H.3 Accuracy (ACC)

Refers to the proportion of the LLM's answers that are correct on the given task. While our method estimates confidence in these answers rather than altering them, ACC provides context for the difficulty of the underlying task.

H.4 Area Under the Precision-Recall Curve (AUCPR)

This metric summarizes the trade-off between precision (the proportion of positively predicted instances that are truly positive, $\text{TP}/(\text{TP} + \text{FP})$) and recall (the proportion of actual positive instances that are correctly predicted, $\text{TP}/(\text{TP} + \text{FN})$) for the binary correctness classification task. The confidence score is used as the discrimination threshold, varied to plot the curve. AUCPR is particularly informative for imbalanced datasets where the number of incorrect answers might significantly outweigh correct ones, or vice-versa.

H.5 Area Under the Receiver Operating Characteristic Curve (AUROC)

This evaluates the discriminative ability of the confidence score to distinguish between correct and incorrect answers. It plots the true positive rate (Recall) against the false positive rate ($\mathrm{FP}/(\mathrm{FP}+\mathrm{TN})$) at various threshold settings of the confidence score. An AUROC of 1.0 indicates perfect discrimination, while 0.5 suggests random guessing.

I Extended Results and Analyses

This section provides supplementary results and analyses that further substantiate the findings presented in the main paper. We include comprehensive performance comparisons across all baseline methods, detailed per-LLM and per-task breakdowns, calibration curve visualizations, and feature importance analyses for our CCPS model.

I.1 Per-Dataset Aggregate Performance Tables

To offer a comprehensive comparison of all evaluated methods, including all baselines, Tables 12, 13, and 14 present aggregate performance metrics for the `MMLU-CHOICE`, `MMLU-PRO-CHOICE`, and `MMLU-OE` datasets, respectively. Unlike the main paper's Table 1 which shows mean scores across tasks for selected methods, these tables detail the mean $\pm$ standard deviation for all methods across all evaluated LLMs for each metric, providing insight into the consistency of performance.

I.2 Per-LLM Performance Bar Charts

For a visual comparison of method performance on each specific LLM, Figures 2, 3, 4, and 5 present bar charts. Each figure corresponds to one of the four LLMs used in our experiments, illustrating the performance of every confidence estimation method across the different MMLU variant datasets on all evaluation metrics.

I.3 Calibration Curves

To visually assess the calibration of the confidence scores produced by different methods, we provide calibration curves. Figure 6 offers an overview, displaying calibration curves across all models and MMLU variants. Additionally, Figures 7, 8, 9, and 10 present detailed calibration curves for each specific LLM across the test datasets, allowing for a more granular inspection of calibration performance.

I.4 Per-Task Performance Analysis

For an in-depth understanding of performance at a finer granularity, this section provides per-task results. Figures 11 through 30 illustrate the comparative performance of all methods on every individual task within the MMLU datasets for each of the four base LLMs, across all evaluation metrics (ECE, Brier score, ACC, AUCPR, and AUROC).

I.5 Feature Importance Analysis

To elucidate the contributions of various engineered features to the predictions of our CCPS model, we employed SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) (MIT License). This analysis utilized a model wrapper around our trained CCPS classifiers and a subset of the respective training data as background references for the `shap.KernelExplainer` with a logit link function. For Multiple-Choice (MC) models, which take a single feature vector as input, SHAP values directly indicate the importance of each of the D_f features. The resulting "Feature-SHAP Correlation" plots (Figures 31 through 34 for MC model results) visualize the Pearson correlation between scaled feature values and their SHAP values, where colors typically distinguish positive and negative correlations, indicating how feature magnitudes influence the prediction towards correctness.

Due to the sequential nature of inputs (a matrix of feature vectors per token) for Open-Ended (OE) models, SHAP analysis was adapted to assess feature importance across the initial N tokens (e.g., $N = 10$) of an answer. For each feature type, SHAP values were computed based on its influence at these initial positions and then averaged across these N positions to derive an overall impact score. Consequently, the "Feature-SHAP Correlation" plots for OE models (also presented in Figures 31 through 34 for the respective LLMs' OE results) illustrate the correlation between these position-averaged feature values and their corresponding position-averaged SHAP values.

Original State Features	
original_log_prob_actual	Log-probability of the actual token based on the model's original (unperturbed) output distribution, i.e. $\log P_{\text{original}}(\text{actual\_token})$.
original_prob_actual	Probability of the actual token based on the model's original output distribution, i.e. $P_{\text{original}}(\text{actual\_token})$.
original_logit_actual	Logit value of the actual token from the model's original output.
original_prob_argmax	Highest probability assigned to any token by the original model, i.e. $P_{\text{original}}(\text{argmax\_token})$.
original_logit_argmax	Highest logit value assigned to any token by the original model.
original_entropy	Entropy of the original predictive distribution: $-\sum_i P_{\text{original}}(i) \log P_{\text{original}}(i)$.
original_margin_logit_top1_top2	Difference between top-1 and top-2 logits in the original output.
original_margin_prob_top1_top2	Difference between top-1 and top-2 probabilities in the original output.
original_norm_logits_L2	L2 norm of the original logit vector.
original_std_logits	Standard deviation of the original logit values.
original_norm_hidden_state_L2	L2 norm of the original last hidden state vector.
is_actual_token_original_argmax	Indicator (1/0) if the actual token is the argmax under the original model.
Overall Perturbation Features	
jacobian_norm_token	L2 norm of the Jacobian of the token's log-prob w.r.t. the original hidden state (sensitivity measure).
epsilon_to_flip_token	Minimum perturbation magnitude along the Jacobian direction to change the top-1 token.
pei_value_token	Perturbation-Effect Integration (PEI): total normalized drop in log-prob of the actual token over all perturbation steps.
Perturbed State Features	
perturbed_log_prob_actual	Log-prob of the actual token after hidden-state perturbation, $\log P_{\text{perturbed}}(\text{actual\_token})$.
perturbed_prob_actual	Probability of the actual token after perturbation, $P_{\text{perturbed}}(\text{actual\_token})$.
perturbed_logit_actual	Logit value of the actual token after perturbation.
perturbed_prob_argmax	Highest probability assigned after perturbation.
perturbed_logit_argmax	Highest logit value assigned after perturbation.
perturbed_entropy	Entropy of the perturbed predictive distribution.
perturbed_margin_logit_top1_top2	Difference between top-1 and top-2 logits post-perturbation.
perturbed_norm_logits_L2	L2 norm of the perturbed logit vector.
Comparison Features (Original vs. Perturbed)	
delta_log_prob_actual_from_original	Change in log-prob: $\log P_{\text{original}} - \log P_{\text{perturbed}}$ for the actual token.
did_argmax_change_from_original	Indicator (1/0) if the argmax token changed after perturbation.
kl_div_perturbed_from_original	KL divergence $D_{KL}(P_{\text{original}} \parallel P_{\text{perturbed}})$.
js_div_perturbed_from_original	Jensen-Shannon divergence between original and perturbed distributions.
cosine_sim_logits_perturbed_to_original	Cosine similarity of logit vectors before vs. after perturbation.
cosine_sim_hidden_perturbed_to_original	Cosine similarity of hidden-state vectors before vs. after perturbation.
l2_dist_hidden_perturbed_from_original	L2 distance between hidden-state vectors before vs. after perturbation.

Table 2: Definitions of features employed in this study, grouped by feature set type.

Task Name	Size	Task Name	Size
Abstract Algebra	100	High School Statistics	216
Anatomy	135	High School Us History	204
Astronomy	152	High School World History	237
Business Ethics	100	Human Aging	223
Clinical Knowledge	265	Human Sexuality	131
College Biology	144	International Law	121
College Chemistry	100	Jurisprudence	108
College Computer Science	100	Logical Fallacies	163
College Mathematics	100	Machine Learning	112
College Medicine	173	Management	103
College Physics	102	Marketing	234
Computer Security	100	Medical Genetics	100
Conceptual Physics	235	Miscellaneous	783
Econometrics	114	Moral Disputes	346
Electrical Engineering	145	Moral Scenarios	895
Elementary Mathematics	378	Nutrition	306
Formal Logic	126	Philosophy	311
Global Facts	100	Prehistory	324
High School Biology	310	Professional Accounting	282
High School Chemistry	203	Professional Law	1,534
High School Computer Science	100	Professional Medicine	272
High School European History	165	Professional Psychology	612
High School Geography	198	Public Relations	110
High School Government And Politics	193	Security Studies	245
High School Macroeconomics	390	Sociology	201
High School Mathematics	270	US Foreign Policy	100
High School Microeconomics	238	Virology	166
High School Physics	151	World Religions	171
High School Psychology	545		
		Total	**14,042**

Table 3: Tasks and sample sizes in the MMLU benchmark.

Task Name	Size
Biology	717
Business	789
Chemistry	1,132
Computer Science	410
Economics	844
Engineering	969
Health	818
History	381
Law	1,101
Math	1,351
Other	924
Philosophy	499
Physics	1,299
Psychology	798
Total	**12,032**

Table 4: Tasks and sample sizes in the MMLU-Pro benchmark.

Format	System Prompt
Multiple-Choice	`You are an expert who responds with concise, correct answers. For multiple-choice questions, respond only with the letter of the correct option (e.g., a, b, c, d, ...). Do not include any explanation or additional text.`
Open-Ended	`You are an expert who responds with concise, correct answers. Directly state the answer without phrases like 'the correct answer is'.`

Table 5: System prompts used for base LLM response generation.

Prompt Type	Content
System Prompt	`You are an automated grading assistant helping a teacher grade student answers.`
User Prompt	`The problem is: "{question}"` `The correct answer for this problem is: "{gt_answer}"` `A student submitted the answer: "{llm_answer}"` `The student's answer should be semantically equivalent to the correct answer—that is, it should express the same meaning, even if the wording or format is slightly different. However, answers that are ambiguous, incorrect, or include conflicting or multiple answers should not be considered equivalent. Do not penalize superficial differences (e.g., spelling, synonyms, or phrasing), but ensure the core meaning is preserved.`  `Did the student provide a semantically equivalent answer to the ground truth? Please answer yes or no without any explanation:`

Table 6: Prompts used for GPT-4o-mini-based grading of open-ended responses.

CT-CHOICE

Model	Train			Validation		
	Correct	Incorrect	Total	Correct	Incorrect	Total
Meta-Llama-3.1-8B-Instruct	12,654 (67.8%)	5,996 (32.1%)	18,650	1,688 (84.4%)	312 (15.6%)	2,000
Qwen2.5-14B-Instruct	15,116 (81.0%)	3,534 (18.9%)	18,650	1,796 (89.8%)	204 (10.2%)	2,000
Mistral-Small-24B-Instruct-2501	15,255 (81.8%)	3,395 (18.2%)	18,650	1,787 (89.3%)	213 (10.7%)	2,000
Qwen2.5-32B-Instruct	15,724 (84.3%)	2,926 (15.7%)	18,650	1,828 (91.4%)	172 (8.6%)	2,000

CT-OE

Model	Train			Validation		
	Correct	Incorrect	Total	Correct	Incorrect	Total
Meta-Llama-3.1-8B-Instruct	9,165 (49.5%)	9,369 (50.5%)	18,534	1,014 (50.7%)	986 (49.3%)	2,000
Qwen2.5-14B-Instruct	11,656 (62.9%)	6,878 (37.1%)	18,534	1,221 (61.0%)	779 (39.0%)	2,000
Mistral-Small-24B-Instruct-2501	10,532 (56.8%)	8,002 (43.2%)	18,534	1,145 (57.2%)	855 (42.8%)	2,000
Qwen2.5-32B-Instruct	12,083 (65.2%)	6,451 (34.8%)	18,534	1,201 (60.0%)	799 (40.0%)	2,000

MMLU-CHOICE

Model	Test		
	Correct	Incorrect	Total
Meta-Llama-3.1-8B-Instruct	9,041 (64.4%)	5,001 (35.6%)	14,042
Qwen2.5-14B-Instruct	10,898 (77.6%)	3,144 (22.4%)	14,042
Mistral-Small-24B-Instruct-2501	11,231 (80.0%)	2,811 (20.0%)	14,042
Qwen2.5-32B-Instruct	11,488 (81.8%)	2,554 (18.2%)	14,042

MMLU-PRO-CHOICE

Model	Test		
	Correct	Incorrect	Total
Meta-Llama-3.1-8B-Instruct	4,135 (34.4%)	7,897 (65.6%)	12,032
Qwen2.5-14B-Instruct	6,187 (51.4%)	5,845 (48.6%)	12,032
Mistral-Small-24B-Instruct-2501	6,523 (54.2%)	5,509 (45.8%)	12,032
Qwen2.5-32B-Instruct	6,870 (57.1%)	5,162 (42.9%)	12,032

MMLU-OE

Model	Test		
	Correct	Incorrect	Total
Meta-Llama-3.1-8B-Instruct	4,225 (30.1%)	9,817 (69.9%)	14,042
Qwen2.5-14B-Instruct	6,386 (45.5%)	7,656 (54.5%)	14,042
Mistral-Small-24B-Instruct-2501	6,338 (45.1%)	7,704 (54.9%)	14,042
Qwen2.5-32B-Instruct	6,814 (48.5%)	7,228 (51.5%)	14,042

Table 7: Distribution of correct and incorrect responses across CT-CHOICE, CT-OE, and MMLU variants.

Table 8: Architectural dimensions for the base LLMs used.

Base LLM	d_h	V	L
`Meta-Llama-3.1-8B-Instruct`	4,096	128,256	32
`Qwen2.5-14B-Instruct`	5,120	152,064	48
`Mistral-Small-24B-Instruct-2501`	5,120	131,072	40
`Qwen2.5-32B-Instruct`	5,120	152,064	64

Table 9: Formulas for additional trainable parameters introduced by each method.

Method	**Trainable Component(s)**	**Formula for Parameters (incl. Biases)**
P(True)	None (prompting only)	0
LTS	Temperature scalar τ	1
P(IK)	Linear layer ($d_h \to 2$)	$2d_h + 2$
SAPLMA	MLP ($d_h \to 256 \to 128 \to 64 \to 2$)	$256d_h + (256\times128+128) + (128\times64+64) + (64\times2+2)$ $= 256d_h + 41,282$
IT & CT (LoRA)	LoRA layers (adapting Q & V matrices in all L layers, rank r)	$2L \cdot (d_h r + r d_h) = 4Ld_h r$
LitCab	Linear bias layer ($d_h \to V$)	$d_h V + V$
CCPS (MC)	Encoder E_{MC} + Head C_{MC} (MLPs) E_{MC} widths: $(D_f, 64, 32, 16, 8)$ C_{MC} widths: $(8, 48, 24, 12, 2)$	$\sum(h_i h_{i+1} + h_{i+1}) + \sum(g_j g_{j+1} + g_{j+1})$
CCPS (OE)	Encoder E_{OE} + Head C_{OE}	(See text for detailed breakdown)

Table 10: Additional trainable parameters introduced by each confidence estimation method per base LLM (CCPS values for MC variant; LoRA rank $r = 8$ adapting Q and V matrices in all L layers).

Base LLM	**P(True)**	**LTS**	**P(IK)**	**SAPLMA**	**IT/CT (LoRA-r)**	**LitCab**	**CCPS (MC)**
`Meta-Llama-3.1-8B-Instruct`	0	1	8,194	1,089,858	4,194,304	525,464,832	**9,542**
`Qwen2.5-14B-Instruct`	0	1	10,242	1,352,002	7,864,320	778,719,744	**9,542**
`Mistral-Small-24B-Instruct`	0	1	10,242	1,352,002	6,553,600	671,219,712	**9,542**
`Qwen2.5-32B-Instruct`	0	1	10,242	1,352,002	10,485,760	778,719,744	**9,542**

Table 11: Relative trainable parameter budgets with respect to CCPS (MC variant; $\downarrow$ indicates better/fewer parameters).

Base LLM	**LitCab $\div$ CCPS**	**IT/CT LoRA-r $\div$ CCPS**
`Meta-Llama-3.1-8B-Instruct`	$55,069\times$	$440\times$
`Qwen2.5-14B-Instruct`	$81,610\times$	$824\times$
`Mistral-Small-24B-Instruct-2501`	$70,344\times$	$687\times$
`Qwen2.5-32B-Instruct`	$81,610\times$	$1,099\times$

Model	Method	ECE ↓	BRIER ↓	ACC ↑	AUCPR ↑	AUROC ↑
Meta-Llama-3.1-8B-Instruct	P(True)	35.9±5.7	39.4±4.2	45.4±5.9	66.0±14.6	49.2±5.7
	P(IK)	18.9±9.6	25.4±2.4	63.9±14.8	65.3±14.9	49.8±1.8
	LTS	28.9±6.6	34.5±3.7	44.6±6.9	66.6±14.1	50.1±4.3
	IT	33.4±5.3	37.5±3.7	47.2±4.9	66.5±14.7	49.8±5.0
	SAPLMA-M	17.9±9.7	24.8±2.7	64.9±15.1	64.9±15.3	49.5±3.1
	SAPLMA-UM	18.1±9.8	24.9±2.7	64.9±15.1	64.6±15.4	49.3±3.3
	SAPLMA-F	18.2±9.8	24.9±2.6	64.9±15.0	65.0±15.0	49.6±2.3
	CT	10.7±6.7	21.1±5.7	67.8±12.2	74.2±15.5	62.8±8.0
	LitCab	10.9±4.8	18.1±5.5	73.2±8.7	84.0±13.5	**77.1**±8.2
	CCPS	**6.5**±3.9	**17.1**±4.7	**73.4**±8.5	**84.1**±13.5	**77.1**±8.5
Qwen2.5-14B-Instruct	P(True)	47.0±6.2	47.0±4.8	41.3±6.4	79.2±12.5	51.2±5.8
	P(IK)	25.1±13.0	24.1±3.1	76.8±12.2	78.3±12.1	49.9±2.4
	LTS	41.5±6.5	43.0±4.3	38.6±6.2	78.9±12.6	49.7±5.7
	IT	44.7±5.9	44.0±5.2	45.7±7.1	79.4±12.4	50.4±6.7
	SAPLMA-M	23.8±13.2	23.0±4.0	78.1±12.1	78.4±12.3	50.5±3.0
	SAPLMA-UM	23.7±13.2	23.0±4.0	78.2±12.1	78.4±12.1	50.3±2.4
	SAPLMA-F	24.0±12.9	23.0±3.7	78.1±12.1	78.5±12.2	50.3±3.0
	CT	12.1±8.1	17.0±8.1	78.6±11.5	84.7±10.9	64.8±9.1
	LitCab	45.6±11.3	20.0±10.8	78.3±12.0	83.7±10.2	65.3±5.4
	CCPS	**6.3**±3.7	**13.1**±5.8	**80.2**±9.5	**92.1**±8.1	**81.6**±7.0
Mistral-Small-24B-Instruct-2501	P(True)	42.1±8.5	43.3±5.7	38.1±7.9	80.5±12.1	49.3±8.0
	P(IK)	12.4±9.3	17.8±8.7	73.9±18.8	82.6±12.2	56.3±8.9
	LTS	36.2±9.0	38.3±4.7	36.1±8.4	80.2±12.6	49.2±6.2
	IT	37.3±7.3	39.4±5.0	42.9±7.7	81.3±12.0	49.8±7.9
	SAPLMA-M	24.5±14.0	22.5±4.0	79.8±12.9	79.9±12.8	49.8±2.0
	SAPLMA-UM	24.6±14.1	22.5±4.1	79.8±12.9	80.1±12.9	50.6±2.9
	SAPLMA-F	25.2±14.3	22.9±4.1	79.8±12.9	79.8±12.9	49.8±2.3
	CT	8.2±7.4	15.5±7.8	79.6±13.1	83.3±11.5	56.5±7.6
	LitCab	13.5±6.7	15.1±7.4	79.5±9.8	91.5±8.4	78.2±8.0
	CCPS	**5.8**±3.2	**11.5**±6.0	**83.0**±10.3	**93.1**±7.8	**83.3**±7.6
Qwen2.5-32B-Instruct	P(True)	44.0±7.0	45.7±5.5	41.9±7.4	84.0±10.3	52.1±7.3
	P(IK)	28.6±12.7	23.5±4.2	81.7±10.7	82.6±10.4	49.9±2.9
	LTS	37.1±6.7	40.2±4.4	41.9±7.4	84.1±10.3	52.2±7.3
	IT	41.9±7.6	44.0±6.2	44.6±8.3	84.6±10.5	54.8±7.6
	SAPLMA-M	27.3±13.2	22.7±4.7	82.3±10.6	82.4±10.6	49.7±4.2
	SAPLMA-UM	27.7±12.8	22.8±4.7	82.3±10.6	82.3±10.7	49.4±3.6
	SAPLMA-F	27.2±12.9	22.5±4.5	82.3±10.6	82.4±10.7	49.9±2.8
	CT	45.2±7.0	46.9±5.1	37.2±6.1	84.3±10.1	51.6±8.0
	LitCab	43.2±11.0	15.9±9.3	82.6±10.4	87.9±7.9	67.2±6.5
	CCPS	**6.3**±3.1	**10.8**±5.2	**84.1**±8.9	**94.1**±5.9	**82.8**±6.9

Table 12: Complete performance metrics for the MMLU-CHOICE dataset. Arrows indicate whether lower (↓) or higher (↑) values are better. All values are percentages and show mean ± standard deviation. Best values per model are bolded.

Model	Method	ECE ↓	BRIER ↓	ACC ↑	AUCPR ↑	AUROC ↑
Meta-Llama-3.1-8B-Instruct	P(True)	25.3±6.7	33.1±4.5	54.8±6.8	37.1±11.7	49.8±2.1
	P(IK)	41.7±15.6	44.1±11.5	38.2±11.7	37.3±13.7	49.9±3.2
	LTS	17.0±6.7	29.1±3.5	55.4±7.0	36.9±12.3	49.9±2.2
	IT	26.8±4.7	33.8±2.9	52.8±4.8	37.7±12.2	50.0±2.7
	SAPLMA-M	40.4±14.0	40.3±8.6	36.7±12.7	37.3±13.0	50.1±1.8
	SAPLMA-UM	41.0±14.3	41.0±9.1	36.7±12.7	37.5±13.3	50.3±3.0
	SAPLMA-F	40.2±14.9	40.7±10.0	36.8±12.7	37.2±12.9	50.3±1.8
	CT	21.5±11.5	29.8±5.9	50.4±11.7	43.7±14.4	57.3±4.4
	LitCab	16.6±2.9	24.7±2.6	66.1±4.2	51.7±18.4	63.6±9.0
	CCPS	**4.5**±2.1	**20.0**±2.2	**70.4**±4.0	**55.2**±19.4	**67.9**±8.1
Qwen2.5-14B-Instruct	P(True)	33.7±7.1	38.6±4.8	49.9±6.0	55.4±12.6	51.4±1.3
	P(IK)	27.3±11.4	33.9±8.2	53.6±11.5	53.5±13.4	49.1±2.3
	LTS	26.7±7.4	34.5±4.5	49.3±6.7	54.6±11.8	50.7±2.2
	IT	33.5±6.0	38.3±3.6	50.5±4.2	55.6±12.1	51.1±2.4
	SAPLMA-M	28.1±13.2	33.4±8.6	53.4±12.5	54.1±13.3	50.1±3.0
	SAPLMA-UM	27.4±13.5	33.0±8.6	53.5±12.5	53.8±13.0	49.9±3.1
	SAPLMA-F	25.7±12.3	32.1±7.7	53.4±12.5	53.8±12.5	49.3±2.8
	CT	20.4±10.3	28.7±6.3	55.6±11.4	59.4±12.9	56.6±3.5
	LitCab	49.7±4.2	38.3±8.8	55.3±11.6	66.2±10.1	68.0±3.7
	CCPS	**4.2**±1.8	**20.1**±2.9	**69.2**±5.4	**75.8**±10.5	**74.0**±4.8
Mistral-Small-24B-Instruct-2501	P(True)	32.0±8.1	37.2±5.0	46.9±7.3	57.5±12.3	50.2±2.2
	P(IK)	32.3±11.4	36.3±9.7	56.1±10.9	57.4±13.4	50.6±2.1
	LTS	24.7±7.6	32.7±3.7	46.2±7.2	56.6±12.3	49.2±1.7
	IT	31.2±6.7	36.2±3.9	47.0±6.1	58.4±12.0	50.3±2.8
	SAPLMA-M	24.5±12.1	30.7±8.0	56.7±12.4	57.0±13.3	49.9±2.8
	SAPLMA-UM	24.5±11.7	30.7±8.0	56.7±12.4	57.6±13.4	50.6±3.1
	SAPLMA-F	25.1±12.8	31.4±8.5	56.7±12.4	56.8±12.2	49.8±2.2
	CT	17.8±9.7	27.4±5.9	58.2±11.6	60.1±13.1	54.3±3.1
	LitCab	32.2±3.1	34.6±3.2	57.0±3.7	66.2±12.8	60.1±5.0
	CCPS	**4.5**±1.9	**18.6**±3.3	**71.3**±6.4	**79.5**±9.4	**77.2**±5.2
Qwen2.5-32B-Instruct	P(True)	34.6±6.8	39.5±4.9	46.1±5.9	60.1±12.2	50.3±2.7
	P(IK)	23.6±9.9	30.8±7.7	58.0±10.8	59.5±11.9	50.2±2.5
	LTS	27.5±6.7	34.8±3.9	46.1±5.9	60.1±12.2	50.3±2.7
	IT	36.6±6.7	40.9±5.3	45.9±5.9	60.1±12.1	51.0±2.7
	SAPLMA-M	24.8±12.0	30.5±8.3	59.3±11.8	59.9±12.1	49.9±2.8
	SAPLMA-UM	26.9±12.1	31.8±8.9	59.3±11.8	60.2±12.0	49.8±3.3
	SAPLMA-F	23.7±11.2	30.0±7.9	59.3±11.8	59.4±11.6	49.5±2.7
	CT	38.0±8.5	41.6±6.4	44.8±7.2	60.5±11.3	49.9±2.7
	LitCab	48.4±3.5	33.7±8.7	60.8±11.0	72.7±8.8	70.3±4.7
	CCPS	**4.6**±2.1	**18.5**±3.4	**71.8**±6.1	**82.4**±7.7	**77.8**±4.7

Table 13: Complete performance metrics for the MMLU-PRO-CHOICE dataset. Arrows indicate whether lower (↓) or higher (↑) values are better. All values are percentages and show mean ± standard deviation. Best values per model are bolded.

Model	Method	ECE ↓	BRIER ↓	ACC ↑	AUCPR ↑	AUROC ↑
Meta-Llama-3.1-8B-Instruct	P(True)	25.9±7.0	32.0±5.2	56.0±7.7	29.9±12.5	46.2±5.8
	P(IK)	22.6±12.0	26.6±5.0	30.5±11.8	29.9±11.5	49.8±1.1
	LTS	27.9±5.6	34.0±3.5	47.8±4.9	31.5±13.6	47.5±5.9
	IT	27.8±5.9	33.2±4.6	53.9±6.1	30.6±13.3	47.2±6.0
	SAPLMA-M	23.0±12.5	26.6±5.2	67.0±15.7	29.6±11.3	49.9±0.9
	SAPLMA-UM	22.8±11.7	26.2±3.5	29.6±11.3	29.6±11.3	49.9±0.9
	SAPLMA-F	22.5±11.5	26.1±3.5	29.7±11.4	29.7±11.4	49.8±1.4
	CT	8.8±6.4	21.1±4.9	65.3±11.4	48.9±16.8	**70.9**±7.5
	LitCab	8.8±7.6	22.5±4.8	65.3±9.1	46.2±13.8	66.0±9.6
	CCPS	**8.0**±5.7	**20.2**±3.8	**69.5**±8.6	**49.4**±15.9	69.3±7.8
Qwen2.5-14B-Instruct	P(True)	33.9±7.6	36.9±5.9	54.1±7.4	46.3±12.6	52.6±5.0
	P(IK)	14.1±9.9	26.3±4.3	55.8±12.7	42.8±12.2	49.5±1.9
	LTS	27.8±5.4	32.8±4.0	55.9±5.6	49.2±13.1	56.5±6.0
	IT	33.6±6.1	36.7±4.8	55.0±6.0	47.5±13.9	54.0±5.9
	SAPLMA-M	14.9±10.8	26.4±4.2	42.8±12.3	42.8±12.3	49.9±0.9
	SAPLMA-UM	14.6±9.9	26.1±3.0	42.8±12.3	42.7±12.3	49.9±0.9
	SAPLMA-F	14.7±10.1	26.2±3.3	42.8±12.3	42.8±12.4	49.9±1.0
	CT	9.4±5.6	22.6±4.0	63.4±8.4	**61.7**±14.3	**69.3**±7.7
	LitCab	34.4±10.3	37.0±7.3	49.4±10.1	56.8±13.4	62.5±6.8
	CCPS	**6.7**±3.5	**22.5**±2.0	**63.6**±6.8	59.0±12.7	66.6±6.8
Mistral-Small-24B-Instruct-2501	P(True)	28.0±8.9	33.5±6.7	55.5±8.7	44.6±13.3	49.8±4.5
	P(IK)	19.9±12.7	29.7±7.4	52.5±11.1	46.3±14.4	52.7±5.2
	LTS	19.4±6.3	29.3±4.0	55.2±6.7	46.1±13.8	50.8±5.3
	IT	26.2±7.9	32.5±5.6	55.2±7.4	45.5±13.6	50.6±4.5
	SAPLMA-M	15.1±10.9	26.2±3.3	42.6±13.0	42.9±12.9	50.2±1.0
	SAPLMA-UM	15.2±11.0	26.3±3.5	42.6±13.0	42.8±13.0	50.1±0.8
	SAPLMA-F	14.9±10.9	26.2±3.4	42.6±13.0	42.7±13.0	50.0±1.6
	CT	10.8±5.4	22.8±3.4	62.2±8.3	60.7±15.8	68.2±8.0
	LitCab	11.2±5.0	24.6±3.1	60.2±6.8	60.5±13.3	66.4±6.5
	CCPS	**6.8**±2.6	**20.8**±2.6	**67.6**±6.0	**64.7**±13.2	**71.4**±6.8
Qwen2.5-32B-Instruct	P(True)	36.3±4.6	38.0±3.7	54.8±4.3	53.8±12.9	57.1±5.5
	P(IK)	13.1±10.4	26.3±4.5	52.8±12.5	46.5±12.3	49.9±0.6
	LTS	29.5±4.8	34.4±3.4	53.7±4.0	52.7±13.3	55.5±5.5
	IT	33.2±7.1	37.3±5.3	52.7±6.8	49.0±12.7	51.6±4.5
	SAPLMA-M	13.6±10.0	26.2±3.7	46.2±12.5	46.2±12.6	49.9±0.6
	SAPLMA-UM	13.7±10.4	26.3±4.1	46.1±12.5	46.2±12.5	49.8±1.2
	SAPLMA-F	13.7±10.2	26.3±3.8	46.1±12.5	46.2±12.5	49.8±0.8
	CT	22.9±4.7	31.1±3.5	57.1±5.2	52.9±12.8	56.3±5.7
	LitCab	28.4±8.1	33.2±5.5	52.7±8.5	60.2±13.0	62.3±7.6
	CCPS	**8.7**±4.9	**23.3**±2.1	**62.6**±6.8	**62.0**±11.8	**66.4**±5.8

Table 14: Complete performance metrics for the MMLU-OE dataset. Arrows indicate whether lower (↓) or higher (↑) values are better. All values are percentages and show mean ± standard deviation. Best values per model are bolded.

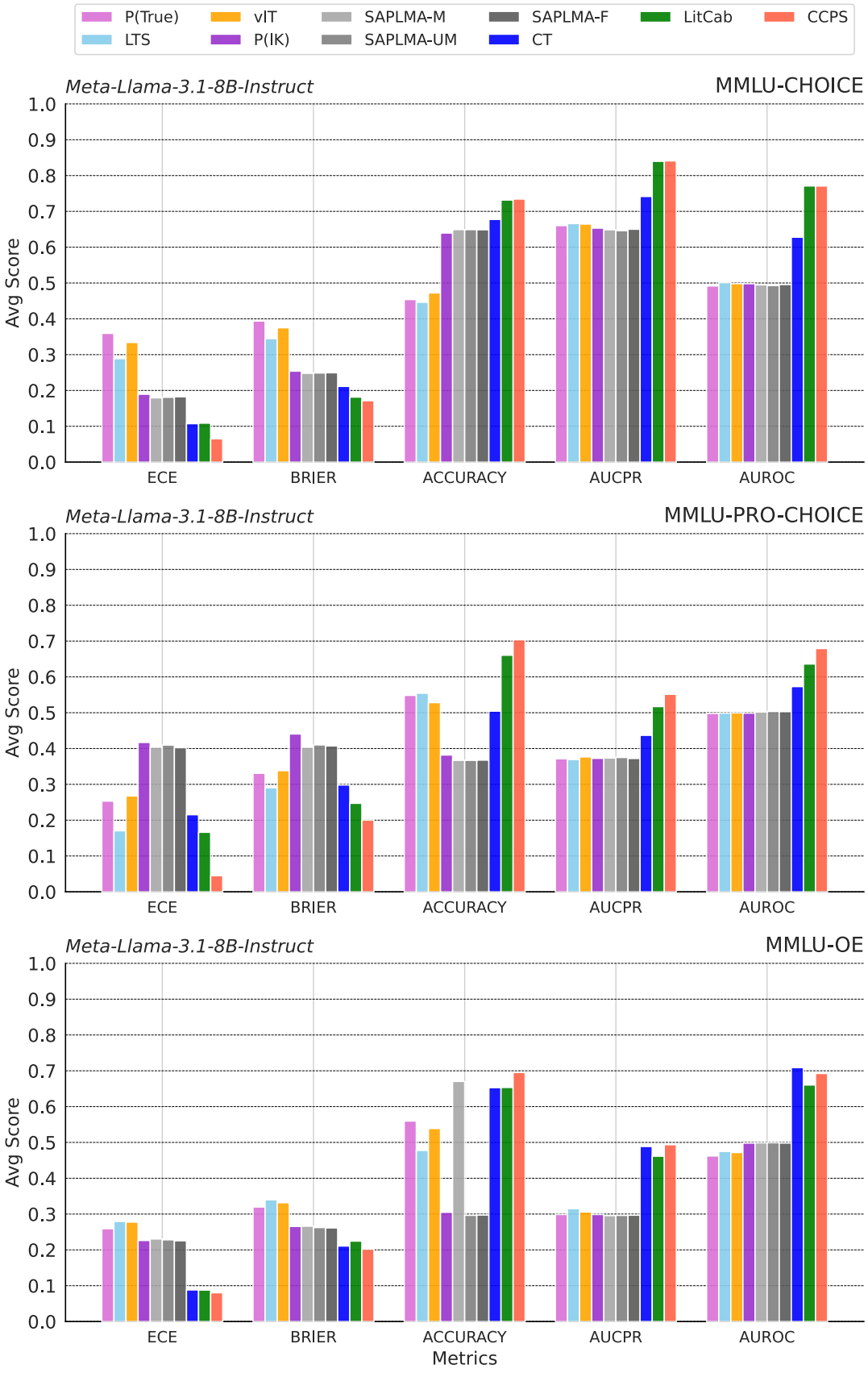

Figure 2: Performance comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across MMLU variants.

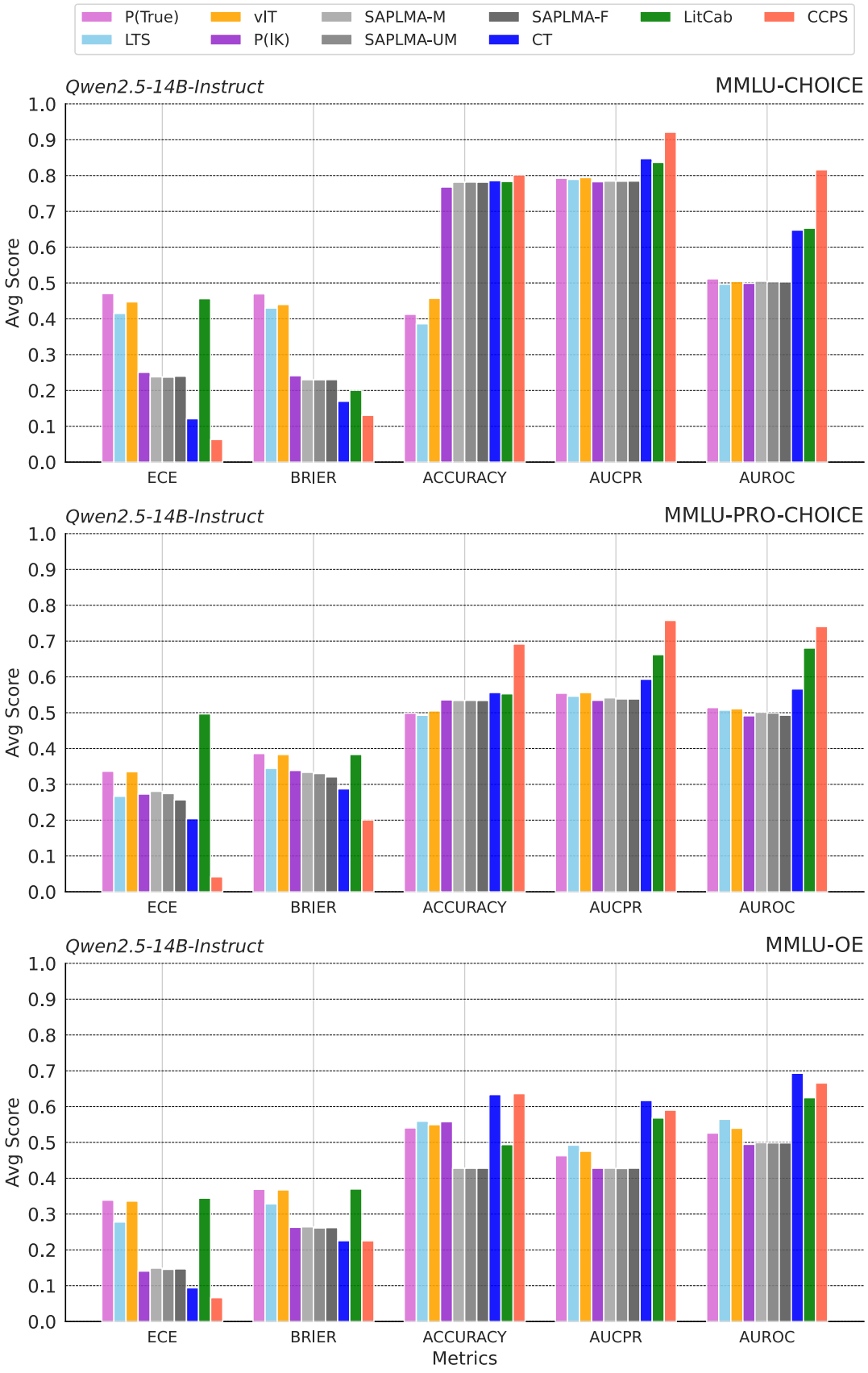

Figure 3: Performance comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across MMLU variants.

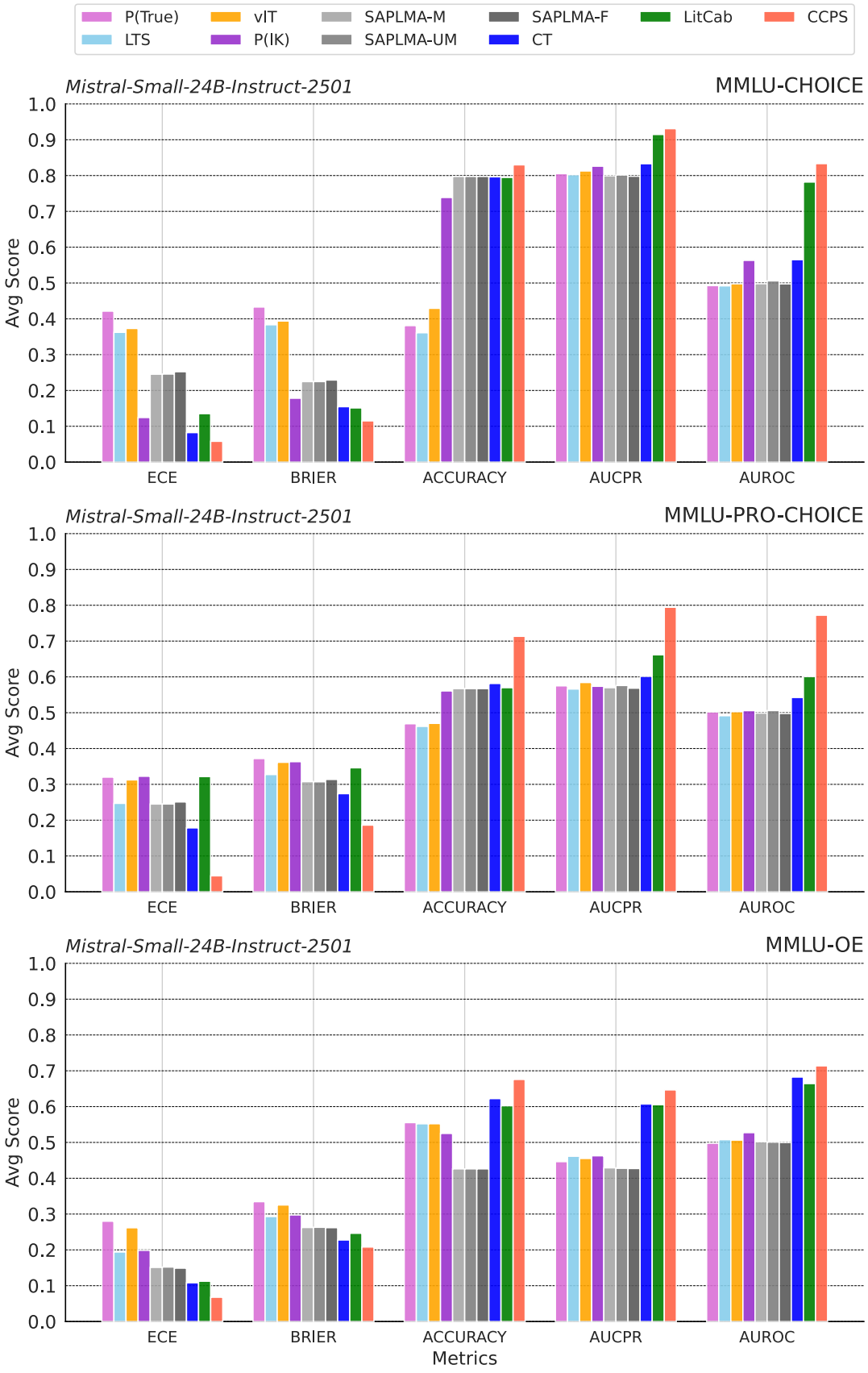

Figure 4: Performance comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across MMLU variants.

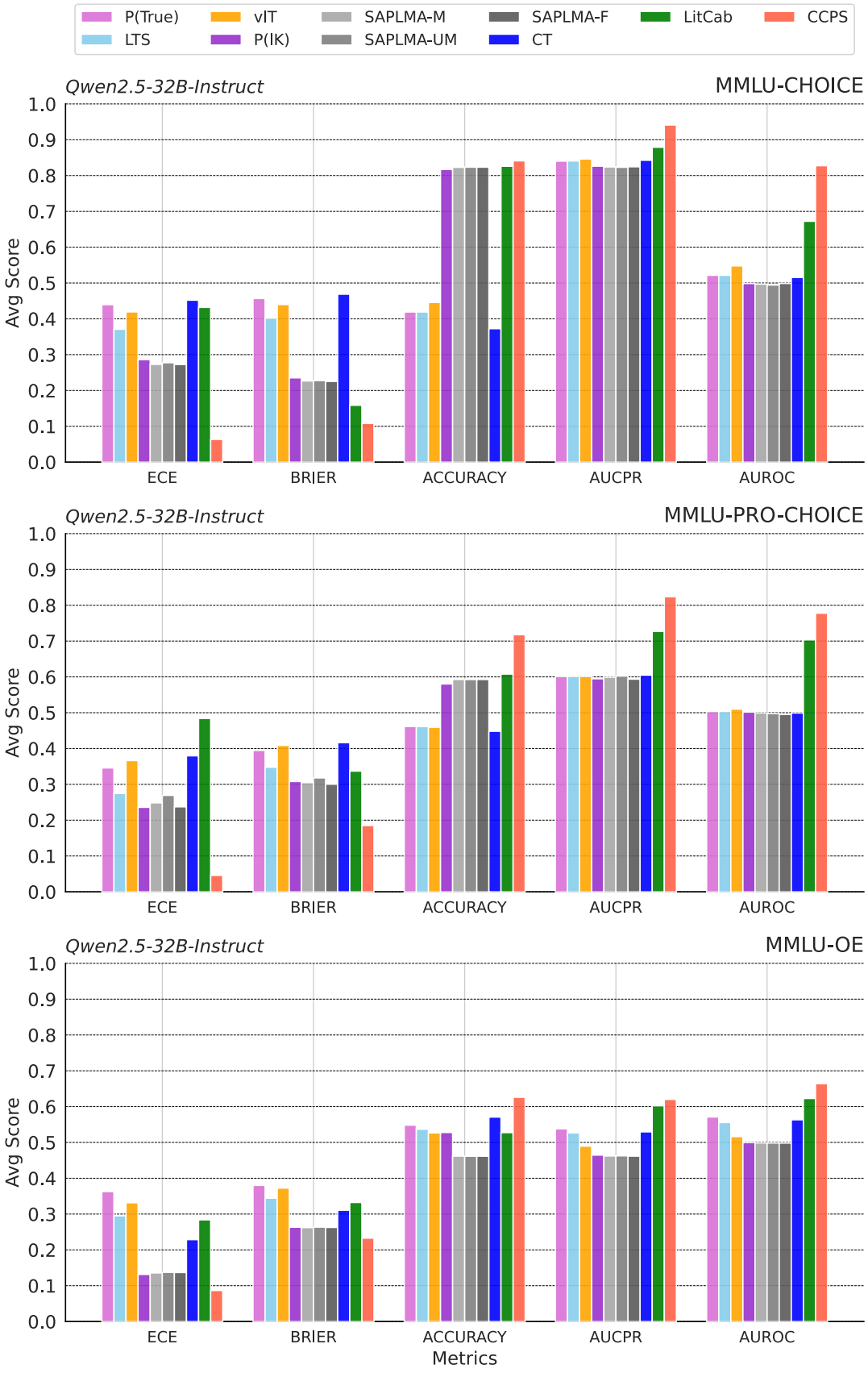

Figure 5: Performance comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across MMLU variants.

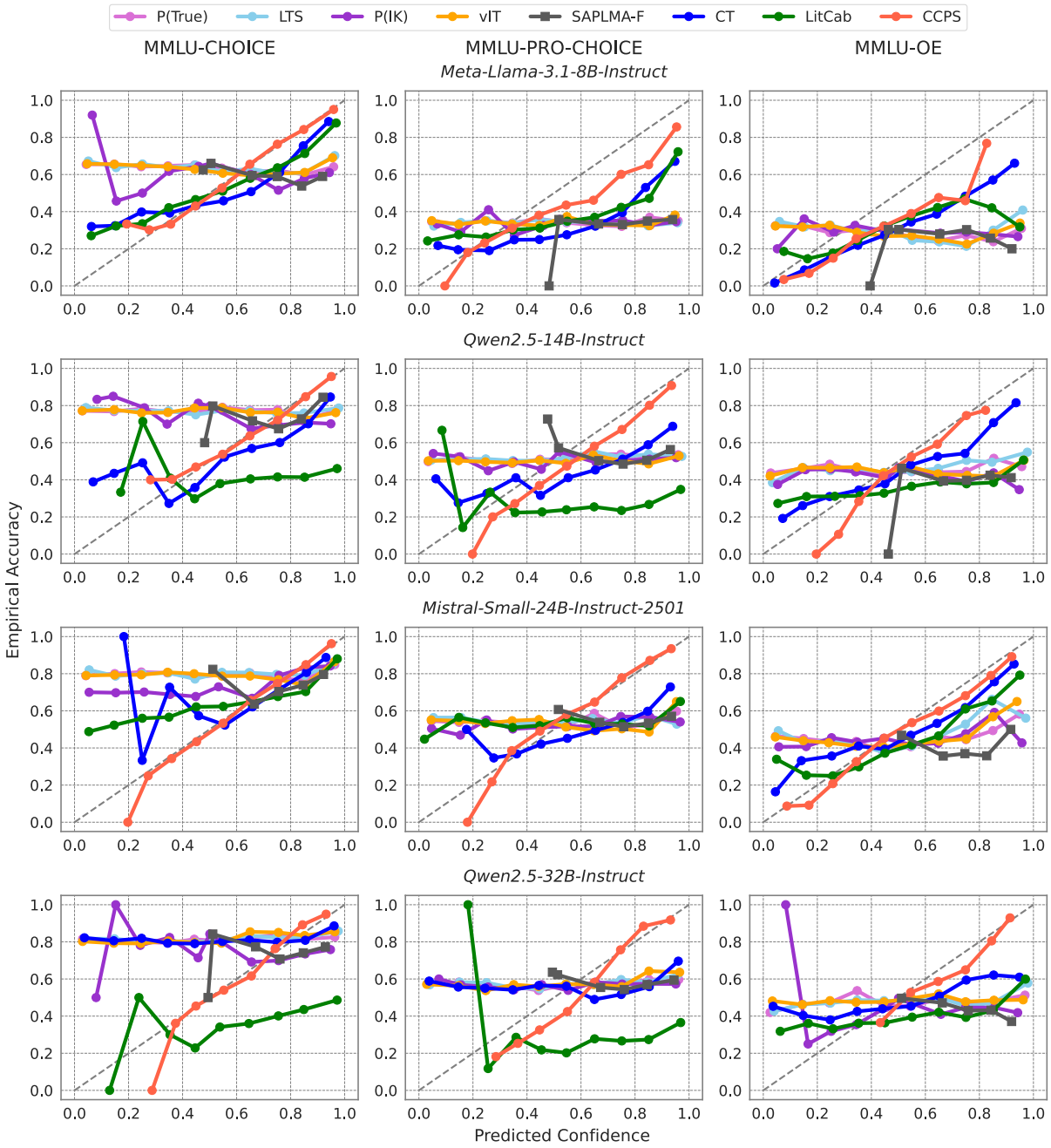

Figure 6: Calibration curves of confidence estimation methods across all models and MMLU variants.

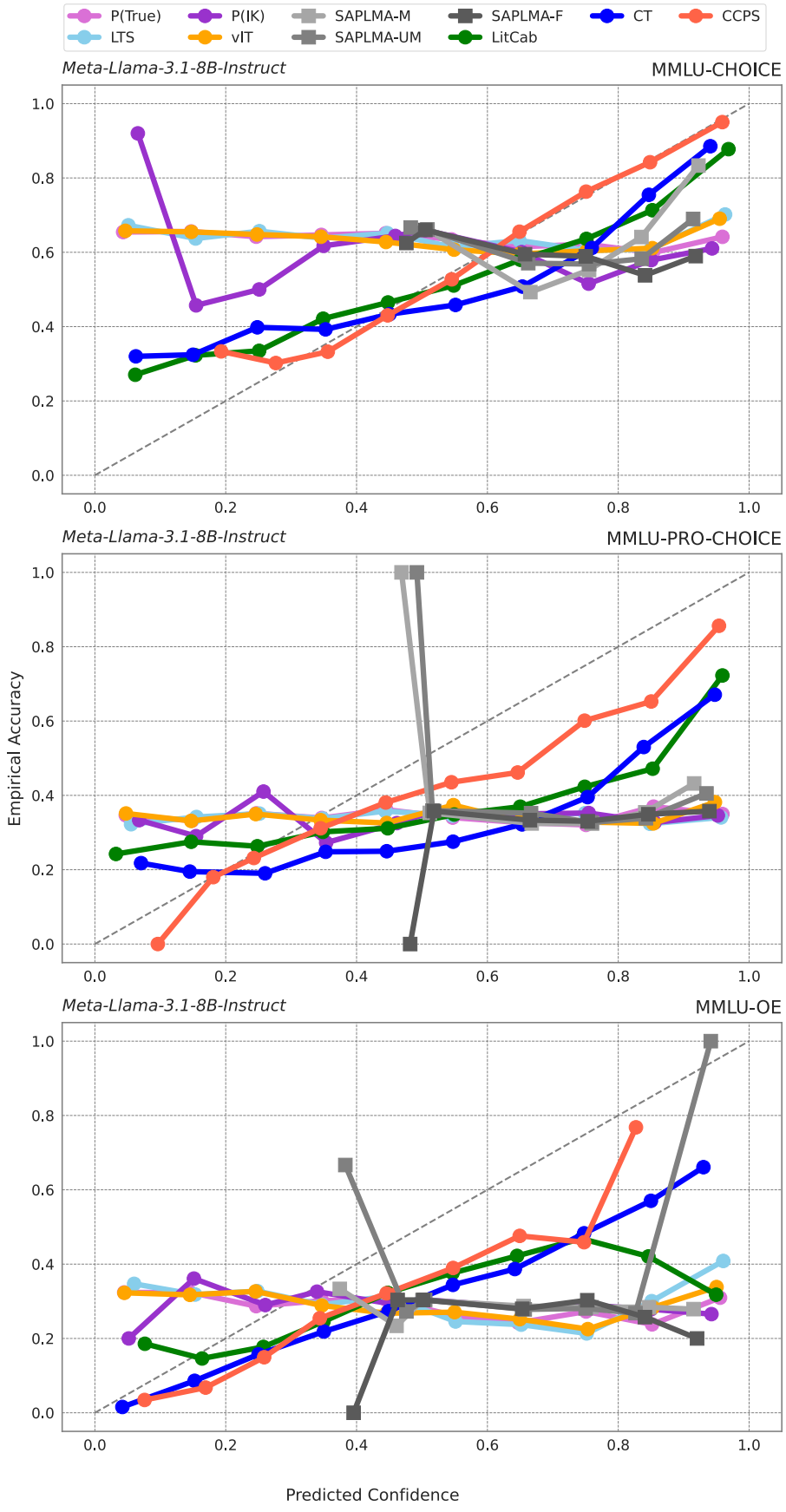

Figure 7: Calibration curves of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across MMLU variants.

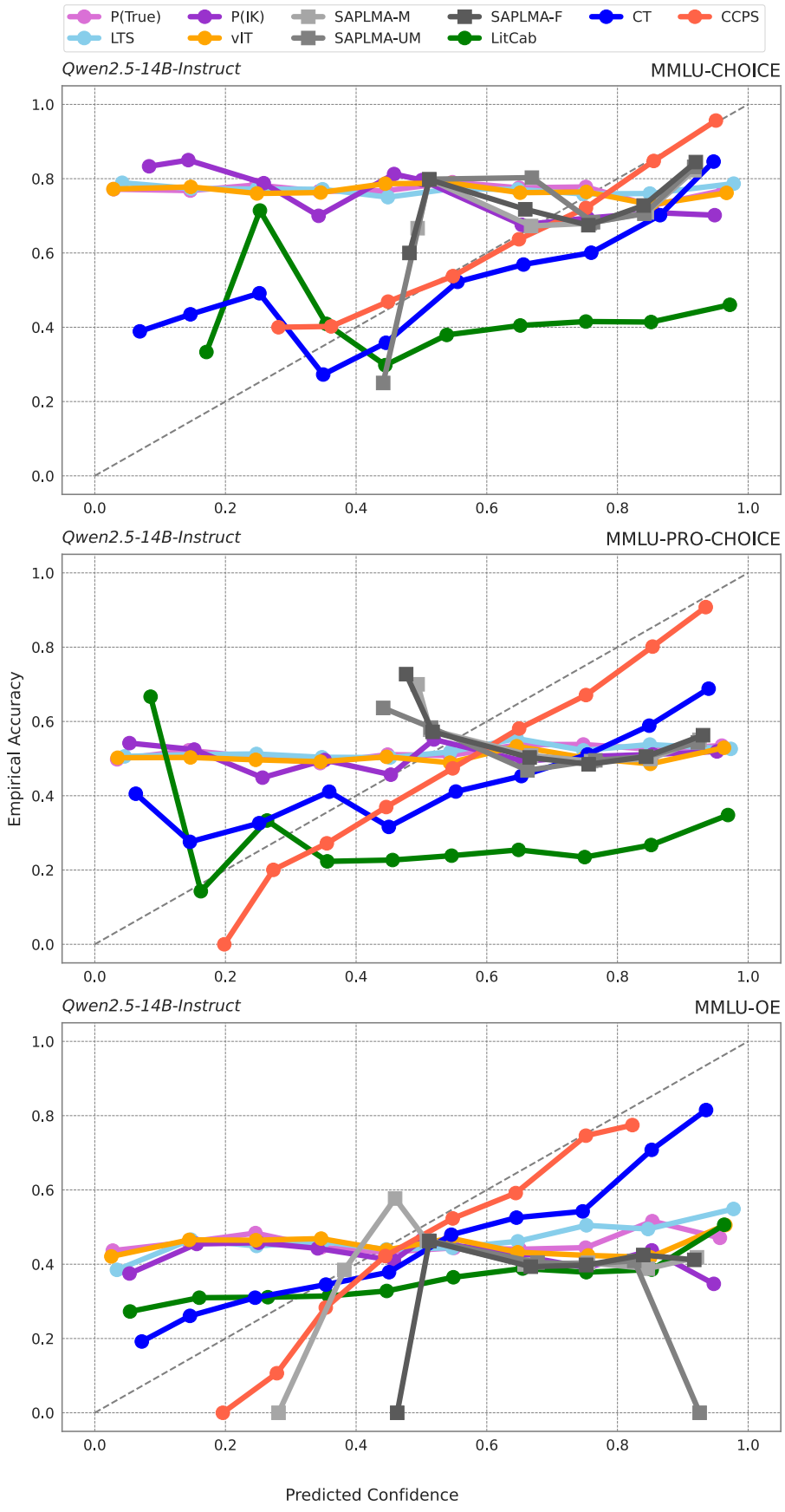

Figure 8: Calibration curves of confidence estimation methods on `Qwen2.5-14B-Instruct` across MMLU variants.

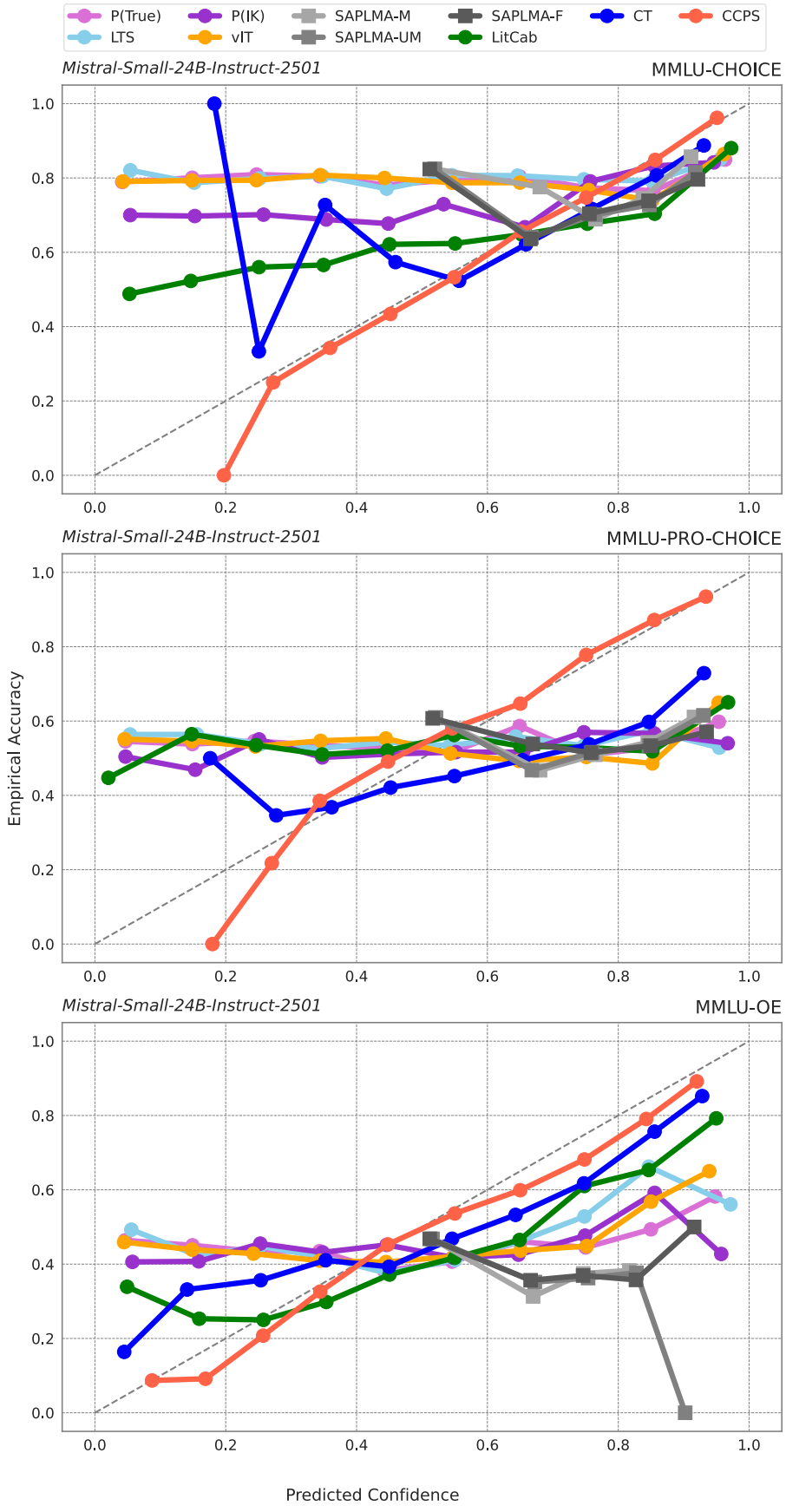

Figure 9: Calibration curves of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across MMLU variants.

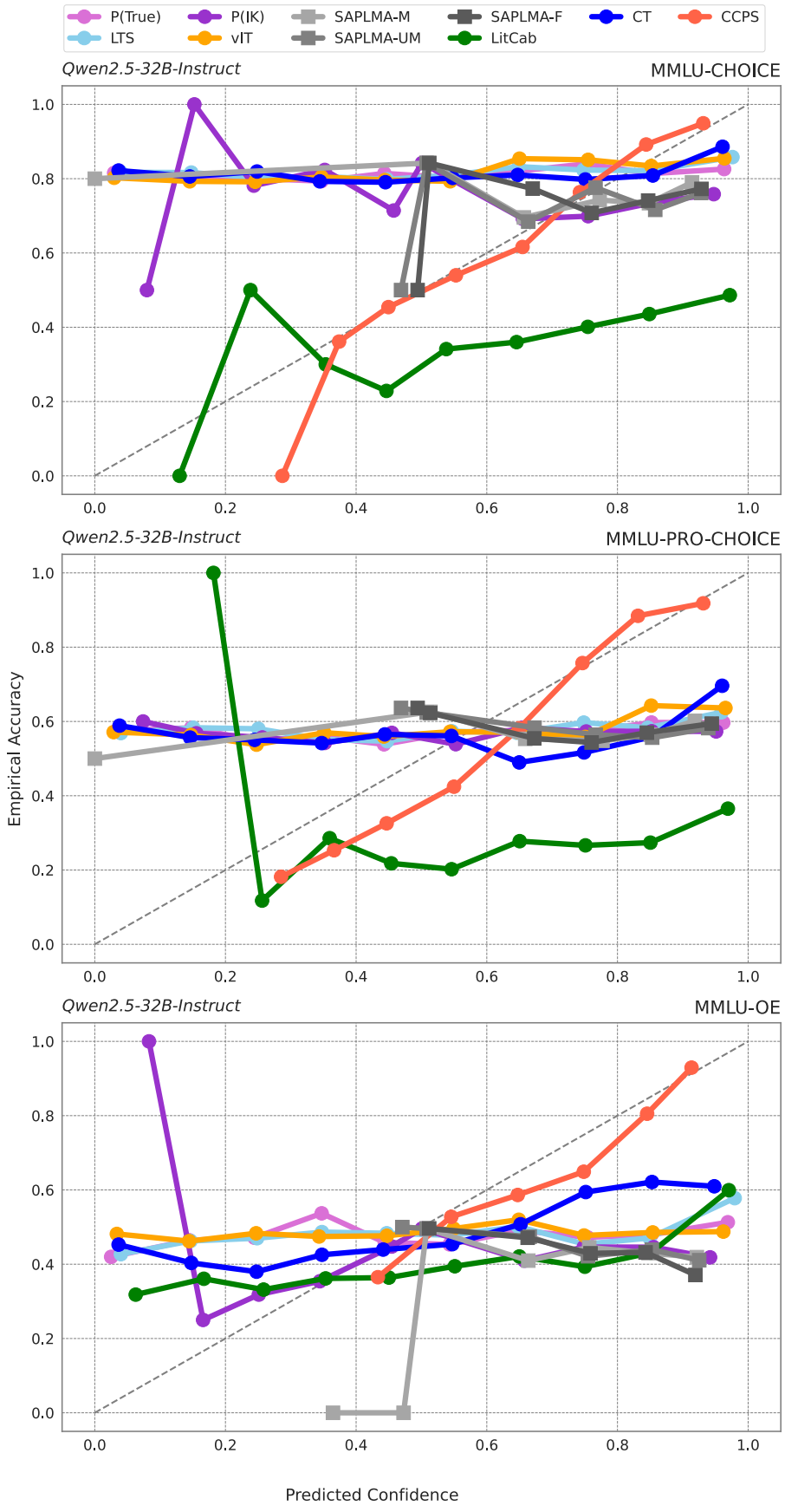

Figure 10: Calibration curves of confidence estimation methods on `Qwen2.5-32B-Instruct` across MMLU variants.

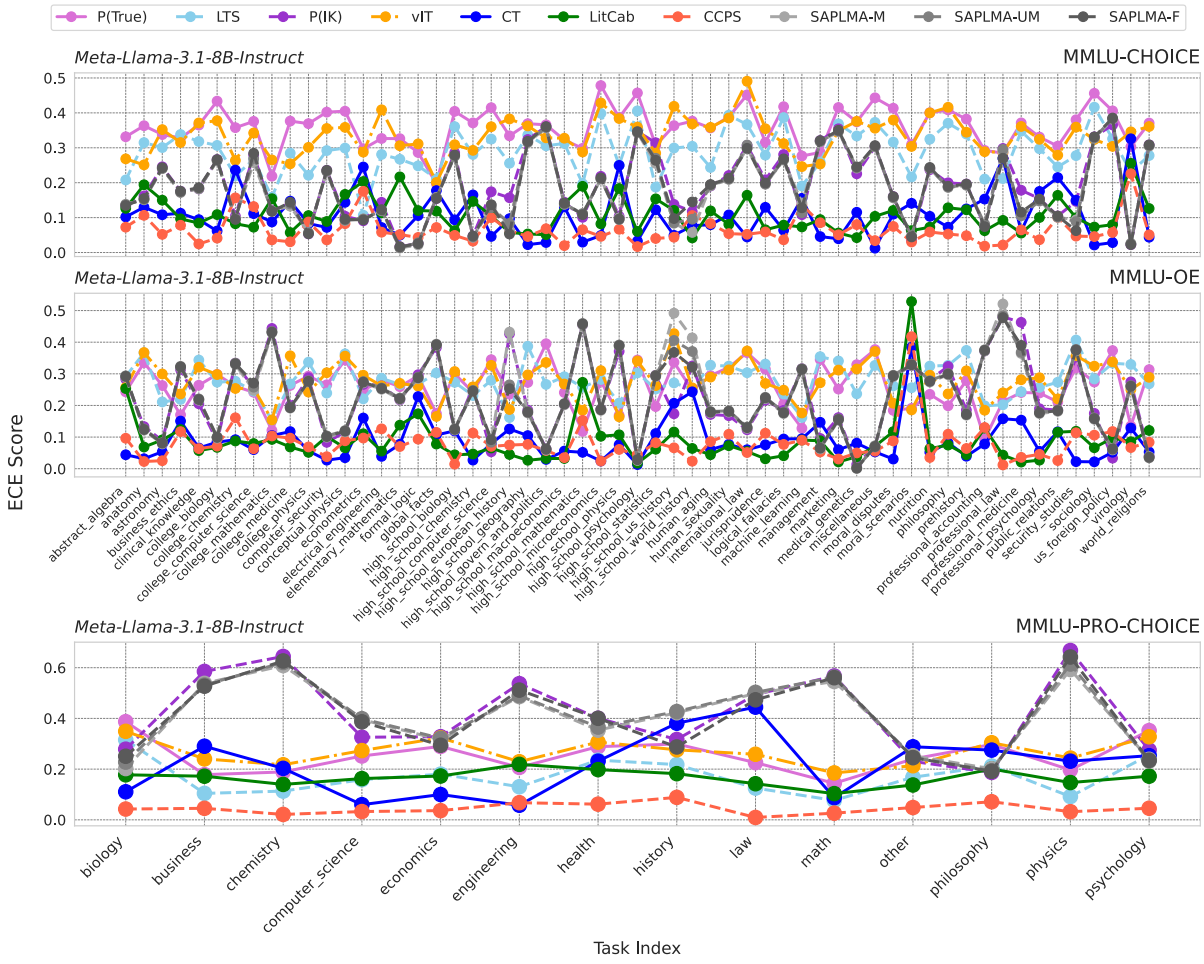

Figure 11: ECE comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across different tasks of MMLU variants.

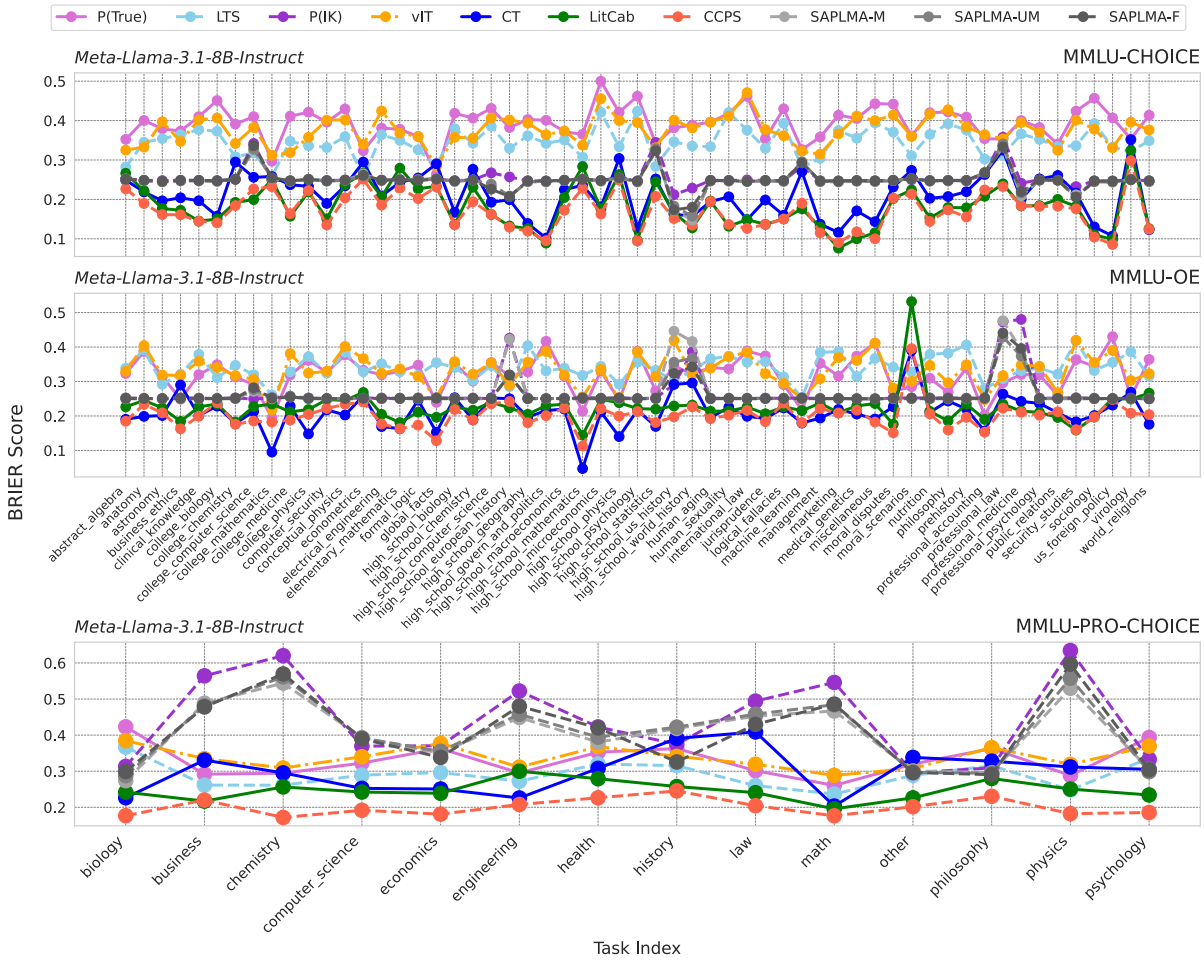

Figure 12: Brier score comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across different tasks of MMLU variants.

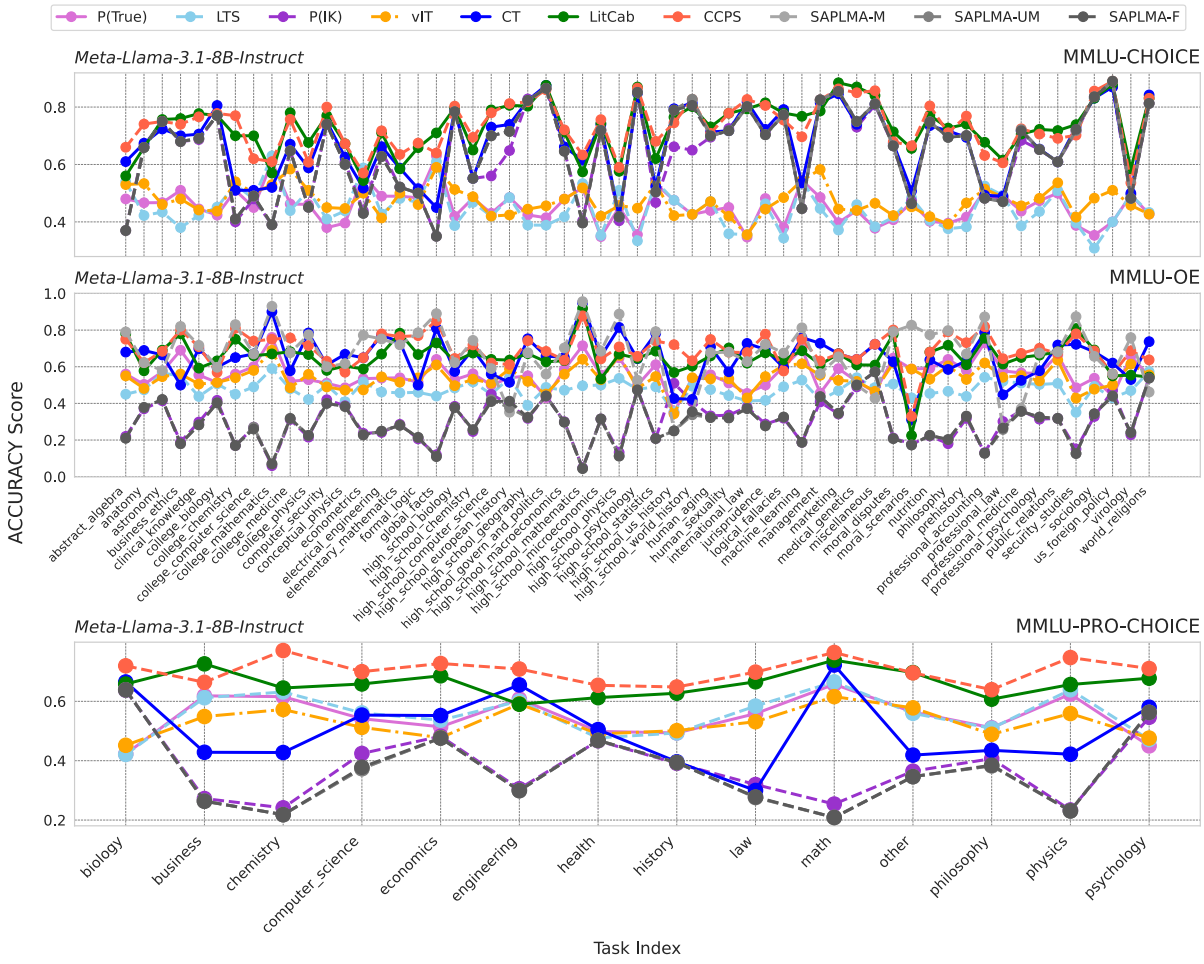

Figure 13: Accuracy (ACC) comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across different tasks of MMLU variants.

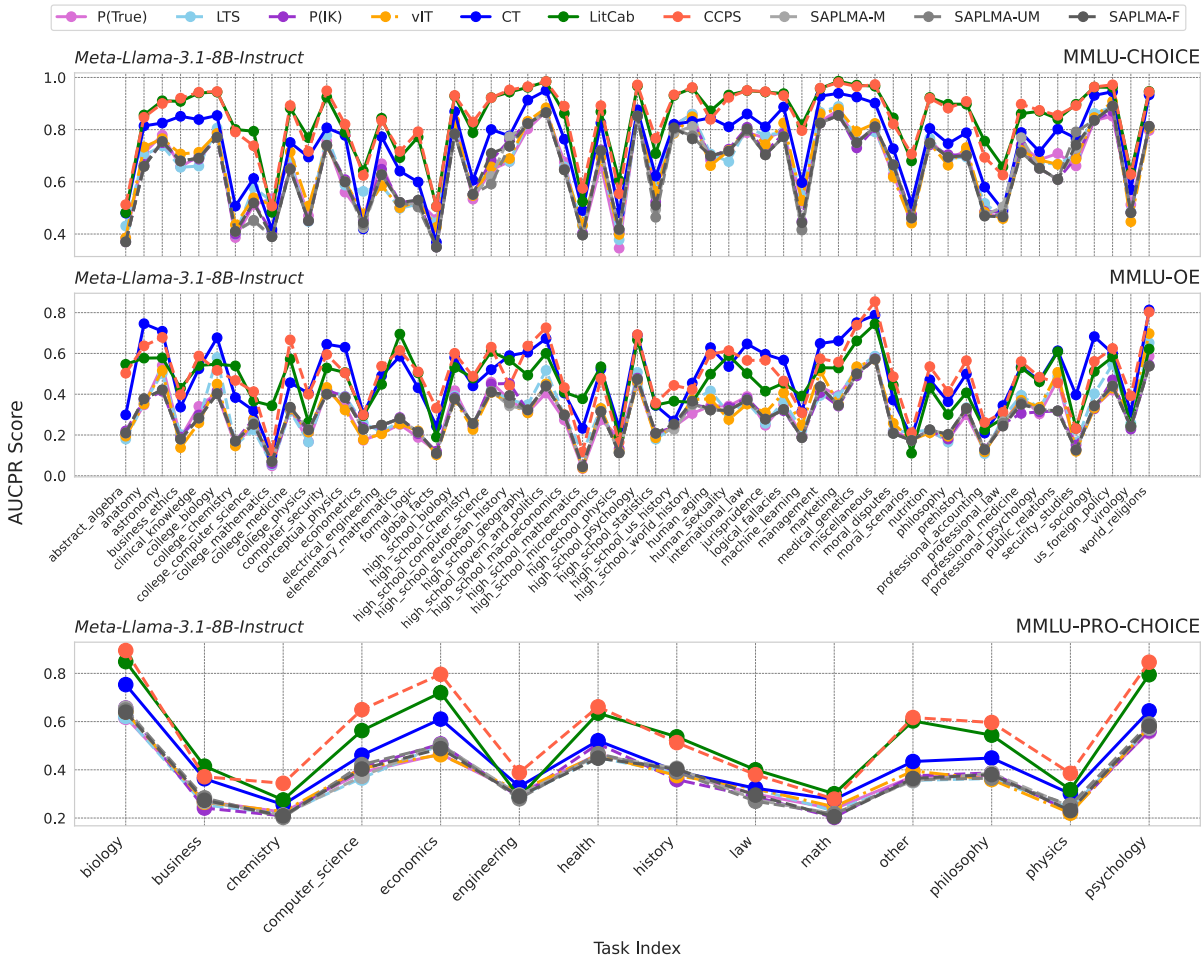

Figure 14: AUCPR comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across different tasks of MMLU variants.

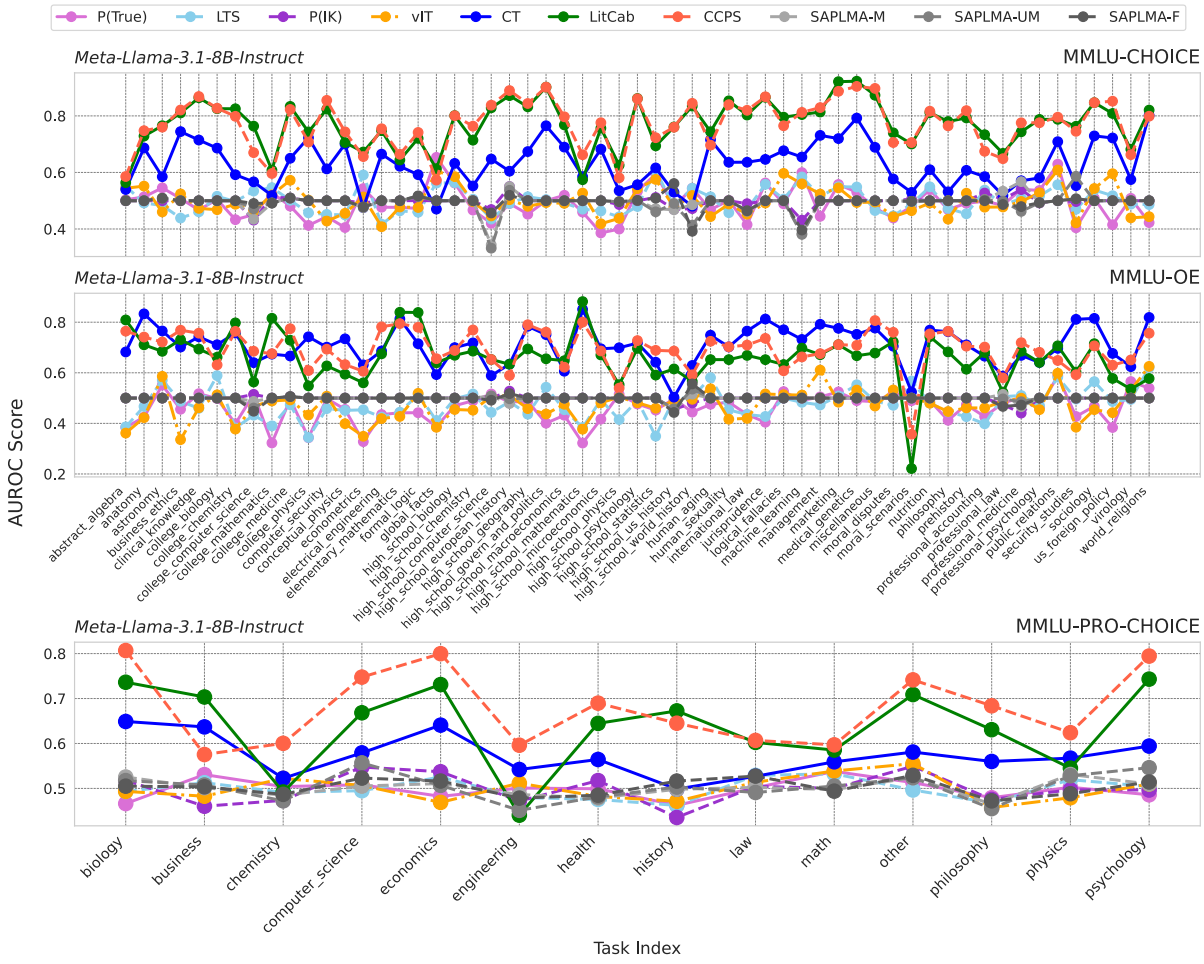

Figure 15: AUROC comparison of confidence estimation methods on `Meta-Llama-3.1-8B-Instruct` across different tasks of MMLU variants.

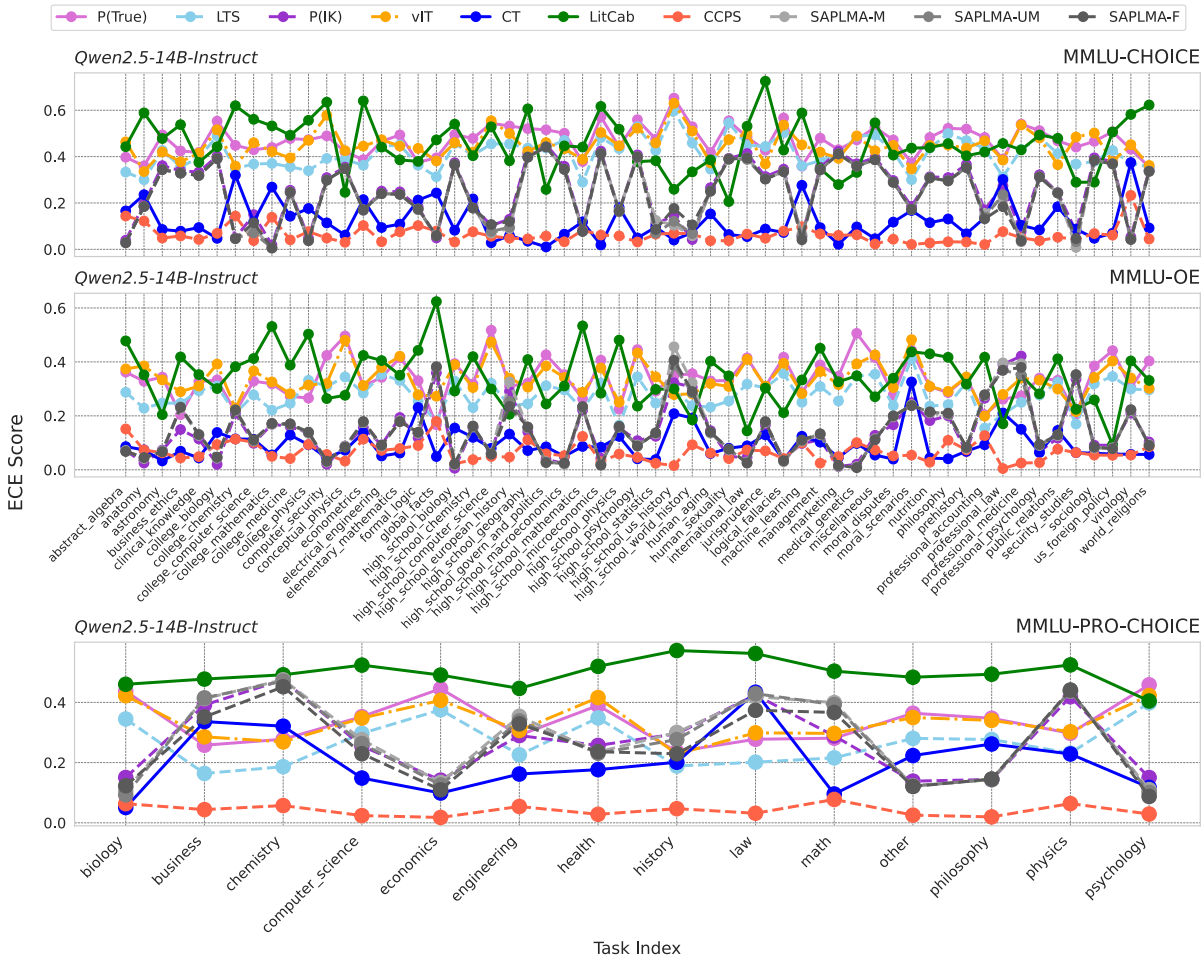

Figure 16: ECE comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across different tasks of MMLU variants.

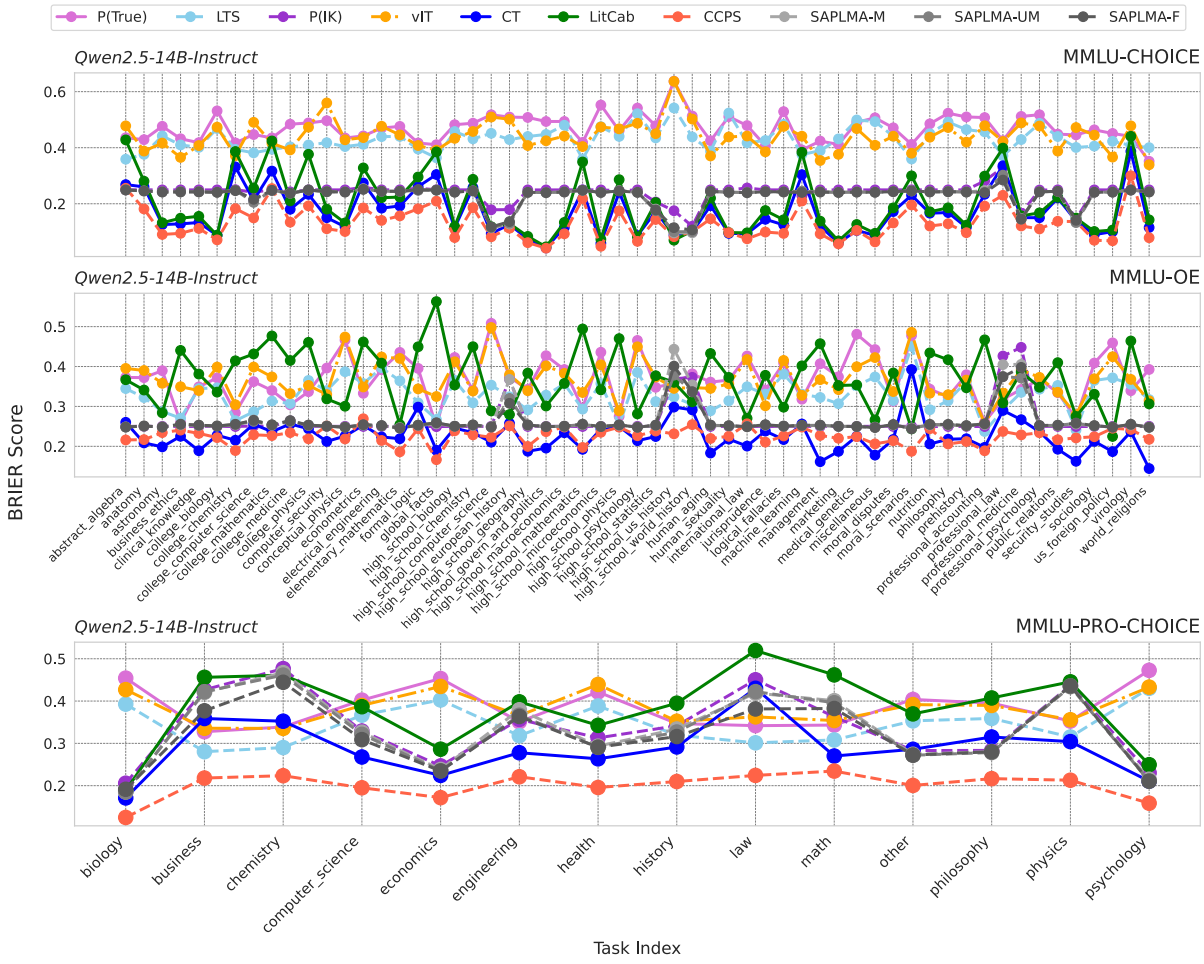

Figure 17: Brier score comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across different tasks of MMLU variants.

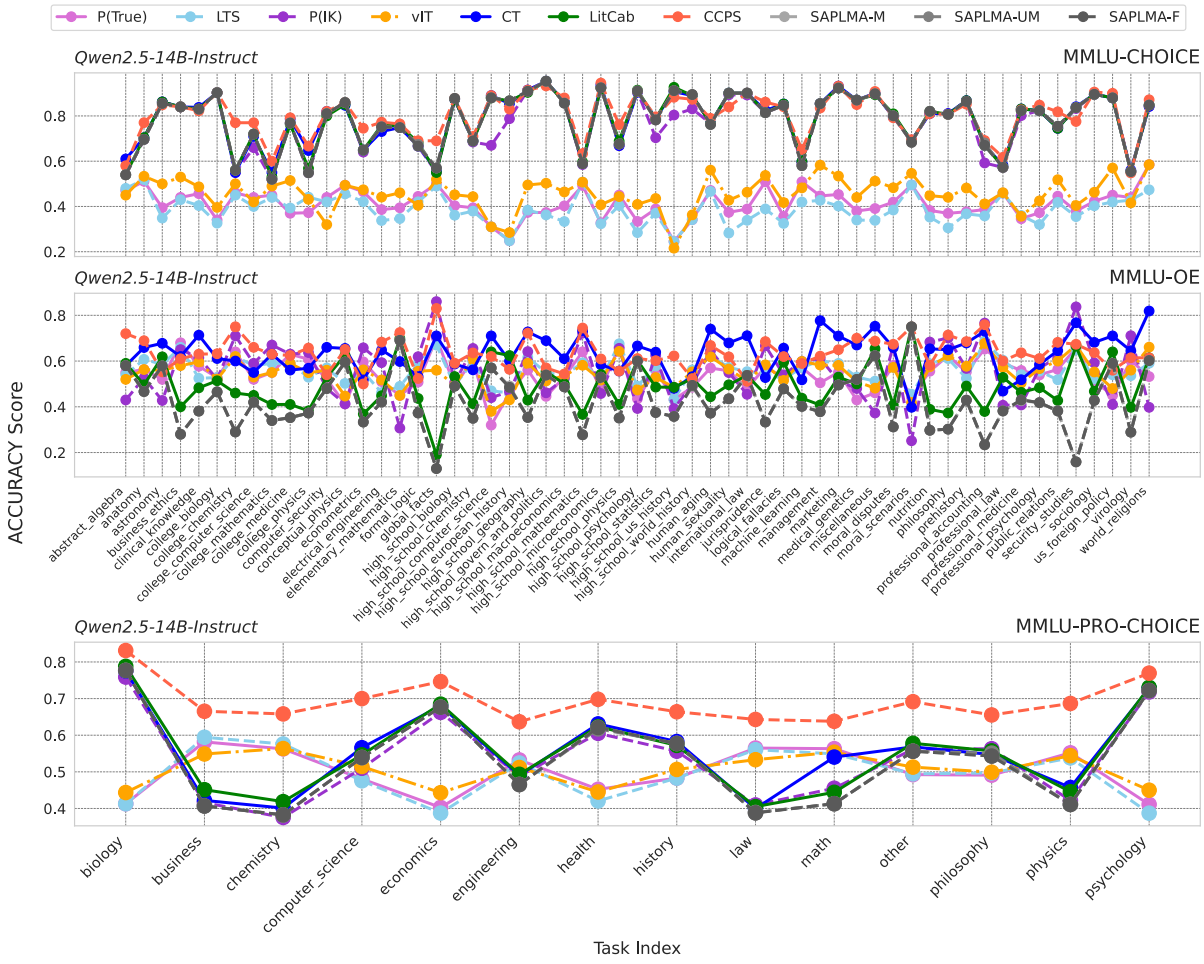

Figure 18: Accuracy (ACC) comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across different tasks of MMLU variants.

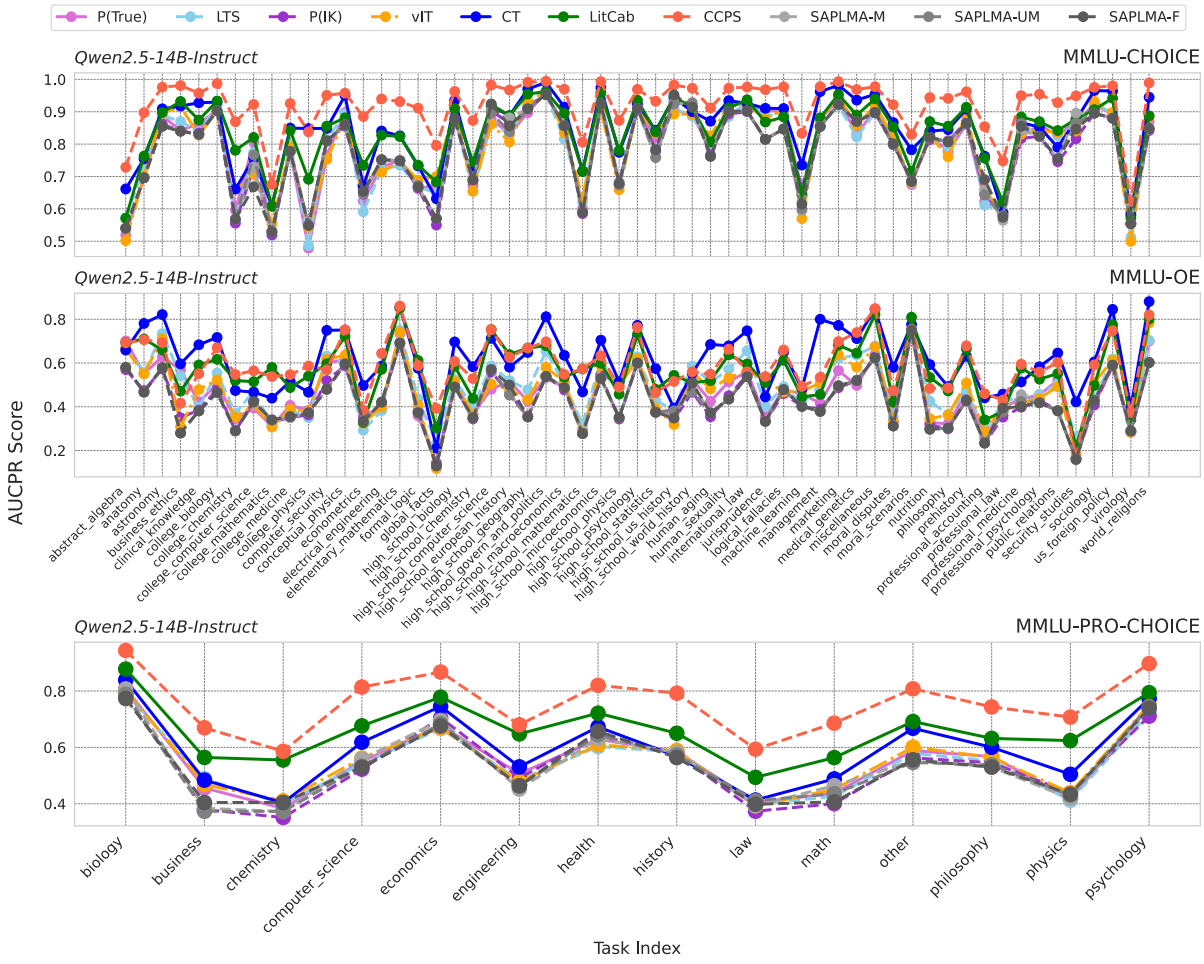

Figure 19: AUCPR comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across different tasks of MMLU variants.

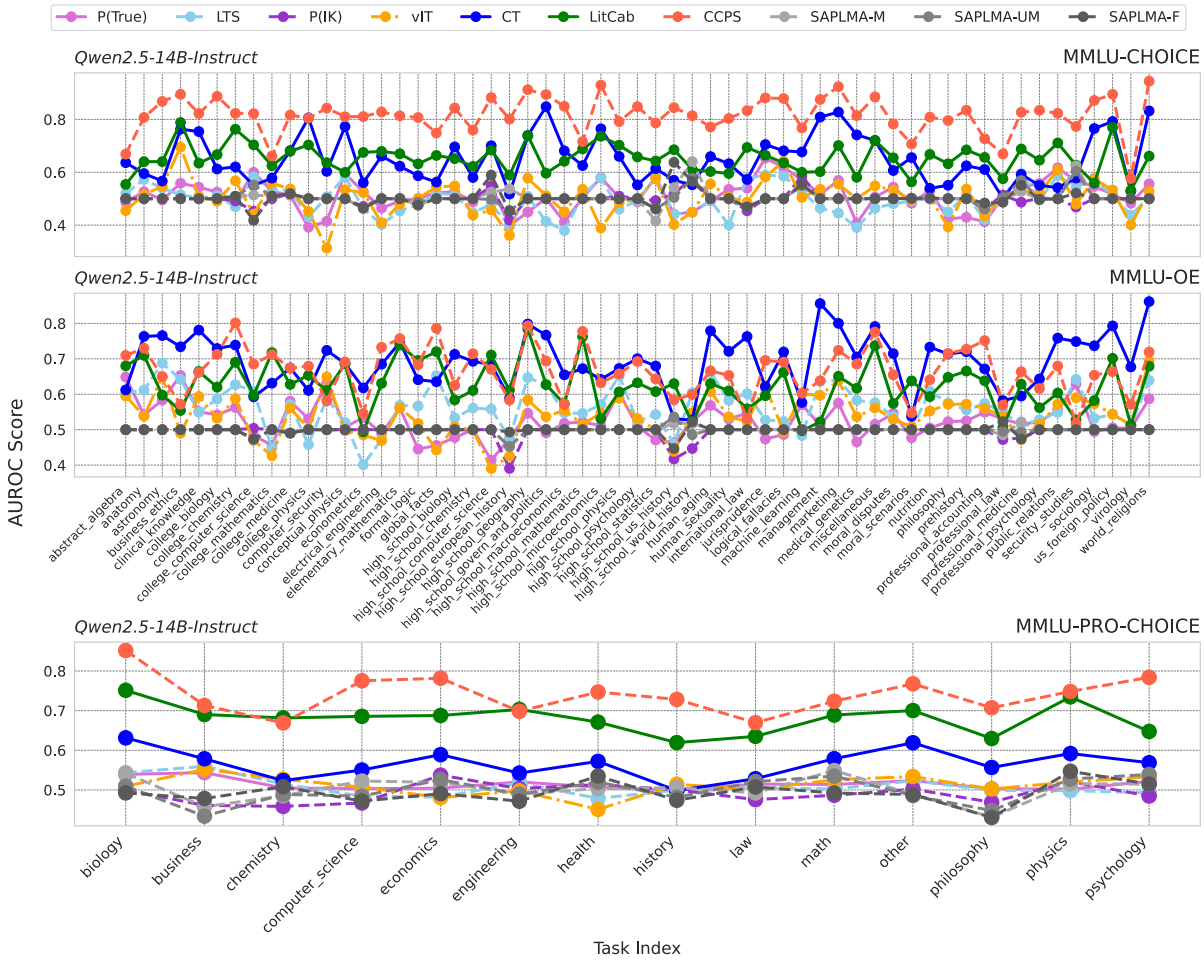

Figure 20: AUROC comparison of confidence estimation methods on `Qwen2.5-14B-Instruct` across different tasks of MMLU variants.

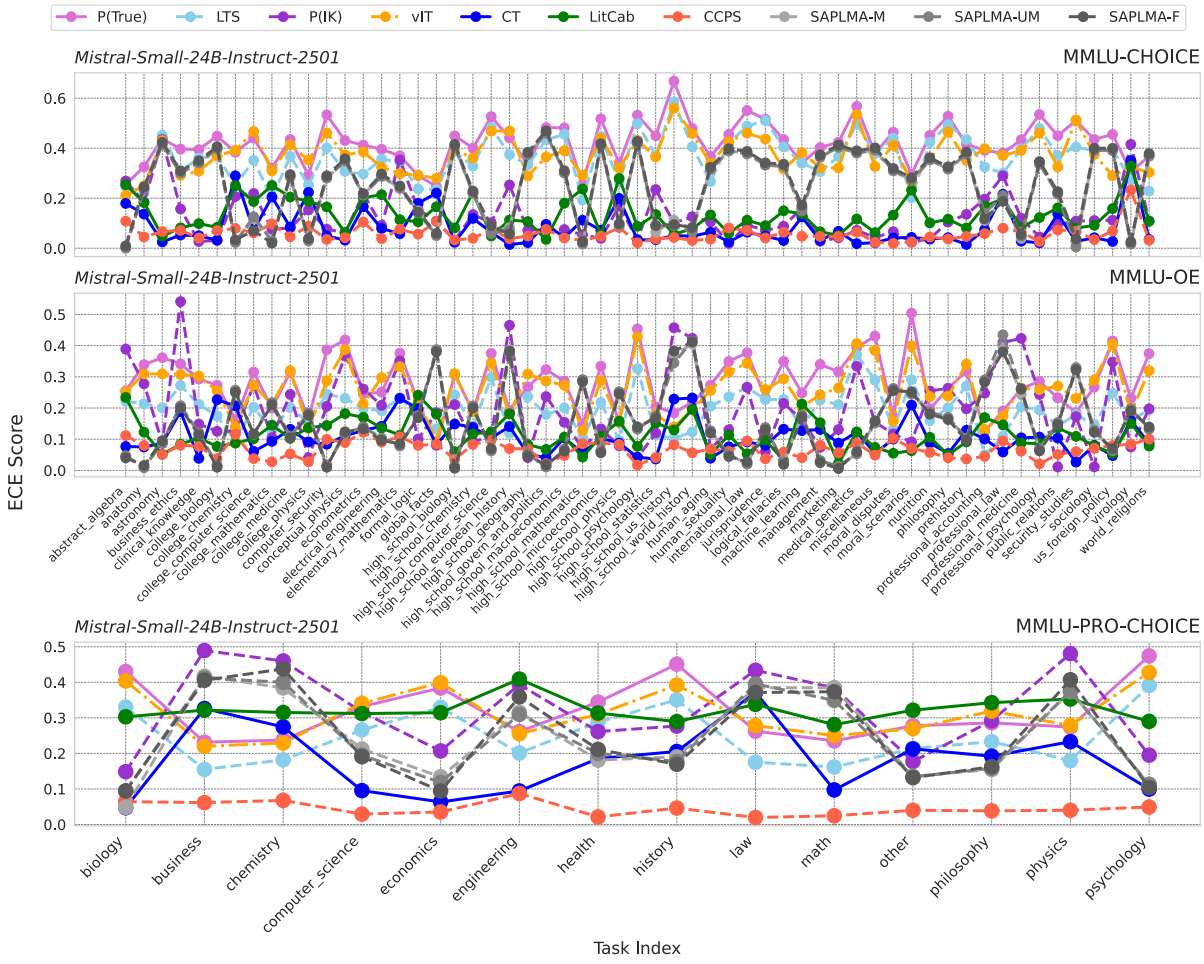

Figure 21: ECE comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across different tasks of MMLU variants.

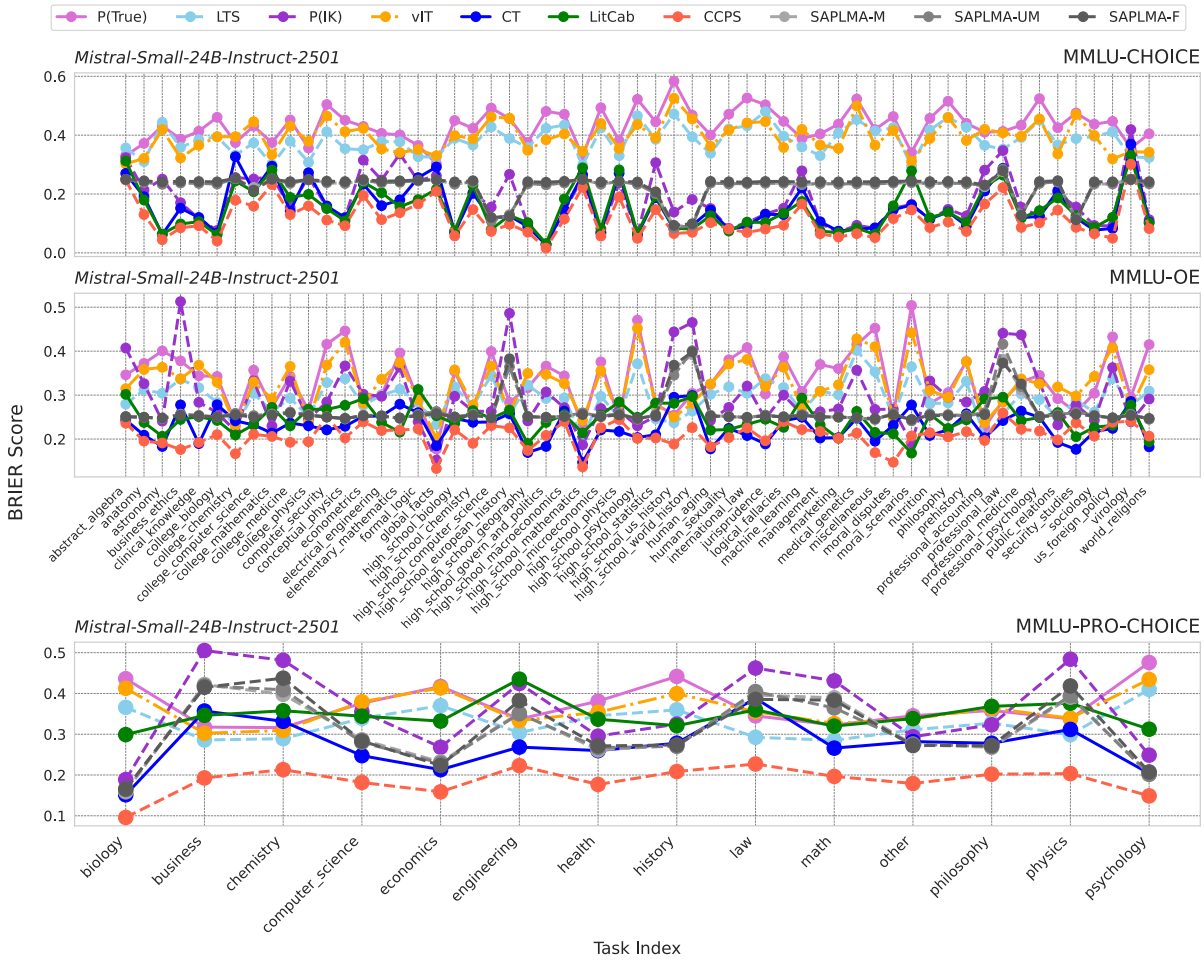

Figure 22: Brier score comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across different tasks of MMLU variants.

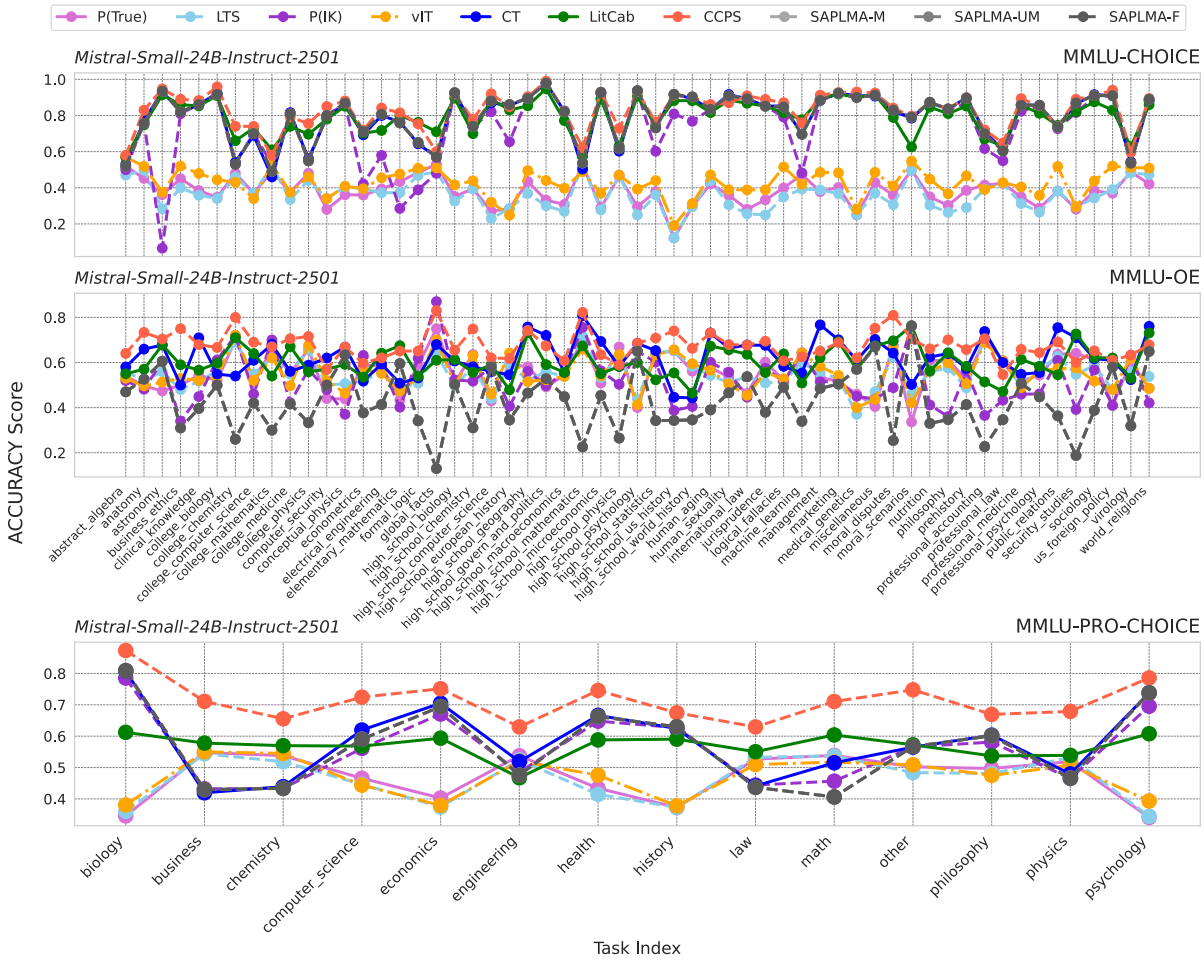

Figure 23: Accuracy (ACC) comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across different tasks of MMLU variants.

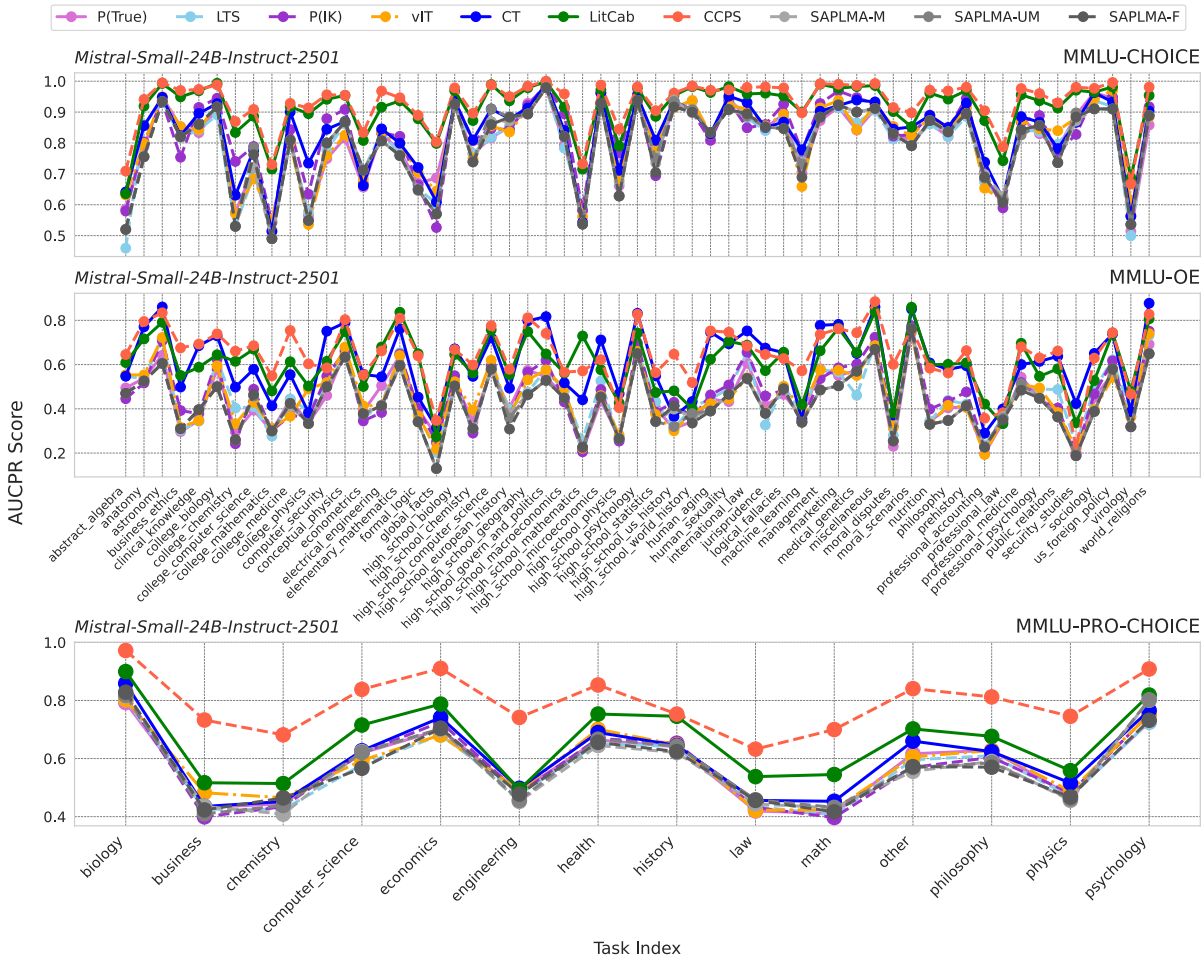

Figure 24: AUCPR comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across different tasks of MMLU variants.

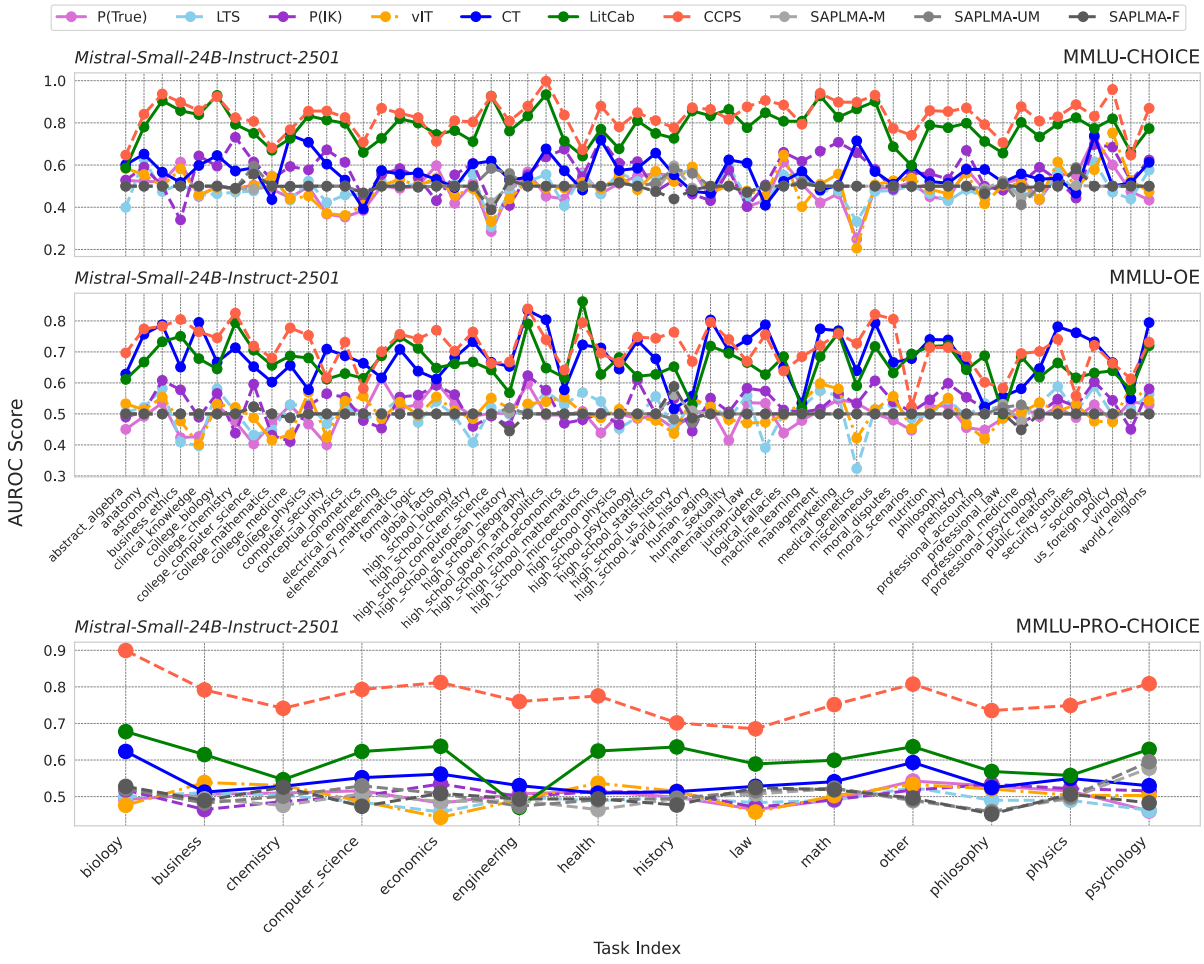

Figure 25: AUROC comparison of confidence estimation methods on `Mistral-Small-24B-Instruct-2501` across different tasks of MMLU variants.

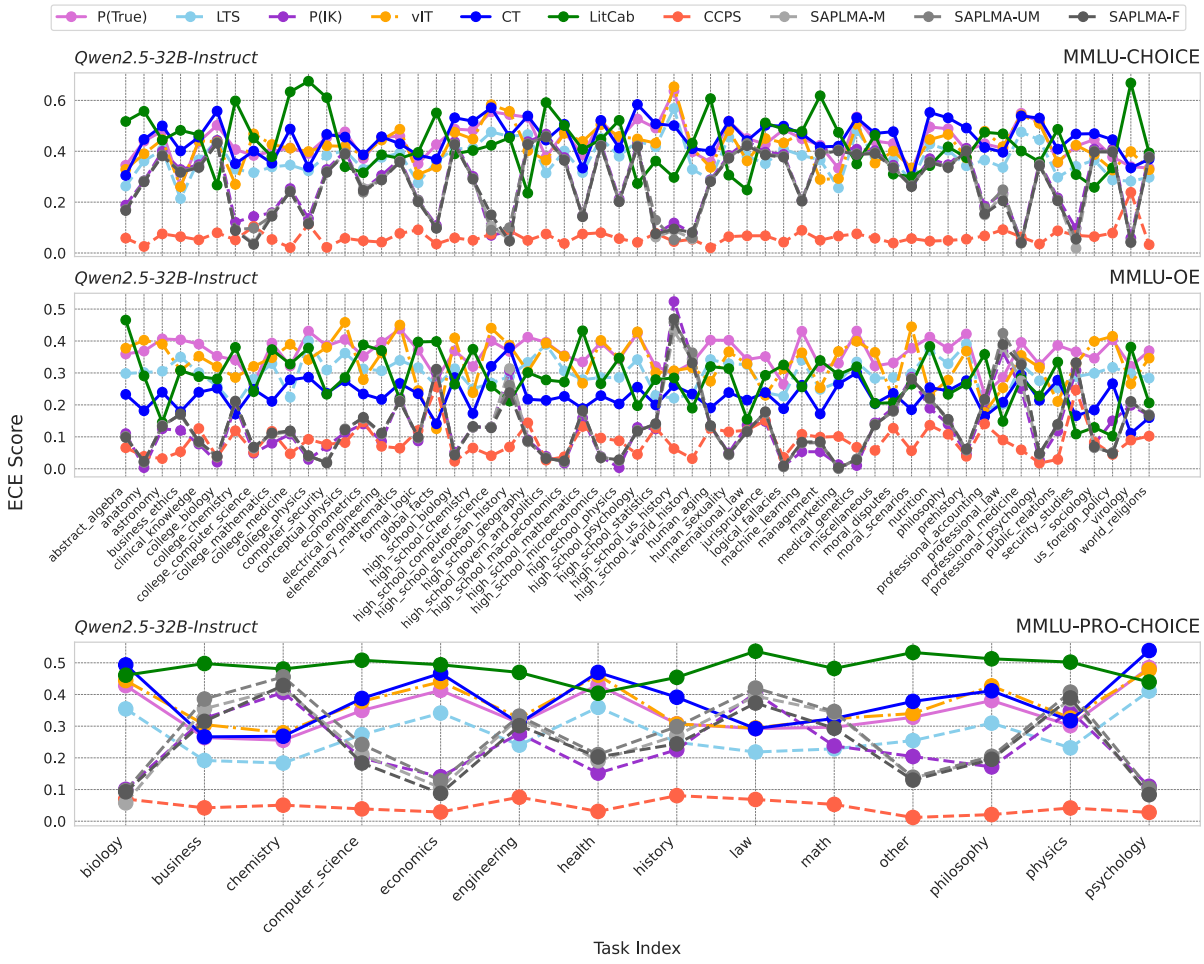

Figure 26: ECE comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across different tasks of MMLU variants.

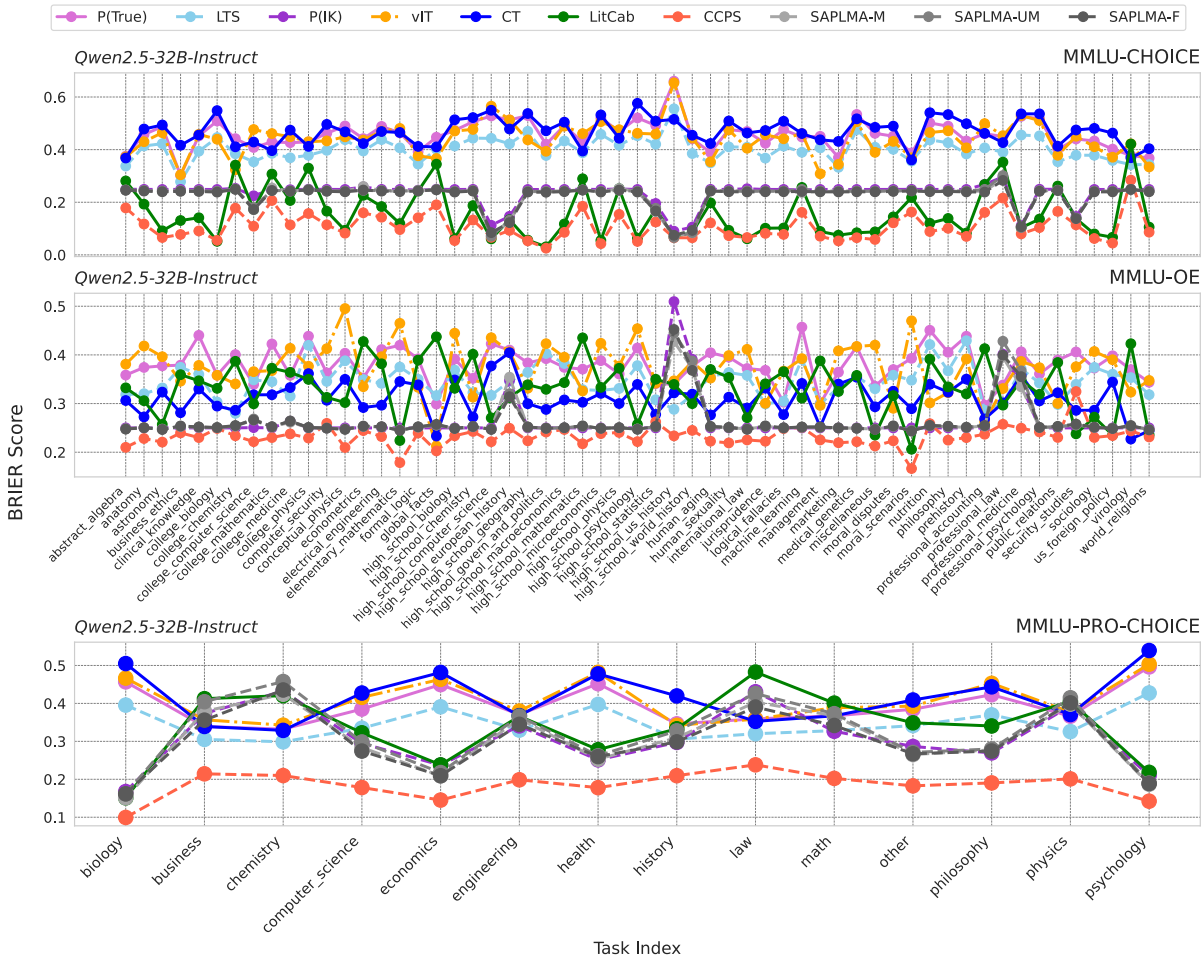

Figure 27: Brier score comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across different tasks of MMLU variants.

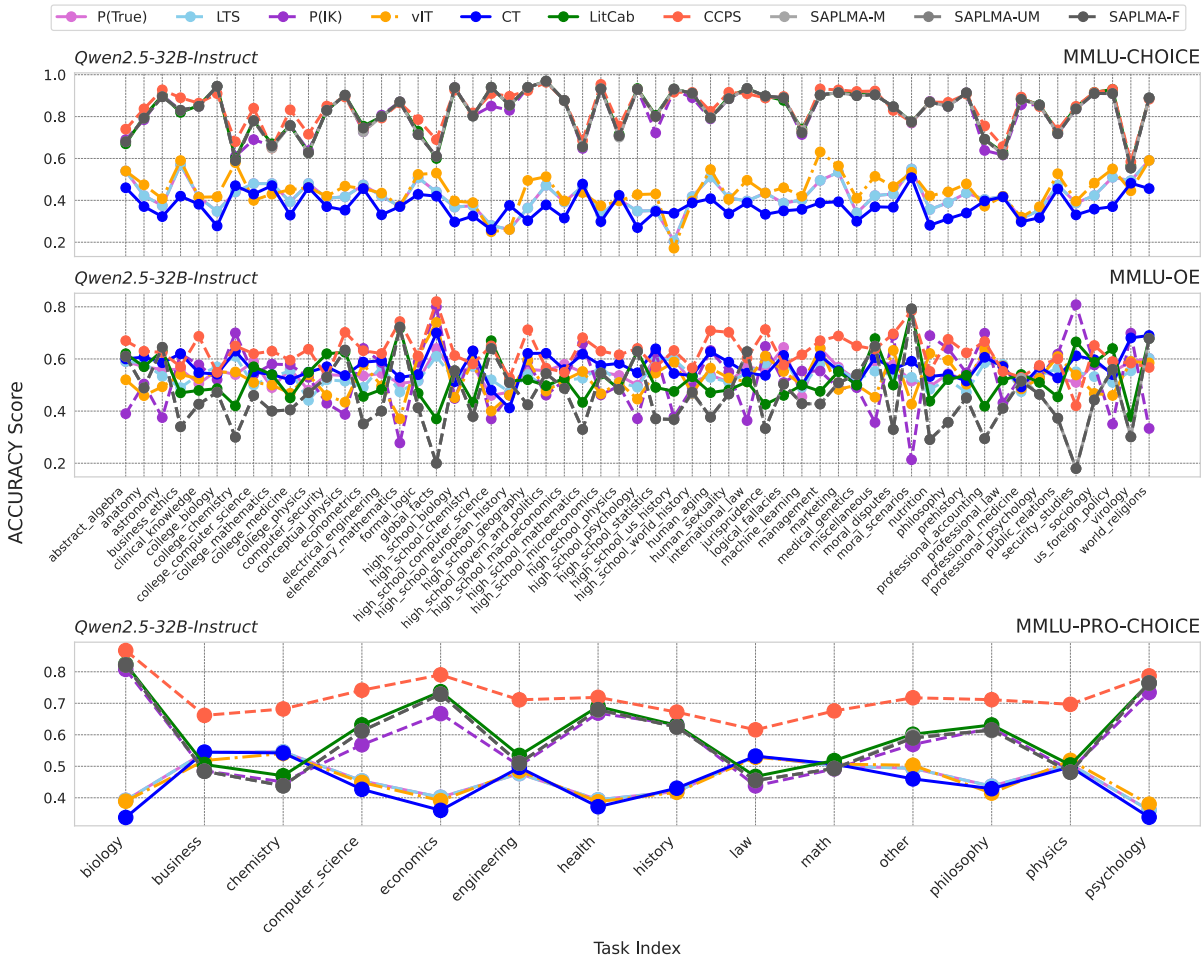

Figure 28: Accuracy (ACC) comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across different tasks of MMLU variants.

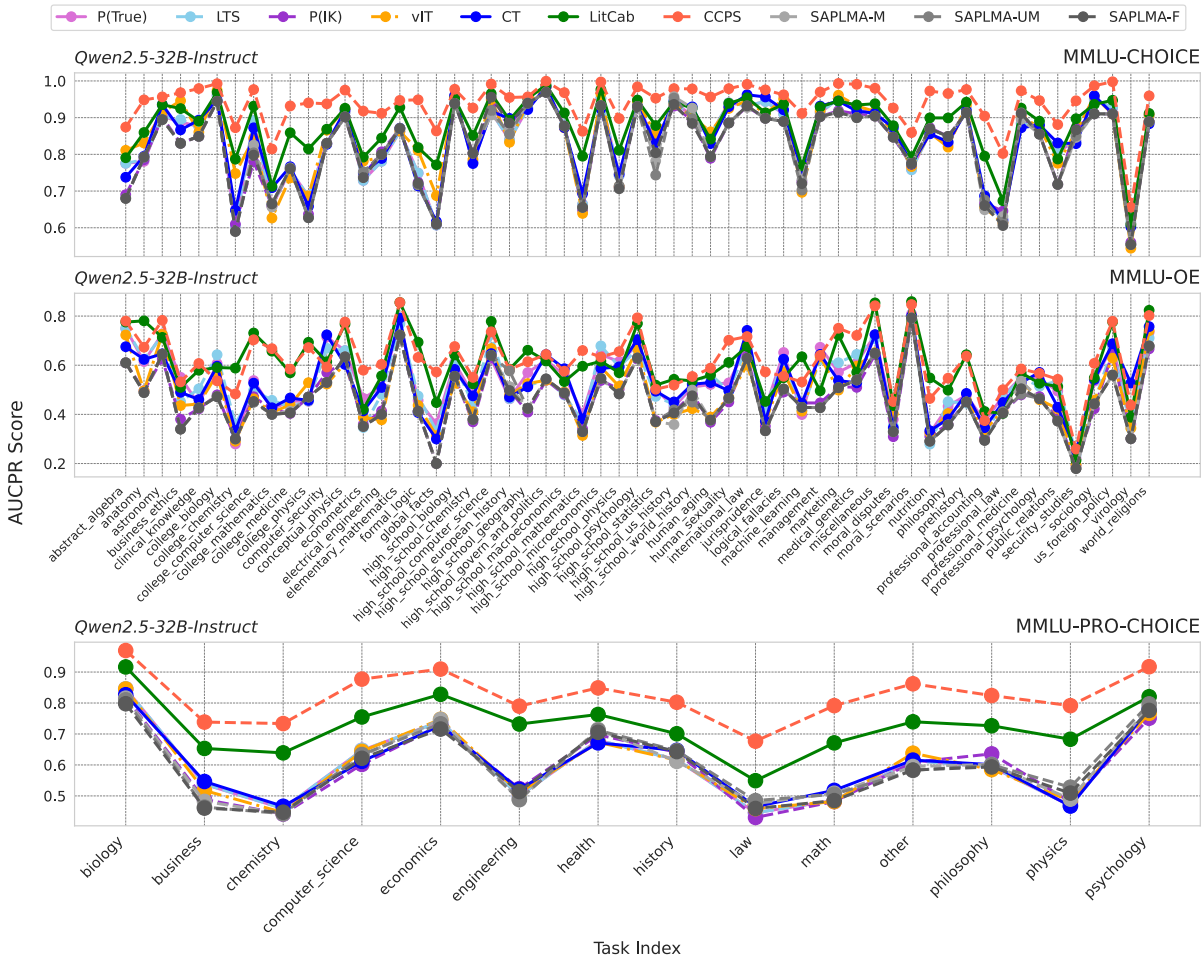

Figure 29: AUCPR comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across different tasks of MMLU variants.

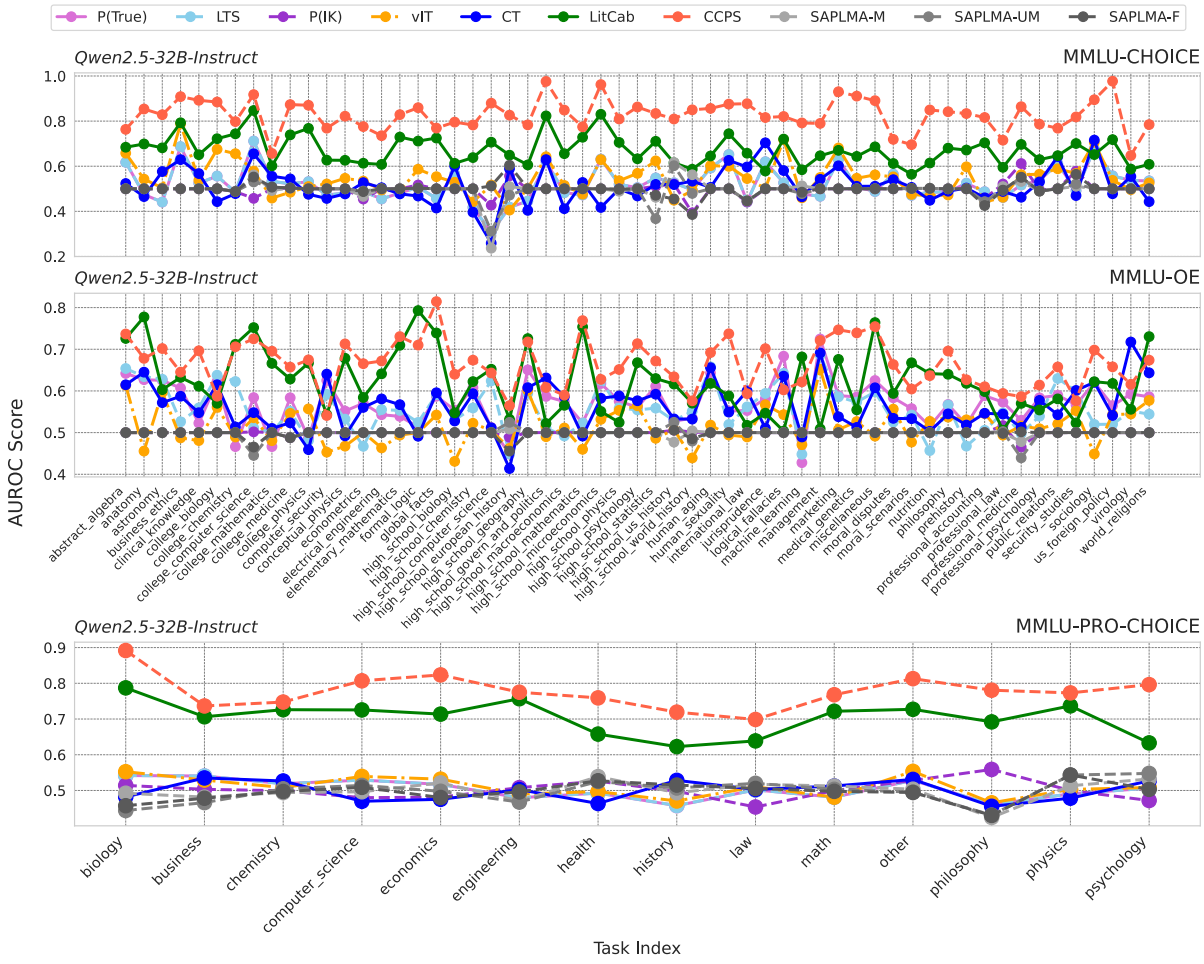

Figure 30: AUROC comparison of confidence estimation methods on `Qwen2.5-32B-Instruct` across different tasks of MMLU variants.

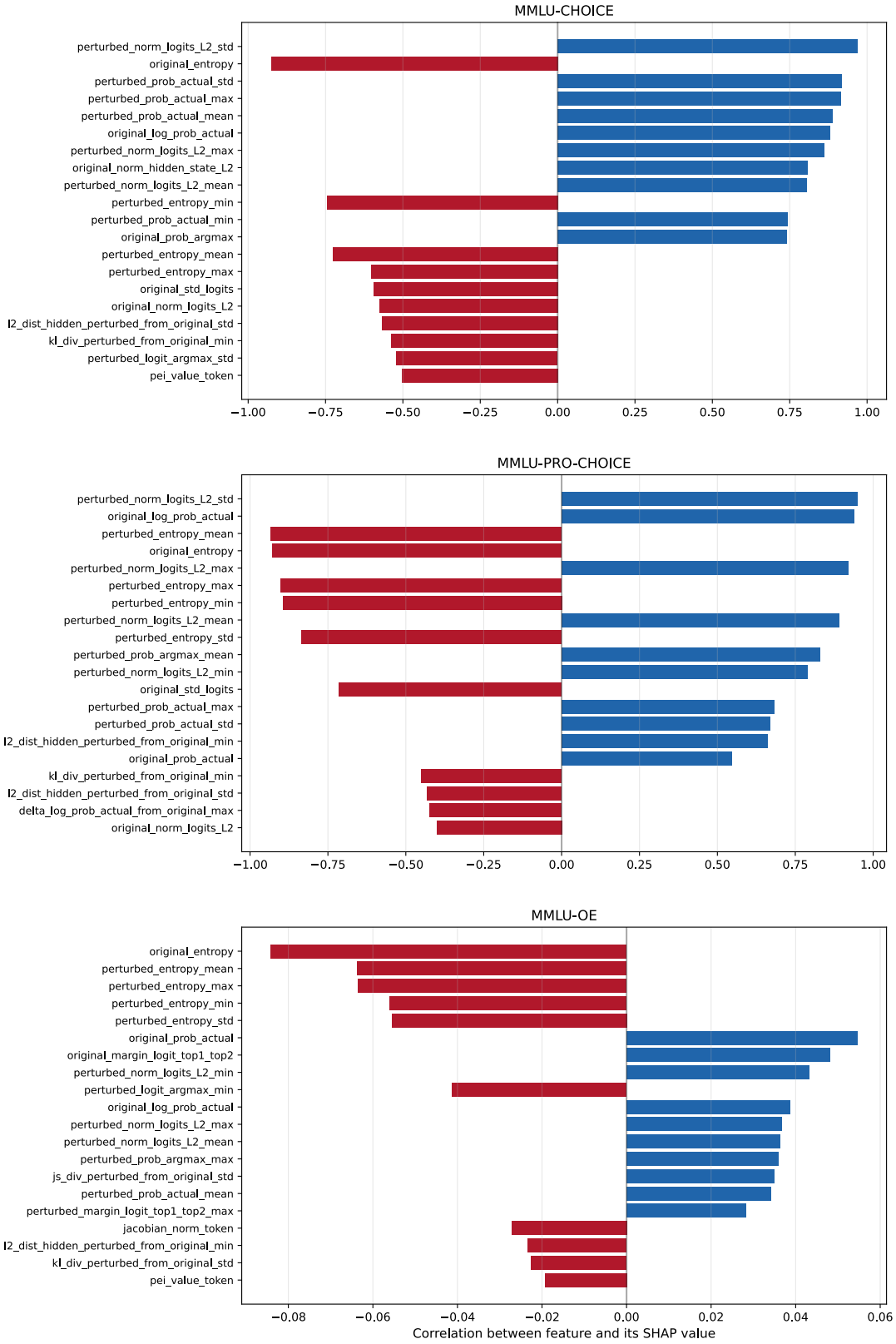

Figure 31: Correlations between feature values and SHAP scores in CCPS on `Meta-Llama-3.1-8B-Instruct` across all datasets. Blue bars denote positive correlations (higher feature values increase prediction ACC), and red bars denote negative correlations.

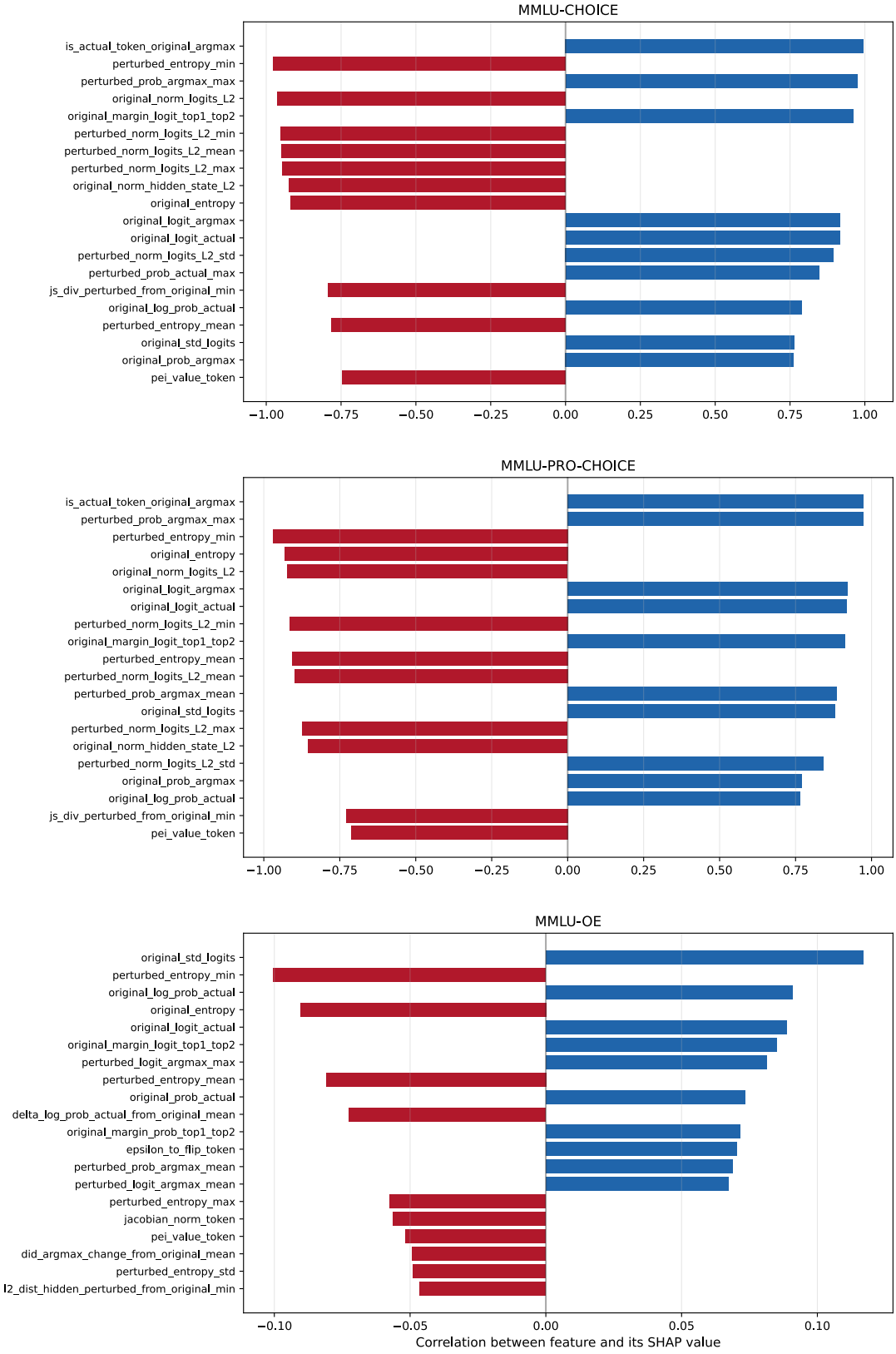

Figure 32: Correlations between feature values and SHAP scores in CCPS on `Qwen2.5-14B-Instruct` across all datasets. Blue bars denote positive correlations (higher feature values increase prediction ACC), and red bars denote negative correlations.

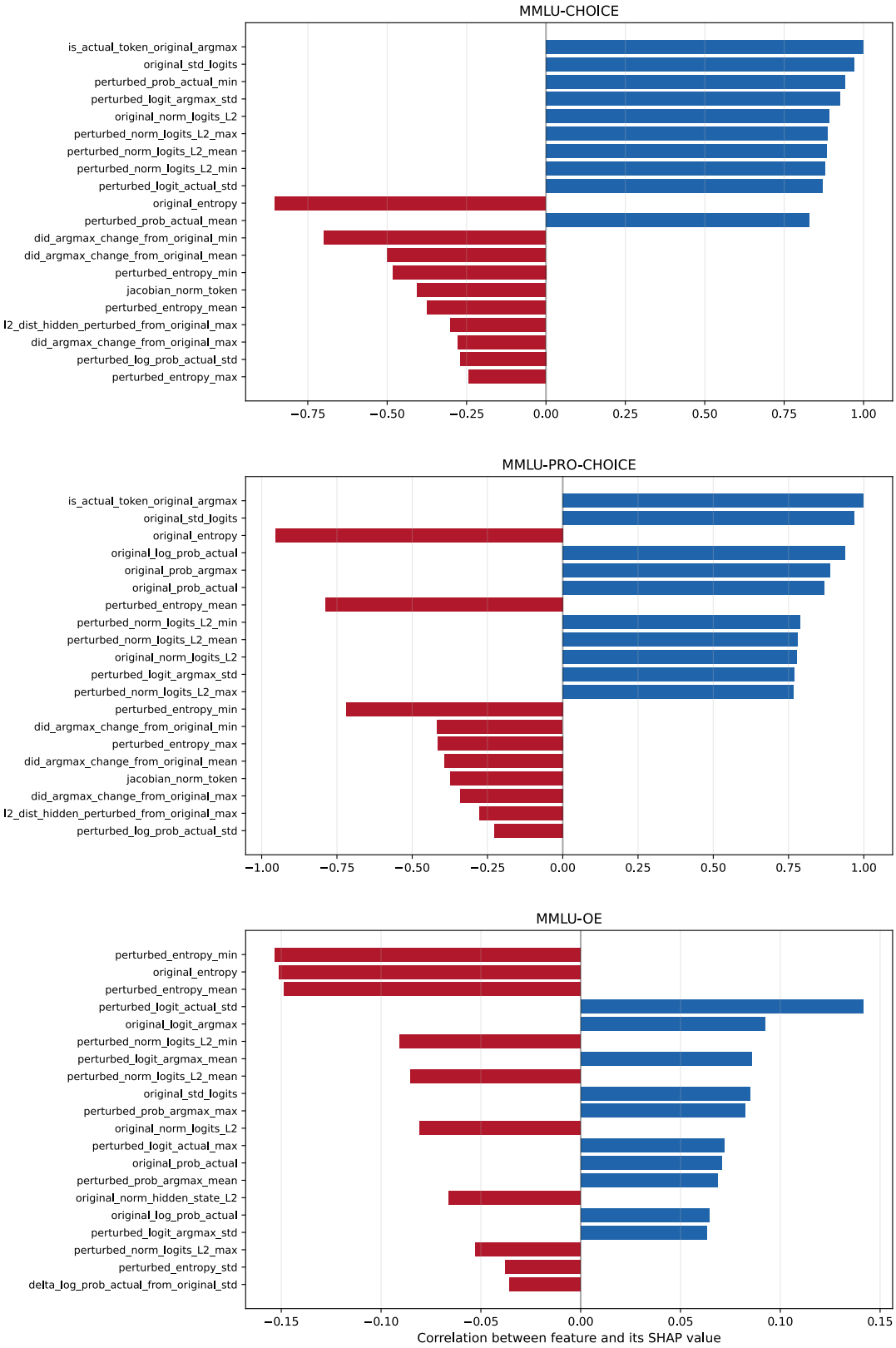

Figure 33: Correlations between feature values and SHAP scores in CCPS on `Mistral-Small-24B-Instruct-2501` across all datasets. Blue bars denote positive correlations (higher feature values increase prediction ACC), and red bars denote negative correlations.

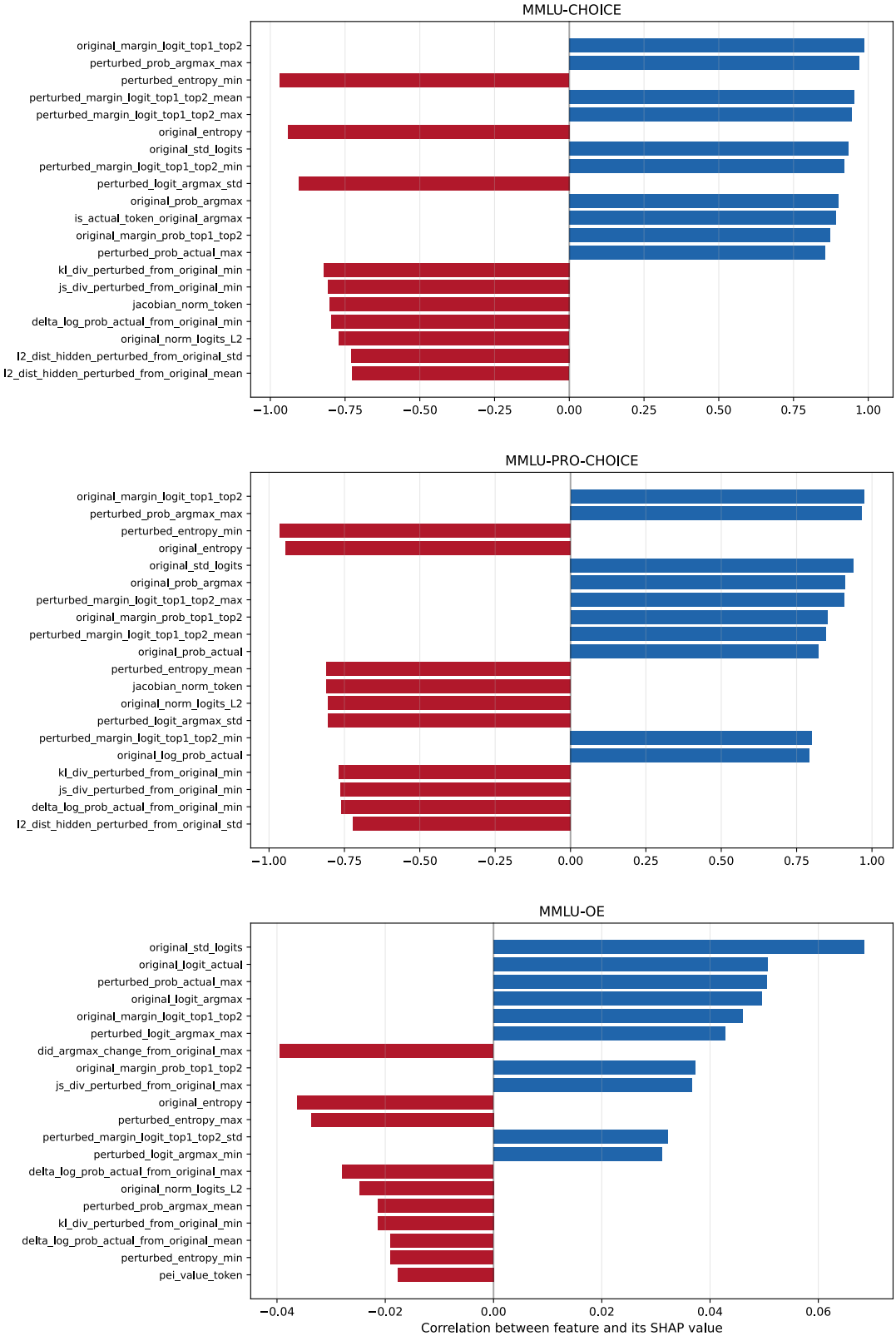

Figure 34: Correlations between feature values and SHAP scores in CCPS on `Qwen2.5-32B-Instruct` across all datasets. Blue bars denote positive correlations (higher feature values increase prediction ACC), and red bars denote negative correlations.