Do Not Design, Learn: A Trainable Scoring Function for Uncertainty Estimation in Generative LLMs

Anonymous ACL submission

Abstract

In this work, we introduce the Learnable Response Scoring Function (LARS) for Uncertainty Estimation (UE) in generative Large Language Models (LLMs). Current scoring functions for probability-based UE, such as length-normalized scoring and semantic contribution-based weighting, are designed to solve specific aspects of the problem but exhibit limitations, including the inability to handle biased probabilities and under-performance in low-resource languages like Turkish. To address these issues, we propose LARS, a scoring function that leverages supervised data to capture complex dependencies between tokens and probabilities, thereby producing more reliable and calibrated response scores in computing the uncertainty of generations. Our extensive experiments across multiple datasets show that LARS substantially outperforms existing scoring functions considering various probability-based UE methods.

1 Introduction

Recent years have seen a transformative shift in AI due to the emergence of generative Large Language Models (LLMs). Their near-human capabilities in understanding, generating, and processing information have revolutionized human-machine interactions and facilitated their integration across various industries such as healthcare, law, finance, and marketing (Ye et al., 2023; OpenAI, 2023; Touvron et al., 2023; Huang et al., 2023). Given that LLMs can sometimes generate misleading or erroneous outputs, it is crucial to evaluate how much reliance should be placed on their responses. Tools such as hallucination detection (Li et al., 2023), fact verification (Wang et al., 2024), and Uncertainty Estimation (UE) (Malinin and Gales, 2021) are essential for assessing the correctness of model responses. The field of Uncertainty Estimation, well-established in classification tasks, has recently been adapted to generative LLMs. Recent studies (Kuhn et al., 2023) demonstrate that these adaptations can effectively predict incorrect LLM outputs without the need for external feedback.

UE methods can be broadly categorized into two approaches. Probability-based methods (Malinin and Gales, 2021; Kuhn et al., 2023) utilize token probabilities externally to predict uncertainty. In contrast, non-probability-based methods (Lyu et al., 2024; Chen et al., 2024) employ heuristics that do not rely on token probabilities. This work focuses exclusively on probability-based methods, with a discussion of related works presented in Section 2.

A fundamental challenge in UE of LLMs with probability-based methods is the necessity to aggregate multiple token probabilities into a single score. To this end, existing methods typically employ a scoring function. A common scoring function is Length-Normalized Scoring (LNS), which calculates the mean of log probabilities, as employed by (Malinin and Gales, 2021; Kuhn et al., 2023), to mitigate bias in longer generations. Subsequent approaches by (Bakman et al., 2024; Duan et al., 2024) introduce heuristics that prioritize semantically important tokens by assigning higher weights to them, rather than simply averaging as in LNS. However, these scoring functions, largely heuristic in design, often overlook potential pitfalls. In this work, we critically analyze the weaknesses of the existing scoring functions and introduce *Learnable Response Scoring Function (LARS),* which learns a scoring function from supervised data.

We summarize our main contributions as follows: **1.** We experimentally demonstrate the limitations of existing scoring functions in terms of their calibration and performance in low-resource languages. **2.** We introduce a novel off-the-shelf scoring function, LARS, which is learned directly from supervised data. **3.** We validate the superiority of LARS over existing baselines across three different datasets and provide an analysis of its components to rationalize the effectiveness of LARS.

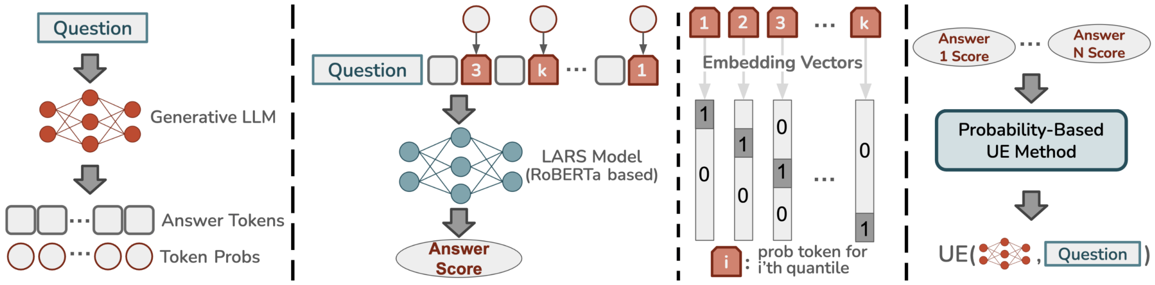

Figure 1: (Left) Answer generation process using a generative LLM. (Mid Left) LARS overview. It utilizes the question, answer tokens, and token probabilities. Token probabilities are fed to LARS model as special probability tokens. (Mid Right) Illustration of few-hot represented embedding vectors of probability tokens. (Right) Summary of probability-based UE methods where they take different sampled answer scores and output a single UE value.

2 Background

Uncertainty Estimation (UE). Uncertainty Estimation (or Quantification) in generative Large Language Models (LLMs) addresses the challenge of predicting a model's uncertainty regarding a given sequence or question. An effective UE method assigns a lower score (indicating less uncertainty) to questions where the model is likely to provide the correct answer, and a higher score otherwise. Mathematically, we have $UE(\theta, x_1) < UE(\theta, x_2)$ if the most probable generation of model θ for question x_1 is more likely to be correct than for question x_2. Previous works formulate this approach for closed-ended questions with well-defined ground truths (Malinin and Gales, 2021; Kuhn et al., 2023; Bakman et al., 2024; Duan et al., 2024).

Related Works. UE has recently become a topic of significant interest, leading to the proposal of various methods. These methods can be broadly categorized into four types: 1. Self-checking methods: The model evaluates its own generation correctness using different strategies (Kadavath et al., 2022; Manakul et al., 2023; Li et al., 2024; Luo et al., 2023; Zhao et al., 2023). 2. Output consistency methods: Uncertainty is predicted by examining the consistency of various outputs for a given question (Lyu et al., 2024; Lin et al., 2023; Zhang et al., 2024; Ulmer et al., 2024; Elaraby et al., 2023). 3. Internal state examination methods: The activations of the model are analyzed to predict the model errors (Chen et al., 2024). 4. Token probability-based methods: These methods utilize token probabilities to estimate uncertainty (Malinin and Gales, 2021; Kuhn et al., 2023; Bakman et al., 2024; Duan et al., 2024). These methods can be used in conformal prediction frameworks, which offer theoretical guarantees for model correctness (Deutschmann et al., 2024; Quach et al., 2023; Yadkori et al., 2024). In this work, we focus on improving token probability-based methods by proposing a learnable scoring function.

Token Probability-based Methods. (Malinin and Gales, 2021) formally proposes using sequence probability as the generation's probability for a given question $\mathbf{x}$ and a model parameterized by θ. This is mathematically defined as follows:

$$P(\mathbf{s}|\mathbf{x}, \theta) = \prod_{l=1}^{L} P(s_l | s_{<l}, \mathbf{x}; \theta), \qquad (1)$$

where $P(\mathbf{s}|\mathbf{x}, \theta)$ is the sequence probability for the generated sequence $\mathbf{s}$, and $s_{<l} \triangleq \{s_1, s_2, \ldots, s_{l-1}\}$ represents the tokens generated before s_l. This sequence probability is used in entropy calculation $\mathcal{H}(\mathbf{x}, \theta)$ by making a Monte Carlo approximation, which requires multiple answer sampling for the given question:

$$\mathcal{H}(\mathbf{x}, \theta) \approx -\frac{1}{B} \sum_{b=1}^{B} \ln P(\mathbf{s}_b | \mathbf{x}, \theta), \qquad (2)$$

where $\mathbf{s}_b$ is a sampled generation to the question $\mathbf{x}$. Later (Kuhn et al., 2023) improves the entropy by utilizing the semantic meaning of the sampled generations. They cluster the generations with the same meaning and calculate entropy using the generation probabilities associated with each cluster:

$$SE(\mathbf{x}, \theta) = -\frac{1}{|C|} \sum_{i=1}^{|C|} \ln P(\mathrm{c}_i | \mathbf{x}, \theta), \qquad (3)$$

where c_i refers to each semantic cluster and C is the set of all clusters. Notably, (Aichberger et al., 2024) enhances semantic entropy by enabling the model to generate semantically more diverse outputs.

Both (Malinin and Gales, 2021) and (Kuhn et al., 2023) observe that sequence probability in (1) is biased against longer generations. To address this, they use length-normalized scoring as follows:

$$\tilde{P}(\mathbf{s}|\mathbf{x},\theta) = \prod_{l=1}^{L} P(s_l|s_{<l},\mathbf{x};\theta)^{\frac{1}{L}}, \qquad (4)$$

where L is the sequence length. Later (Bakman et al., 2024) and (Duan et al., 2024) improve this scoring function by incorporating the meaning contribution of the tokens. Their scoring functions, MARS and TokenSAR, respectively, adopt different approaches in integrating token meaning but can be generalized with the following formulation:

$$\bar{P}(\mathbf{s}|\mathbf{x},\theta) = \prod_{l=1}^{L} P(s_l|s_{<l},\mathbf{x};\theta)^{w(\mathbf{s},\mathbf{x},L,l)}, \qquad (5)$$

where $w(\mathbf{s},\mathbf{x},L,l)$ is the weight of the lth token assigned by MARS or TokenSAR. These scoring functions aim to give more weight to tokens that directly answer the question and are calibrated such that if a generation is likely to be incorrect, they yield a lower score, and vice versa. Our goal in this work is to enhance this calibration by learning the scoring function directly from the data.

3 Shortcomings of Existing Scoring Functions

In this section, we critically and empirically analyze the shortcomings of existing scoring functions, namely Length-Normalized Scoring (LNS), MARS, and TokenSAR.

Manually Crafted Design Choices. Existing scoring functions are designed to address particular challenges within the UE problem domain. For instance, LNS mitigates length bias, whereas MARS and TokenSAR focus on reducing the impact of non-essential token probabilities. However, the complexities of designing an optimal scoring function may not be immediately evident. Typically, scoring functions involve a dot product of log probabilities and assigned weights, but alternative formulations could provide more finely calibrated estimations. Additionally, these existing functions may not adequately capture complex dependencies between tokens, such as grammatical and semantic interactions (De Marneffe and Nivre, 2019). While MARS attempts to address this by weighting phrases rather than individual tokens, it only partially solves the problem and fails to capture deeper dependencies. Lastly, both MARS and TokenSAR apply normalization on their weights $w(\mathbf{s},\mathbf{x},L,l)$, through methods like sum-normalization (TokenSar) or softmax (MARS). These design choices directly impact the model's output, potentially making the model converge to sub-optimal points.

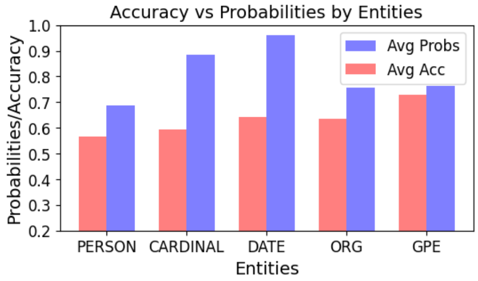

Figure 2: Average accuracy and probability assignments of LLama2-7b model to specific entities in TriviaQA.

Biased Probabilities. Existing scoring functions often directly utilize token probabilities, which can exhibit biases against specific types of entities. To explore this issue, we conducted an experiment with Llama2-7b (Touvron et al., 2023) using the TriviaQA dataset (Joshi et al., 2017). We posed questions from TriviaQA to the model and analyzed the probabilities assigned to tokens in the answer representing different entity types such as person names, organizations, and dates. Additionally, we assessed the accuracy of the model across these categories. As presented in Figure 2, although the model shows comparable accuracy for date and person entities, it assigns higher probabilities to tokens associated with dates. This finding suggests a notable positive bias towards date entities. Similar patterns can be observed in other entities. Such differences in probability assignment highlight the need for recalibration across entities, a feature that current scoring functions fail to adequately address.

Low-Resource Language Challenges. MARS and TokenSAR are dependent on existing NLP tools for implementation. Specifically, TokenSAR uses a sentence similarity model (Duan et al., 2024), and MARS relies on a QA evaluator model (Bulian et al., 2022). These models may not be readily available for some low-resource languages. Moreover, the design of MARS and TokenSAR is primarily oriented towards English. This orientation can be challenging when these tools are applied to languages that are morphologically distinct from English, such as Turkish (Göksel and Kerslake, 2005). In Section 5.4, we experimentally demonstrate that existing methods do not yield comparable improvements in Turkish (compared to English).

4 LARS: Learnable Response Scoring

Let f denote the scoring function, which accepts three arguments: the input prompt $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, the generated sequence $\mathbf{s} = \{s_1, s_2, \ldots, s_L\}$, and the corresponding probability vector $\mathbf{p} = \{p_1, p_2, \ldots, p_L\}$, where p_i represents the probability of token s_i. The function f outputs a real number o. In token probability-based methods, it is desirable for o to be lower when the generation $\mathbf{s}$ is more likely to be incorrect, improving the model's uncertainty estimation. As discussed in Section 3, manually designing an effective scoring function is a challenging endeavor. Thus, we propose making the scoring function f directly learnable through supervised data.

We construct a calibration set to train our scoring function, f_w, which is parameterized by w. This calibration set comprises 4-tuples: input prompt $\mathbf{x}$, generated sequence $\mathbf{s}$, probability vector $\mathbf{p}$, and binary ground truth label g. The label g indicates whether $\mathbf{s}$ is a correct response to $\mathbf{x}$. To optimize the parameters of f_w, we employ the binary cross-entropy loss, denoted by L, applied as follows:

$$L(f_w(\mathbf{x}, \mathbf{s}, \mathbf{p}), g).$$

To train the scoring function f_w, we start with the pre-trained RoBERTa model (Liu et al., 2019) and augment it by adding a linear layer that outputs a single logit. The input format for the LARS model is structured as follows: initial prompt $\mathbf{x}$, followed by a series of response tokens $\mathbf{s} = \{s_1, s_2, \ldots, s_L\}$. Each response token s_i is immediately succeeded by a special probability token $\tilde{p_i}$. This probability token $\tilde{p_i}$ is associated with the probability p_i.

The model incorporates a total of k distinct probability tokens, each corresponding to a specific partition of the [0, 1] probability range. These partitions are mutually exclusive, cover the entire probability range, and are determined based on the quantiles of the probabilities in the calibration dataset. The probability token $\tilde{p_i}$ for p_i is selected according to the partition into which p_i falls.

The embedding vectors of probability tokens are structured by few-hot encoding approach. Assuming the pretrained model has an input dimension d, r-th probability token will be represented by setting the vector positions from $(r-1) \times \frac{d}{k}$ to $r \times \frac{d}{k}$ to 1, while all other positions are set to 0. To ensure consistency with the pretrained model's token embedding norms, we scale these probability vectors by a fixed divisor. f_w is visualized in Figure 1.

With this architecture and input strategy, we enable our scoring function to accurately associate each probability p_i with its corresponding token s_i. By representing p_i using a few-hot vector format, the scoring function effectively utilizes probability information in a manner analogous to conditional image generation tasks (van den Oord et al., 2016). Additionally, using a pretrained model allows the scoring function to grasp the linguistic dependencies and semantic nuances of the tokens. This capability may be crucial in yielding a well-calibrated scoring function to properly employ the probabilities of certain entities, as discussed in Section 3.

5 Experiments

5.1 Experimental Setup

Test Datasets. To test the performance of UE methods, we employ 3 different closed-ended QA datasets. Following (Kuhn et al., 2023), we use a subset of the validation set of TriviaQA (Joshi et al., 2017). Second, we test on the entire validation split of NaturalQA (Kwiatkowski et al., 2019). Lastly, we combine train and validation splits of Web Questions, shortly WebQA (Berant et al., 2013).

LARS Calibration Datasets. To train the model of the proposed method LARS, we employ subsets of the train splits of TriviaQA (Joshi et al., 2017) and NaturalQA (Kwiatkowski et al., 2019). We randomly select ∼13k questions from each dataset and sample six generations per question, ensuring the most likely generation is included, for each model mentioned below. From these generations, we curate unique QA pairs for calibration data. Typically, we train distinct LARS models for each model-dataset combination. In some experiments, we merge TriviaQA and NaturalQA for each model and train accordingly, which we specify when used. To obtain binary ground truths for QA pairs, we utilize GPT-3.5-turbo as in (Bakman et al., 2024; Lin et al., 2023; Chen and Mueller, 2023). Please refer to Appendix D for details and prompt.

Models. We test UE methods on 4 popular models. Llama2-7b-chat (Touvron et al., 2023) and Llama3-8b-instruct (AI@Meta, 2024) are optimized for dialogue use cases. Mistral-7b-instruct (Jiang et al., 2023) and Gemma-7b-it (Team et al., 2024) are instruction tuned versions of the corresponding base models. For the sake of simplicity, we do not use instruction indicator words of the models in the rest of the paper.

Dataset	UE Method	Scoring Function	**Llama2-7b**	**Llama3-8b**	**Mistral-7b**	**Gemma-7b**
TriviaQA	**Lexical Similarity**	-	0.647	0.683	0.720	0.594
	# Semantic Groups	-	0.792	0.819	0.757	0.728
	p(True)	-	0.616	0.842	0.808	0.713
	Confidence	LNS	0.697	0.748	0.722	0.604
		MARS	0.751	0.799	0.745	0.602
		TokenSAR	0.747	0.792	0.747	0.604
		LARS	**0.851**	**0.872**	**0.844**	**0.835**
	Entropy	LNS	0.692	0.747	0.738	0.596
		MARS	0.736	0.801	0.755	0.600
		TokenSAR	0.734	0.793	0.763	0.605
		LARS	**0.842**	**0.864**	**0.849**	**0.830**
	SE	LNS	0.795	0.835	0.810	0.732
		MARS	0.797	0.845	0.810	0.729
		TokenSAR	0.796	0.839	0.813	0.729
		LARS	**0.849**	**0.866**	**0.854**	**0.828**
NaturalQA	**Lexical Similarity**	-	0.600	0.651	0.637	0.546
	# Semantic Groups	-	0.705	0.736	0.675	0.656
	p(True)	-	0.561	0.761	0.730	0.683
	Confidence	LNS	0.677	0.697	0.666	0.608
		MARS	0.714	0.717	0.692	0.645
		TokenSAR	0.703	0.717	0.682	0.637
		LARS	**0.780**	**0.812**	**0.782**	**0.784**
	Entropy	LNS	0.661	0.698	0.679	0.597
		MARS	0.707	0.707	0.701	0.646
		TokenSAR	0.683	0.714	0.694	0.633
		LARS	**0.775**	**0.805**	**0.781**	**0.779**
	SE	LNS	0.721	0.759	0.727	0.667
		MARS	0.730	0.750	0.735	0.670
		TokenSAR	0.721	0.756	0.726	0.669
		LARS	**0.772**	**0.794**	**0.778**	**0.785**

Table 1: AUROC performance of UE methods.

Metrics. Following previous works, we calculate AUROC (Area Under the Receiver Operating Characteristic) score, a commonly used metric used to evaluate the performance of a binary classifier (Kuhn et al., 2023; Bakman et al., 2024; Duan et al., 2024). The ROC curve plots the true positive rate against the false positive rate at various thresholds. AUROC score is the area under this curve, and it provides a single number that summarizes the model's ability to discriminate between the positive and negative classes regardless of the threshold. An AUROC score of 1.0 represents a perfect classifier, while 0.5 is equivalent to random guessing.

Baselines. We use three probability-based UE methods following (Bakman et al., 2024). Confidence is the negative of the response score. It is calculated as the negative score of the most likely generation to a given question. The other UE methods are Entropy as in (2) and Semantic Entropy (SE) (3). Each method uses a scoring function to assign a score to a model generation. We compare LARS with 3 SOTA scoring functions for this purpose: Length-normalized scoring (LNS)(Malinin and Gales, 2021), MARS (Bakman et al., 2024) and TokenSAR (Duan et al., 2024). Our proposal LARS is a scoring function, compared with other baseline scoring functions combined with all probability-based UE methods.

Further, We add three non-probability-based UE approaches to our baseline set. Lexical Similarity (Fomicheva et al., 2020), is the average of the Rouge-L scores between unique sampled generation pairs to a given question. p(True) (Kadavath et al., 2022), a self-check method, asks the model itself if the most likely answer is correct by providing the question, sampled generations, and the answer. Lastly, following (Kuhn et al., 2023), we compare with # Semantic Groups, the number of semantic clusters, as in SE. In all of our experiments, number of sampled generations is 5.

5.2 Main Results

We present the results of our method alongside other baselines in Table 1. Notably, LARS significantly enhances the performance of all existing scoring functions across each probability-based UE method, with improvements reaching up to 0.231 points over LNS. Additionally, LARS boosts the confidence metric to levels comparable with Semantic Entropy (SE) and Entropy. This is particularly important considering the inference cost: Entropy-based methods require multiple output samples (5 in our experiments), which can be com-

UE Method	Scoring Function	**Llama2-7b**	**Llama3-8b**	**Mistral-7b**	**Gemma-7b**
Lexical Similarity	-	0.643	0.640	0.645	0.607
# Semantic Groups	-	0.612	0.599	0.601	0.594
p(True)	-	0.558	0.636	0.668	0.677
Confidence	LNS	0.656	0.645	0.634	0.608
	MARS	0.669	0.659	0.637	0.607
	TokenSAR	0.664	0.656	0.640	0.607
	LARS (TriviaQA only)	**0.718**	0.704	0.681	0.739
	LARS (NaturalQA only)	0.701	0.690	0.682	**0.756**
	LARS (TriviaQA+NaturalQA)	0.715	**0.713**	**0.686**	0.739
Entropy	LNS	0.656	0.650	0.647	0.610
	MARS	0.675	0.664	0.647	0.616
	TokenSAR	0.668	0.661	0.649	0.610
	LARS (TriviaQA only)	**0.719**	**0.704**	0.690	0.730
	LARS (NaturalQA only)	0.712	0.690	0.691	**0.748**
	LARS (TriviaQA+NaturalQA)	0.714	0.703	**0.693**	0.733
SE	LNS	0.672	0.664	0.665	0.629
	MARS	0.679	0.669	0.665	0.629
	TokenSAR	0.674	0.667	0.663	0.625
	LARS (TriviaQA only)	**0.716**	**0.697**	0.689	0.732
	LARS (NaturalQA only)	0.709	0.685	0.693	**0.745**
	LARS (TriviaQA+NaturalQA)	0.711	0.694	**0.697**	0.729

Table 2: AUROC performance of UE methods with different scoring functions on WebQA dataset. LARS models are trained with TriviaQA and/or NaturalQA.

putationally expensive in the context of LLMs. Further, SE necessitates $O(N^2)$ model passes for semantic clustering, where N is the number of sampled outputs. In contrast, LARS operates with a single pass using a RoBERTa-based model with 125M parameters—a computation level that is negligible compared to models with capacities of 7B parameters or more. Lastly, the LARS scoring function demonstrates that probability-based UE methods outperform response clustering methods, including Lexical Similarity, the number of Semantic Groups, and the self-checking method $p(\text{True})$.

5.3 Out-of-Distribution (OOD) Experiments

We train LARS using a calibration dataset, which is curated from a set of questions and the corresponding responses of a chat model. It is crucial to assess the out-of-distribution capabilities of LARS, which we analyze from two perspectives in this section.

OOD Data Generalization. First, we investigate how the performance of LARS is affected when the model encounters questions which have a distribution deviating from that of the calibration set. To this end, we conduct tests using WebQA, with LARS models trained on TriviaQA and/or NaturalQA for each distinct chat model. The results are presented in Table 2, and additional results on out-of-distribution (OOD) data generalization are available in Appendix C.2. Impressively, LARS, despite being trained on different datasets, outperforms all other scoring functions across all probability-based UE methods, achieving an average improvement of approximately ~ 0.04 points.

OOD Model Generalization. Next, we analyze how LARS performs when the responses in the calibration set are derived from a different chat model than the one used at test time. Due to space limitations, we provide a subset of the results in Table 3; however, comprehensive results are presented in Appendix C.1. Notably, optimal LARS performance is achieved when the same chat model is used for both training and testing. Nevertheless, OOD model scores still surpass those of baseline scoring functions (see Table 1 for baselines), confirming the effectiveness of LARS.

UE Method	Calib Model	**Llama2 7b**	**Llama3 8b**	**Mistral 7b**
Confidence	Llama2-7b	**0.858**	0.852	0.835
	Llama3-8b	0.836	**0.874**	0.833
	Mistral-7b	0.831	0.850	**0.852**
Entropy	Llama2-7b	**0.842**	0.852	0.841
	Llama3-8b	0.823	**0.864**	0.841
	Mistral-7b	0.827	0.850	**0.849**
SE	Llama2-7b	**0.850**	0.863	0.850
	Llama3-8b	0.836	**0.872**	0.849
	Mistral-7b	0.840	0.862	**0.859**

Table 3: AUROC scores of UE methods with LARS models trained with answers from various chat models.

5.4 Turkish TriviQA Experiment

To experimentally support our claims regarding the limitations of existing scoring functions in low-resource languages discussed in Section 3, we translated the TriviaQA test and calibration datasets into Turkish using the Googletrans [1]. As illustrated in Table 4, the performance gains of MARS and TokenSAR over the LNS baseline are diminished in

[1] https://py-googletrans.readthedocs.io

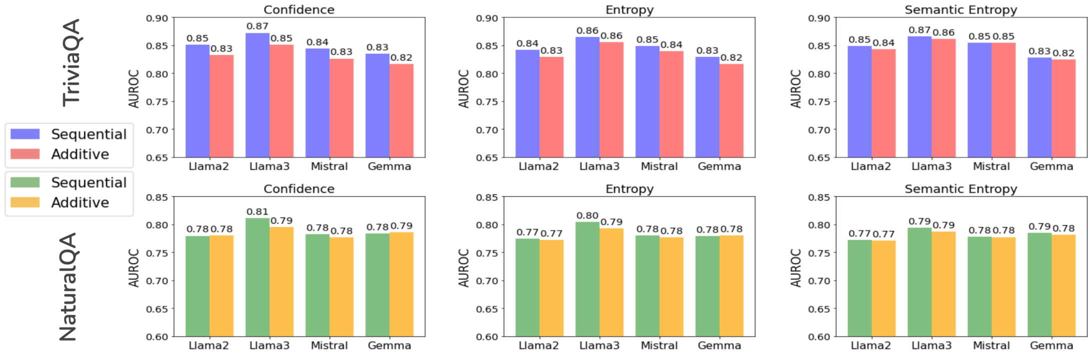

Figure 3: AUROC scores of two different probability association methods for LARS on 2 datasets and 4 models.

the Turkish dataset. This decline is particularly notable for MARS, which incorporates language-specific assumptions in its design, such as phrase separation. In contrast, LARS continues to demonstrate a significant advantage, maintaining its superiority even though the RoBERTa model is pre-trained in English. This indicates that calibration training enables LARS to adapt effectively to different languages.

Scoring Function	**English**	**Turkish**
LNS	0.747	0.692
MARS	0.791 (+0.044)	0.695 (+0.003)
TokenSAR	0.793 (+0.046)	0.720 (+0.028)
LARS	**0.864** (**+0.117**)	**0.814** (**+0.122**)

Table 4: AUROC performance of Entropy with different scoring functions on Llama3-8B for the TriviaQA dataset in different languages.

5.5 LARS without Labeled Data

In this section, we explore the performance of LARS in the absence of labeled data. For this, for each question in the calibration dataset, we first use Llama3-8b to generate answers. To assess the correctness of these answers, we employ a teacher LLM (either Llama3-70b or Llama3-8b) and prompt it to evaluate the correctness of the generated answers. This method produces noisy labels, some of which are incorrect.

Despite these noisy labels, training LARS with them yields a good performance, surpassing both other baselines and the self-evaluation of the LLM (see Table 5). This finding is promising and suggests that the pre-trained nature of the RoBERTa model, which already possesses some understanding of textual inputs, enables it to understand key features from the noisy and partial feedback provided by the teacher LLM. This capability contributes to getting a better scoring function than asking the LLM itself. Such effectiveness of pre-trained models in handling noisy labels supports previous research (Kim et al., 2021), underscoring the potential of LARS for further investigation in such environments.

	Teacher Model	
UE Method	Llama3-70b	Llama3-8b
Ask LLM	0.746	0.635
LARS (No Labeled Data)	**0.837**	**0.809**

Table 5: Results for LARS trained without labeled data on TriviaQA. The Confidence method is used for UE.

6 Ablation Studies

6.1 Probability Association Strategies

In Section 4, we explain a sequential approach to associate tokens of the response with their probabilities, where special probability tokens are placed after each response token in the input to LARS. As an alternative, we explore an additive approach. In this method, the embedding vectors of the probability tokens are added to the embedding vectors of their corresponding response tokens. This strategy effectively reduces the input sequence length for the LARS model. Results (see Figure 3) demonstrate that the sequential approach is, on average, 0.15 points better when used with Confidence, although the gap narrows for Entropy and SE. Comparing the additive approach with other baselines from Table 1, we observe that it still significantly outperforms the baselines. Overall, these two probability association approaches highlight a possible trade-off between shortened input length (to the LARS model) and improved UE performance.

6.2 Size of the Calibration Dataset

To evaluate the scalability of LARS, we calibrate it using different amounts of labeled data. The results, depicted in Figure 4, show that even with as few as 1,000 labeled question-ground truth pairs, LARS outperforms the best-performing baseline. More notably, LARS demonstrates good scalability

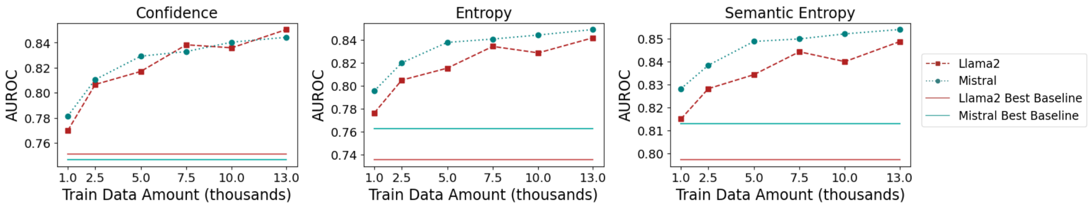

Figure 4: AUROC scores of LARS for different amount of questions in calibration data on TriviaQA. For each UE method, the best score across baseline scoring functions is provided for each model.

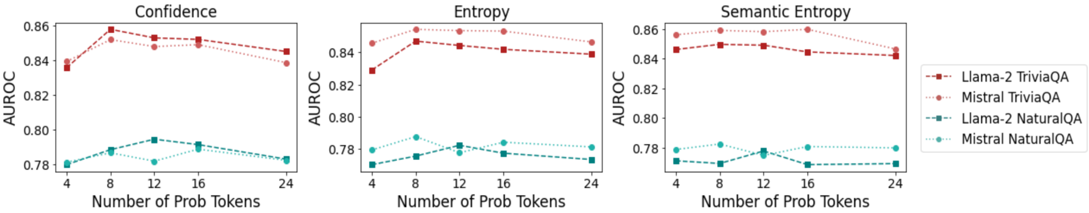

Figure 5: AUROC scores for varying number of probability tokens for LARS on 2 models and 2 datasets.

with calibration data size. Exploring the scaling of LARS with even more data remains as a future direction.

6.3 Importance of LARS Input Components

In this section, we assess the impact of individual input components of LARS on UE.

Number of Probability Tokens. Figure 5 shows the impact of varying the number of probability tokens, k during LARS training. Probabilities are divided into k quantiles, each represented by a unique few-hot vector, as described in Section 4. The choice of k directly influences the bias-variance trade-off of the model. With a high number of probability tokens, the model may overfit, reflecting minor fluctuations in probability within the inputs. Conversely, a small number of tokens might hinder the model's ability to distinguish between significantly different probabilities, as they are represented by identical tokens. Our results indicate that using 8 quantiles for the probability vectors generally yields the best generalization.

Effect of Probability Information. To assess the importance of probability information for LARS, we train a version of the model using only textual inputs: the question and the generated answer. The results (Table 6) indicate that excluding probability information leads to a decrease in the performance of LARS by up to 0.101 points. This significant drop underscores the critical role that probability information plays in the efficacy of LARS.

Effect of Textual Information. To assess the impact of textual and semantic information in the input, we conduct an experiment using only the probability information. Specifically, we train a Multilayer Perceptron (MLP) with two hidden layers, which accepts only the probability vector as input. As presented in Table 6, the probability-only model achieves an AUROC of **0.721** with the Confidence metric, significantly underperforming compared to MARS (**0.751**), TokenSAR (**0.747**), and LARS (**0.851**). These results highlight the crucial role of integrating textual and probability information in enhancing the performance of LARS.

UE Method	Scoring Function	AUROC
Confidence	Only text	0.750
	Only probs	0.721
	LARS	0.851
Entropy	Only text	0.754
	Only probs	0.733
	LARS	0.842
SE	Only text	0.817
	Only probs	0.799
	LARS	0.849

Table 6: Comparison of AUROC performance for the Llama2-7b model on the TriviaQA Dataset across different input modalities: text-only, probabilities-only, and combined text and probabilities.

7 Conclusion

In this study, we demonstrated the shortcomings of existing scoring functions and introduced LARS, an off-the-shelf scoring function directly learned from data. We demonstrated that LARS significantly outperforms existing baselines across three different QA datasets with low computational cost. Additionally, we showed that LARS can be effectively trained even without labeled data, by using a teacher labeling model, and still surpasses the performance of the teacher model. Furthermore, our results indicate that LARS' performance scales well with increased data.

8 Limitations

One limitation of LARS is its reliance on labeled data, which is not a requirement for other scoring functions. While LARS shows promise in environments without labeled data, this aspect requires further investigation to enhance its performance. Further, LARS depends on a pretrained RoBERTa model, which has a limited sequence length capability. This may necessitate the pre-training of Bert-like models that can handle longer sequences. Lastly, training LARS with a transformer model reduces the interpretability of the features. Traditional scoring functions modify the weighting of probabilities and compute a dot product between log probabilities and weights, offering a level of interpretability that LARS, with its more complex function (despite its superior performance), lacks.

9 Ethics Statement

Although LARS demonstrates superior performance compared to existing scoring functions, it is important to remember that these methods still fall short of perfection. Consequently, the results from UE methods should still be taken with a grain of salt, especially in critical domains such as law and medicine. Additionally, LARS may propagate any biases that may be present in its training data into the scoring function, potentially introducing biases in UE related to gender, ethnicity, age, and so on. Such risks must be carefully managed in real-world applications.

References

Lukas Aichberger, Kajetan Schweighofer, Mykyta Ielanskyi, and Sepp Hochreiter. 2024. How many opinions does your llm have? improving uncertainty estimation in nlg. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

AI@Meta. 2024. Llama 3 model card.

Yavuz Faruk Bakman, Duygu Nur Yaldiz, Baturalp Buyukates, Chenyang Tao, Dimitrios Dimitriadis, and Salman Avestimehr. 2024. Mars: Meaning-aware response scoring for uncertainty estimation in generative llms.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. Tomayto, tomahto. beyond token-level answer equivalence for question answering evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 291–305.

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. Inside: Llms' internal states retain the power of hallucination detection. *arXiv preprint arXiv:2402.03744*.

Jiuhai Chen and Jonas Mueller. 2023. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness.

Marie-Catherine De Marneffe and Joakim Nivre. 2019. Dependency grammar. *Annual Review of Linguistics*, 5:197–218.

Nicolas Deutschmann, Marvin Alberts, and María Rodríguez Martínez. 2024. Conformal autoregressive generation: Beam search with coverage guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11775–11783.

Jinhao Duan, Hao Cheng, Shiqi Wang, Alex Zavalny, Chenan Wang, Renjing Xu, Bhavya Kailkhura, and Kaidi Xu. 2024. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models.

Mohamed Elaraby, Mengyin Lu, Jacob Dunn, Xueying Zhang, Yu Wang, and Shizhu Liu. 2023. Halo: Estimation and reduction of hallucinations in open-source weak large language models. *arXiv preprint arXiv:2308.11764*.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

A. Göksel and C. Kerslake. 2005. *Turkish: A Comprehensive Grammar*. Comprehensive grammars. Routledge.

Qian Huang, Jian Vora, Percy Liang, and Jure Leskovec. 2023. Benchmarking large language models as AI research agents.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of* *the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know.

Dohyung Kim, Jahwan Koo, and Ung-Mo Kim. 2021. Envbert: multi-label text classification for imbalanced, noisy environmental news data. In *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–8. IEEE.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The Eleventh International Conference on Learning Representations*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Junyi Li, Xiaoxue Cheng, Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A large-scale hallucination evaluation benchmark for large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6449–6464, Singapore. Association for Computational Linguistics.

Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. 2024. Think twice before trusting: Self-detection for large language models through comprehensive answer reflection.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Junyu Luo, Cao Xiao, and Fenglong Ma. 2023. Zero-resource hallucination prevention for large language models.

Qing Lyu, Kumar Shridhar, Chaitanya Malaviya, Li Zhang, Yanai Elazar, Niket Tandon, Marianna Apidianaki, Mrinmaya Sachan, and Chris Callison-Burch. 2024. Calibrating large language models with sample consistency. *arXiv preprint arXiv:2402.13904*.

Andrey Malinin and Mark Gales. 2021. Uncertainty estimation in autoregressive structured prediction. In *International Conference on Learning Representations*.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.

OpenAI. 2023. GPT-4 Technical Report.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S Jaakkola, and Regina Barzilay. 2023. Conformal language modeling. *arXiv preprint arXiv:2306.10193*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Pier Giuseppe Sessa, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Christian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, Justin Mao-Jones, Katherine Lee, Kathy Yu, Katie Millican, Lars Lowe Sjoesund, Lisa Lee, Lucas Dixon, Machel Reid, Maciej Mikuła, Mateo Wirth, Michael Sharman, Nikolai Chinaev, Nithum Thain, Olivier Bachem, Oscar Chang, Oscar Wahltinez, Paige Bailey, Paul Michel, Petko Yotov, Rahma Chaabouni, Ramona Comanescu, Reena Jana, Rohan Anil, Ross McIlroy, Ruibo Liu, Ryan Mullins, Samuel L Smith, Sebastian Borgeaud, Sertan Girgin, Sholto Douglas, Shree Pandya, Siamak Shakeri, Soham De, Ted Klimenko, Tom Hennigan, Vlad Feinberg, Wojciech Stokowiec, Yu hui Chen, Zafarali Ahmed, Zhitao Gong, Tris Warkentin, Ludovic Peran, Minh Giang, Clément Farabet, Oriol Vinyals, Jeff Dean, Koray Kavukcuoglu, Demis Hassabis, Zoubin Ghahramani, Douglas Eck, Joelle Barral, Fernando Pereira, Eli Collins, Armand Joulin, Noah Fiedel, Evan Senter, Alek Andreev, and Kathleen Kenealy. 2024. Gemma: Open models based on gemini research and technology.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Dennis Ulmer, Martin Gubri, Hwaran Lee, Sangdoo Yun, and Seong Joon Oh. 2024. Calibrating large language models using their generations only. *arXiv preprint arXiv:2403.05973*.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. 2016. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Yuxia Wang, Revanth Gangi Reddy, Zain Muhammad Mujahid, Arnav Arora, Aleksandr Rubashevskii, Jiahui Geng, Osama Mohammed Afzal, Liangming Pan, Nadav Borenstein, Aditya Pillai, Isabelle Augenstein, Iryna Gurevych, and Preslav Nakov. 2024. Factcheck-bench: Fine-grained evaluation benchmark for automatic fact-checkers.

Yasin Abbasi Yadkori, Ilja Kuzborskij, David Stutz, András György, Adam Fisch, Arnaud Doucet, Iuliya Beloshapka, Wei-Hung Weng, Yao-Yuan Yang, Csaba Szepesvári, Ali Taylan Cemgil, and Nenad Tomasev. 2024. Mitigating llm hallucinations via conformal abstention.

Junjie Ye, Xuanting Chen, Nuo Xu, Can Zu, Zekai Shao, Shichun Liu, Yuhan Cui, Zeyang Zhou, Chao Gong, Yang Shen, Jie Zhou, Siming Chen, Tao Gui, Qi Zhang, and Xuanjing Huang. 2023. A comprehensive capability analysis of GPT-3 and GPT-3.5 series models.

Jiaxin Zhang, Zhuohang Li, Kamalika Das, Bradley A. Malin, and Sricharan Kumar. 2024. Sac3: Reliable hallucination detection in black-box language models via semantic-aware cross-check consistency.

Yukun Zhao, Lingyong Yan, Weiwei Sun, Guoliang Xing, Chong Meng, Shuaiqiang Wang, Zhicong Cheng, Zhaochun Ren, and Dawei Yin. 2023. Knowing what llms do not know: A simple yet effective self-detection method. *arXiv preprint arXiv:2310.17918*.

A Details of Turkish Experiment

We translate the same 13k question-ground truth pairs from the train split of TriviaQA to Turkish using Googletrans library[2]. Then, we apply the same procedure as for English: Make the LLM generate 6 answers to the question, ensuring the most likely generation is included. To train LARS, we utilize unique question-response pairs. The labels for training are again obtained by using GPT-3.5-turbo for each QA pair.

To test the performance of varying scoring functions in Turkish, we also translate the question-ground truth pairs of the same test samples of TriviaQA. The same multiple-generation procedure is performed for this set as well and the label is obtained for the most likely generation. After having the translated test set, the Entropy UE metric is calculated by using various scoring functions.

Lastly, the prompts for the LLM are also translated into Turkish to make sure it provides answers in Turkish. Prompts are provided below.

For Llama3-8b to generate answers: [3]

```
System: Sen yardımcı, saygılı ve dürüst
bir asistansın. Sorularımı Türkçe olacak
şekilde net, kısa ve öz cevapla.
User: {question}
```

For GPT-3.5-turbo to obtain labels:

```
You will behave as a question answer
evaluator. I will give you a question,
the ground truth of the question, and
a generated answer by a language model
in Turkish. You will output "correct"
if the generated answer is correct
regarding question and ground truth.
Otherwise, output "false".
Question: {question},
Ground Truth: {gt_answer},
Generated Answer: {generation}
```

B Details of LARS training

We use the pre-trained RoBERTa-base model with a single logit fully-connected layer added at the end. Binary cross entropy loss is used, while the optimizer is AdamW with a learning rate of $5e-6$. The model is trained for 5 epochs. We did a search for batch size in the set of $\{4, 8, 16, 32\}$ and found the optimal batch size as 8 and used it in all of the experiments. The search set for learning rate was $\{1e{-}6, 5e{-}6, 1e{-}5, 5e{-}4, 1e{-}4, 5e{-}4\}$. Lastly, we explored training the model for more epochs (up to 10); however, after epoch 5, we observed overfitting.

The embedding vectors of probability tokens are initialized as few-hot as explained in Section 4 and kept frozen during the training of the model. We also experimented with training those vectors as well as initializing them as fully non-zero random vectors. We observed that the mentioned few-hot strategy gives superior and more stable results. On the other hand, for the additive probability association approach explained in Section 6.1, initializing the embedding vectors as few-hot while keeping them trainable gave the best performance.

C Additional Experiments

C.1 OOD Model Experiments - LARS

In this section, we present extensive OOD Model experiments for LARS. The results are detailed in Table 7, with interpretations similar to those in Table 3. Training LARS on outputs from different LLMs results in an expected performance drop. Nonetheless, LARS continues to outperform other scoring functions, demonstrating its robustness and potential.

In this experiment, for each LLM we use, we train a LARS model using all of the TriviaQA and NaturalQA samples we created for training.

C.2 OOD Data Experiments - LARS

Table 8 details OOD data experiments on NaturalQA, and Table 9 covers OOD data experiments on TriviaQA. Training LARS with data from different distributions results in a performance drop. However, when we integrate the original calibration data with OOD data, LARS achieves better results in NaturalQA experiments. This suggests that increasing the dataset size, even with data from other distributions, might enhance the performance of LARS depending on the dataset.

D Experimental Details

Datasets. To train the LARS model, for each TriviaQA and NaturalQA training split, we randomly select ∼13k samples resulting in ∼60k sampled unique QA pairs. To evaluate the UE methods we use 3 datasets: ∼9k samples from the TriviaQA validation split, the validation set of NaturalQA

[2] https://py-googletrans.readthedocs.io

[3] English translation: You are a helpful, respectful and honest assistant. Give short and precise answers to given questions.

Dataset	UE Method	Scoring Function	**Llama2-7b**	**Llama3-8b**	**Mistral-7b**	**Gemma-7b**
TriviaQA	**Confidence**	Best Score of Baselines	0.7510	0.7994	0.7468	0.6043
		Llama2-7b	0.8577	0.8519	0.8352	0.7932
		Llama3-8b	0.8355	0.8737	0.8327	0.7745
		Mistral-7b	0.8309	0.8499	0.8518	0.7860
		Gemma-7b	0.7997	0.8118	0.8093	0.8399
	Entropy	Best Score of Baselines	0.7356	0.8012	0.7634	0.6053
		Llama2-7b	0.8416	0.8520	0.8410	0.7973
		Llama3-8b	0.8298	0.8642	0.8407	0.7851
		Mistral-7b	0.8271	0.8501	0.8488	0.7926
		Gemma-7b	0.8014	0.8139	0.8216	0.8295
	SE	Best Score of Baselines	0.7973	0.8451	0.8132	0.7318
		Llama2-7b	0.8497	0.8625	0.8496	0.8084
		Llama3-8b	0.8358	0.8719	0.8490	0.7978
		Mistral-7b	0.8402	0.8623	0.8591	0.8057
		Gemma-7b	0.8281	0.8454	0.8400	0.8310
NaturalQA	**Confidence**	Best Score of Baselines	0.7137	0.7166	0.6923	0.6453
		Llama2-7b	0.7886	0.7732	0.7538	0.7232
		Llama3-8b	0.7546	0.8113	0.7543	0.7158
		Mistral-7b	0.7512	0.7679	0.7868	0.7165
		Gemma-7b	0.7455	0.7552	0.7351	0.8091
	Entropy	Best Score of Baselines	0.7071	0.7144	0.7014	0.6459
		Llama2-7b	0.7756	0.7734	0.7569	0.7332
		Llama3-8b	0.7582	0.8103	0.7642	0.7367
		Mistral-7b	0.7550	0.7767	0.7877	0.7317
		Gemma-7b	0.7447	0.7577	0.7403	0.7982
	SE	Best Score of Baselines	0.7301	0.7591	0.7352	0.6701
		Llama2-7b	0.7695	0.7767	0.7627	0.7581
		Llama3-8b	0.7590	0.8038	0.7681	0.7430
		Mistral-7b	0.7574	0.7820	0.7826	0.7458
		Gemma-7b	0.7500	0.7691	0.7489	0.7901

Table 7: OOD Model Experiments on TriviaQA and NaturalQA datasets.

UE Method	Scoring Function	**Llama2-7b**	**Llama3-8b**	**Mistral-7b**	**Gemma-7b**
Confidence	Best Score of Baselines	0.7137	0.7166	0.6923	0.6453
	LARS (NaturalQA only)	0.7685	0.7940	0.7765	0.7846
	LARS (TriviaQA only)	0.7455	0.7689	0.7365	0.7456
	LARS (TriviaQA+NaturalQA)	0.7731	0.7997	0.7774	0.7818
Entropy	Best Score of Baselines	0.7071	0.7144	0.7014	0.6459
	LARS (NaturalQA only)	0.7655	0.7936	0.7781	0.7786
	LARS (TriviaQA only)	0.7434	0.7736	0.7392	0.7468
	LARS (TriviaQA+NaturalQA)	0.7629	0.7918	0.7761	0.7814
SE	Best Score of Baselines	0.7301	0.7591	0.7352	0.6701
	LARS (NaturalQA only)	0.7665	0.7873	0.7770	0.7758
	LARS (TriviaQA only)	0.7511	0.7750	0.7497	0.7572
	LARS (TriviaQA+NaturalQA)	0.7635	0.7849	0.7766	0.7760

Table 8: OOD data experiments on NaturalQA dataset

UE Method	Scoring Function	**Llama2-7b**	**Llama3-8b**	**Mistral-7b**	**Gemma-7b**
Confidence	Best Score of Baselines	0.7510	0.7994	0.7468	0.6043
	LARS (TriviaQA only)	0.8505	0.8721	0.8443	0.8350
	LARS (NaturalQA only)	0.7780	0.8243	0.7893	0.7720
	LARS (TriviaQA+NaturalQA)	0.8414	0.8620	0.8305	0.8152
Entropy	Best Score of Baselines	0.7356	0.8012	0.7634	0.6053
	LARS (TriviaQA only)	0.8381	0.8514	0.8213	0.8415
	LARS (NaturalQA only)	0.7852	0.8348	0.8090	0.7775
	LARS (TriviaQA+NaturalQA)	0.8354	0.8602	0.8373	0.8145
SE	Best Score of Baselines	0.7973	0.8451	0.8132	0.7318
	LARS (TriviaQA only)	0.8488	0.8662	0.8541	0.8281
	LARS (NaturalQA only)	0.8181	0.8515	0.8349	0.7911
	LARS (TriviaQA+NaturalQA)	0.8457	0.8621	0.8493	0.8184

Table 9: OOD data Experiments on TriviaQA dataset

	Question	**Ground Truth**
TriviaQA	David Lloyd George was British Prime Minister during the reign of which monarch?	King George V
	How many symphonies did Jean Sibelius compose?	Seven
	The capital of Brazil was moved from Rio de Janeiro to the purpose-built capital city of Brasilia in what year?	1960
NaturalQA	when was the last time anyone was on the moon	December 1972
	who wrote he ain't heavy he's my brother lyrics	Bobby Scott, Bob Russell
	how many seasons of the bastard executioner are there	one
WebQA	what is the name of justin bieber brother?	Jazmyn Bieber
	what character did natalie portman play in star wars?	Padmé Amidala
	what character did john noble play in lord of the rings?	Denethor II

Table 10: Data samples from the datasets we use to evaluate UE methods: TriviaQA, NaturalQA, and WebQA.

consisting of ∼3500 samples, and ∼6k samples coming from the train and validation sets of WebQA combined.

Example Samples from Datasets. We provide samples from the datasets we use for the evaluation of UE methods in Table 10.

Generation Configurations. We utilize Huggingface library and its built-in `generate()` function to obtain answers. We use `num_beams=1`. For the most likely responses we set `do_sample=False` while for the set of sampled generations, it is True. We set the default LLMs' eos token as end of sentence token to stop the generation.

Computational Cost. We use 40 GB Nvidia A-100 GPUs for all the experiments. The total GPU-hours for training a LARS model with a calibration dataset generated from ∼13k questions is approximately 4. Labeling of the calibration data for one dataset and one model takes approximately 30 GPU-hours. Getting all the results in Table 1 compromises ∼230 GPU-hours excluding LARS training. All presented results are obtained with a single run.

Prompts. The prompts for the LLM models to generate answers to questions are given below.

For LLama2-7b and Llama3-8b:

```
System:You are a helpful, respectful
and honest assistant. Give precise,
short, one sentence answers to given
questions. Do not use emojis.
User:{question}
```

For Mistral-7b:

```
User: Give precise, short, one
sentence answers to given
questions. {question}
```

For Gemma-7b:

```
User: {question}
```

The prompt used for GPT-3.5-turbo to obtain labels:

```
You will behave as a question answer
evaluator. I will give you a question,
the ground truth of the question, and
a generated answer by a language model.
You will output ”correct” if the
generated answer is correct regarding
question and ground truth.
Otherwise, output ”false”.
Question: {question},
Ground Truth: {gt_answer},
Generated Answer: {generation}
```

The prompt for the teacher models explained in Section 5.5 is as follows:

```
System: You are a helpful, respectful
and honest question-answer evaluator.
You will be given a question and a
possible answer. Evaluate the
possible answer as true or false
considering the question. Output
”true” if the answer is correct.
Otherwise, output ”false”. Do not
make any explanation.
User: Question:{question}
Possible answer:{answer}
```

The prompts for the LLM models to self-check their answers for p(True) evaluation is provided below.

For Llama2-7b and Llama3-8b:

```
System: You are a helpful, respectful
and honest question-answer evaluator.
You will be given a question, some
brainstormed ideas and a possible
answer. Evaluate the possible answer
as True or False considering the
question and brainstormed ideas.
Output only True or False.
User: Question:{few_shot_q1}
Here are some ideas that were
brainstormed:{few_shot_samples1}
Possible answer:{few_shot_ans1}
The possible answer is:
Assistant: True
User: Question:{few_shot_q2}
Here are some ideas that were
brainstormed:{few_shot_samples2}
Possible answer:{few_shot_ans2}
The possible answer is:
Assistant: False
User: Question:{question}
Here are some ideas that were
brainstormed:{sampled_generation}
Possible answer:{most_likelt_gen}
The possible answer is:
```

For Mistral-7b and Gemma-7b:

```
User: You are a helpful, respectful
and honest question-answer evaluator.
You will be given a question, some
brainstormed ideas and a possible
answer. Evaluate the possible answer
as True or False considering the
question and brainstormed ideas.
Output only True or False.
Question:{few_shot_q1}
Here are some ideas that were
brainstormed:{few_shot_samples1}
Possible answer:{few_shot_ans1}
The possible answer is:
Assistant: True
User: Question:{few_shot_q2}
Here are some ideas that were
brainstormed:{few_shot_samples2}
Possible answer:{few_shot_ans2}
The possible answer is:
Assistant: False
User: Question:{question}
Here are some ideas that were
brainstormed:{sampled_generation}
Possible answer:{most_likelt_gen}
The possible answer is:
```