Label with Confidence: Effective Confidence Calibration and Ensembles in LLM-Powered Classification

Karen Hovsepian
Amazon Web Services
Seattle, Washington, USA
khhovsep@amazon.com

Dantong Liu
Amazon
Sunnyvale, California, USA
lidanton@amazon.com

Sugumar Murugesan
Amazon
Sunnyvale, California, USA
musuguma@amazon.com

ABSTRACT

Large Language Models (LLMs) have been employed as crowd-sourced annotators to alleviate the burden of human labeling. However, the broader adoption of LLM-based automated labeling systems encounters two main challenges: 1) LLMs are prone to producing unexpected and unreliable predictions, and 2) no single LLM excels at all labeling tasks. To address these challenges, we first develop fast and effective logit-based confidence score calibration pipelines, aiming to leverage calibrated LLM confidence score to accurately estimate the LLM's level of confidence. We propose novel calibration error based sampling strategy to efficiently select labeled data for calibration, leading to a reduction of calibration error by 46%, compared with uncalibrated scores. Leveraging calibrated confidence scores, we then design a cost-aware cascading LLM ensemble policy which achieves improved accuracy, while reducing inference cost by more than 2 times compared with the conventional weighted majority voting ensemble policy.

ACM Reference Format:
Karen Hovsepian, Dantong Liu, and Sugumar Murugesan. 2024. Label with Confidence: Effective Confidence Calibration and Ensembles in LLM-Powered Classification. In *Proceedings of The 1st Workshop on Generative AI for E-Commerce (GenAIECommerce 2024).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Large labeled datasets are essential for building deep neural net based classification models. They are traditionally labor-intensive and time-consuming to create. With the rapid growth of AI research, Large Language Models (LLMs) are being enlisted as automated subject matter experts, tasked with performing complex annotation tasks, and alleviating the burden and cost of manual labeling [1, 2]. There are however two key challenges that undermine wider adoption of LLM-based automated labeling systems in large-scale production environments: 1) LLMs have been known to generate poorly calibrated outputs, resulting in unexpected and unreliable predictions, and 2) there is no single LLM, including GPT4 [3] and Claude Opus [4], that is adept at all labeling tasks.

GenAIECommerce 2024, Oct 25, 2024, Boise, Idaho

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

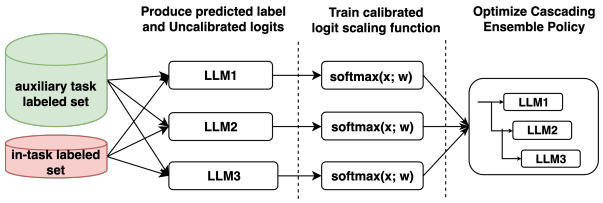

Figure 1: End-to-end view of our LLM confidence calibration and ensemble policy training pipeline.

Undertaking the first challenge requires precise calibration of prediction confidence scores, and there has been significant research on this topic in the past several years [5–8]. We contribute to this body of research by developing a fast and effective logit-based confidence score calibration pipeline, tailored to causal and non-causal autoregressive LMs and classification tasks. We propose several novel strategies for selecting labeled examples used for fitting calibration parameters, aiming to reduce the calibration error with the smallest number of labeled examples.

Following the well-documented successes of conventional ML classification/regression ensembles [9], we could address the second challenge by deploying LLM ensembles as automated labelers. Unfortunately, unlike conventional ML ensembles, deploying LLM ensembles at scale is financially prohibitive in most situations due to high costs associated with LLM inference. With this in mind, we develop a novel algorithm for constructing cascading LLM ensemble policies, which leverage calibrated confidence scores to decide, on a per-input basis, when we can make do with fewer and cheaper models without sacrificing accuracy, and when we need to enlist additional costlier models to ensure correct prediction.

Novel research contributions of our work include:

- Novel "surrogate-error" based label-efficient strategies for sampling in-task labeled examples for learning confidence score calibration parameters; our experimental findings show that these strategies lead to 10% improvement in calibration accuracy over standard labeled example selection strategies and 46% improvement over uncalibrated scores.
- A novel algorithm for optimizing cascading LLM ensemble policies that exploit all opportunities to avoid using costly LLMs, while attempting to maximize the prediction accuracy. Our results show that the proposed algorithm leads to policies that achieve highest accuracy among all tested ensemble and non-ensemble policies, while reducing the inference cost by more than 2X compared with conventional majority voting ensembles.
- To the best of our knowledge, ours is the first work that contains end-end experiments on confidence score derivation, calibration, and utilization in LLM ensembles. These experiments enable us to empirically validate the downstream

business impact of enhancing confidence score calibration in production systems.

Practical contributions and business impact of our work include:

- Implementation of configurable python modules for deriving, calibrating, and evaluating confidence scores for any causal and non-causal autoregressive LM (available via a huggingface interface and BedRock) in classification tasks.
- Implementation of our cascading LLM ensemble policy training and inference algorithms, which can be applied on any classification tasks and any choices of LLMs.
- Experimentally-derived best-practices recommendations for calibration and efficient LLM ensembling; these recommendations come at a time of rapid adoption of GenAI and LLM technologies across the e-commerce sector, and could have significant force-multiplier effect on using LLMs to launch new products and services.

2 CONFIDENCE SCORE CALIBRATION PIPELINE

2.1 Deriving and Calibrating Confidence Scores

A simple yet effective method for generating confidence scores for an autoregresive LM in discrete labeling tasks is to use token-level logits of the output tokens. Prior to computing the confidence scores, we assume that we have the following information: 1) answer a given by the LLM; 2) unscaled activations (a.k.a. logits) $\mathbf{x} \in \mathbb{R}^{D \times T}$ in the last layer of the LLM, over every token in the vocabulary and at each position in the LLM's output (D is the size of vocabulary and T is output sequence length); 3) set of candidate answers/classes C; and 4) matching token sets $\mathcal{M}$, where each set corresponds to one of the candidate answers. Derivation of the calibrated confidence scores uses the following four steps.

Step 1: Logit Aggregation: $\widetilde{x}_c = \max_{i \in \mathcal{M}_c, j=1...T} x_{i,j}$
For every class $c \in C$, take the max value of the logits across all matching tokens and across all positions of the output sequence. This results in a single logit value for each class. In the case of binary classification where the candidate classes are `yes` or `no`, we get two logits, one for `yes` and the other one for `no`.

Step 2: Uncalibrated Scaling: $\mathbf{z} = \mathbf{softmax}(\widetilde{\mathbf{x}})$
Scale the logits by computing the softmax over the logits of the candidate classes.

Step 3: Calibrated Scaling: $\mathbf{z} = \mathbf{softmax}(\widetilde{\mathbf{x}}; \theta)$
To calibrate the scaling step, we endow the softmax function with learnable parameters. We utilize likelihood maximization with a dev-set of logit-label pairs to learn the parameters. We consider two parameterization variants: a) Softmax Vector Scaling **[SV]**, $\mathbf{softmax}(\mathbf{A} \cdot \widetilde{\mathbf{x}} + \mathbf{B})$; and b) Softmax Temperature Scaling **[ST]**, $\mathbf{softmax}(\alpha\widetilde{\mathbf{x}})$. While there are other logit scaling and calibration algorithms, such as Isotonic regression [10] and sigmoid scaling [11], softmax scaling has been showcased as a best-in-class logit calibration strategy in several papers on neural network calibration, such as [6].

Step 4: Prediction Score Selection: $s = z_a$
Conditioned on the LLM's predicted answer, a, we keep the corresponding scaled logit. For example, in the case of binary classification, if the LLM predicts `yes`, we keep the scaled logit for `yes`, otherwise the score for `no`.

2.2 Learning Calibration Parameters

To learn the parameters of our softmax scaling functions (Step 3 in Section 2.1), we must collect a dev-set of labeled examples. We envision two sources of such examples: out-of-task or auxiliary labeled data, and in-task labeled data, recognizing that labeled data for the target tasks may not always be accessible. In both cases, we strive to maximize the calibration accuracy within the budget of the dev-set size.

2.2.1 Out-of-task Labeled Example Setting. In a setting without any labeled examples available from the target task, we can use labeled data from other tasks. For reference, in our product classification use-case, we consider each classification definition (such as 'classify office products') as a distinct task. In this setting, we consider and compare two out-of-task example selection strategies: a) selecting all out-of-task labeled examples **[ALL-OOT]**; and b) selecting only examples from tasks that are semantically similar to the target task **[SIM-OOT]**. The motivation for SIM-OOT strategy is that it may be preferable to fit the calibrator on tasks semantically related to the target task, rather than all auxiliary tasks, which may include unrelated ones. To implement SIM-OOT, we first cluster all auxiliary labeled examples using semantic text representation (we use sentence-T5-xlarge [12]). Then, we learn the parameters of the target task softmax scaler using only the auxiliary tasks belonging to the cluster that also contains the target task.

2.2.2 In-task Labeled Example Setting. In settings where we do have some in-task labeled examples, we can sample a portion to learn the parameters of the softmax scaler. However, since all such labeled examples are ineligible for evaluation or further fine-tuning, we note the importance of maximizing label efficiency. We observe that the logit distribution is often highly skewed, and hypothesize that, because of this, default uniform sampling may not produce enough logits and labels for sections of the logit distribution that need calibration the most.

With this in mind, in addition to Uniform **[U]** sampling we introduce two novel 'surrogate-based' sampling strategies: i) stratification-by-error **[SE]**: stratified sampling with partitions of values of 'surrogate' calibration error, defined as the absolute difference between the uncalibrated score and the top-performing out-of-task calibration score [1], and ii) importance-sampling-by-error **[IE]**: importance sampling using an importance distribution with mass proportional to the 'surrogate' calibration error, defined identically as for SE. The motivation behind these sampling strategies is that we can use the top-performing out-of-task calibrated scores, which we view as 'surrogates' of the actual unknown precision, to prioritize sampling from under-performing uncalibrated score bins.

3 COST-SAVING CASCADING ENSEMBLE POLICY

Calibrated confidence scores enable effective LLM ensembles, since the confidence scores are now comparable across diverse LLMs. Such LLM ensemble policies rely on two key notions: 1) no one model is an expert on all possible tasks, or even on all possible questions within the same task, and 2) different models can have different and complementary specializations. In this section, we

[1] We found it experimentally to be SIM-OOT with Softmax Temp Scaling

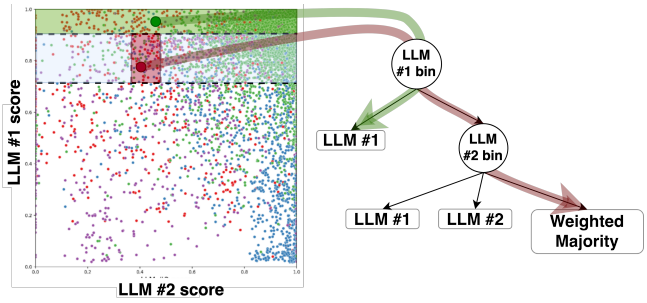

Figure 2: Illustration of two scenarios envisioned by the cascading ensemble policy. In the first scenario, after mapping the confidence score for LLM #1's response for some input prompt [green point] to a bin [green horizontal band], the policy decides LLM #1's answer is good enough. In the second scenario [red point], the policy decides that, based on the LLM #1's confidence bin [light-blue horizontal band], it is better to use two LLMs. After mapping LLM #2's confidence score to a bin [red rectangle], nested in LLM #1's confidence bin, the policy decides that a weighted majority voting strategy using both LLMs' answers is good enough.

present two policies that leverage confidence scores: 1) baseline weighted majority voting policy that requires predictions by all LLMs, and 2) our novel cascading ensemble policy that leverages confidence scores to know when we can make accurate predictions using cheaper models versus when we must use the costlier ones.

Our baseline for benchmarking the performance and cost of our cascading ensemble policy is a weighted majority voting policy [13] which gets model outputs, and associated confidence scores, from all LLMs, and applies the following equation. We assume there are K LLMs in our ensemble, with each one outputting answer a_k, and having an associated confidence score s_k. The formula for the weighted majority vote is:

$$a = \arg\max_{c \in C} \sum_{k \in K} s_k \mathbb{1}[a_k = c], \tag{1}$$

where C is the set of possible classes for a classification problem.

3.1 Cascading Ensemble Policy Training and Inference

There are two key shortcomings of the majority voting policy: 1) requires inference on all LLMs, which can be cost prohibitive, and 2) may not be the optimal strategy, in general. To address these shortcomings, we propose a cost-aware cascading ensemble policy. The cascading approach allows the policy to be judicious about the use of LLMs in the ensemble. Rather than getting answers and confidence scores from all LLMs in parallel, our policy runs inferences on LLMs in sequence, from the cheapest to the costliest. Consider the motivating example illustrated in Figure 2. Suppose that the best answer strategy for some input question is to use the cheapest LLM's answer, and all we need to recognize this is the cheapest LLM's confidence score. Then, there is no need to use the confidence scores of the other costlier LLMs in the ensemble. Continuing with this example, if, however, the cheapest LLM's confidence score does not inspire much trust, then we may decide to get an answer, and an associated confidence score, from another LLM in the ensemble. We will then use both confidence scores

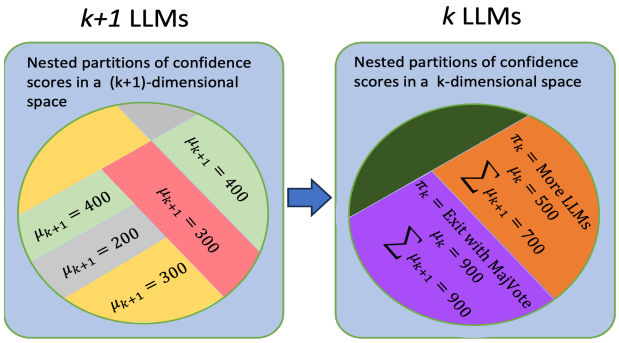

Figure 3: Illustration of the cascading policy training algorithm.

to decide if a "good" answer can be inferred from the first two LLMs' answers, or if, in fact, we may need three or more additional predictions from other LLMs.

To implement this logic in an algorithm, we simplify the problem of mapping real valued confidence scores to discrete decisions by adopting a binning approach. We map each score to a categorical bin index, corresponding to mutually exclusive bins. Bins are parameterized by their edges, which are defined a-priori. The sequential nature of LLM inferences in the cascade means that each LLM's score is binned separately, with its bin index appended to the previous bin indices. After k LLM predictions, the system will generate a vector of k bin indices. This approach allows us to implement the policy's decisions as a simple key-value lookup, with vectors of bin indices as keys, and decisions as values. The trained policy will have a decision for every unique vector of bin indices. Figure 2 contains an illustration of this algorithm using example bins. Algorithm 2 in the appendix includes a formal pseudo-code of our cascading ensemble algorithm.

Figure 3 contains a high-level illustration of the cascading policy training algorithm. We also include the formal pseudo-code in the appendix, in Algorithm 1. The algorithm uses a training set of (score, label) pairs, which can be sourced from out-of-task or in-task datasets [same as for training calibration parameters]. As the illustration shows, the algorithm proceeds in the reverse order, from the full ensemble of K LLMs down to a single LLM ensemble. Furthermore, the figure highlights the grouping of confidence scores into nested bins, defined by the a-priori computed bin-edges. These bins also correspond to nested partitions of the training set. At step k, corresponding to an ensemble of k LLMs, the algorithm iterates over all partitions. Consider the purple partition at step k. The algorithm iterates over a library of ensemble aggregation functions [2], and computes a reward value for each[3]. The algorithm records that the max reward is 900 and the corresponding aggregation function is Majority Vote (MajVote). In the previous iteration of the policy training algorithm – corresponding to $k+1$ LLMs – we had similarly computed max rewards for all partitions, including the partitions nested in the purple partition. This allows us to compare the max k-LLM reward μ_k for the purple partition to the max (k+1)-LLM reward for the purple partition, defined as the sum of max (k+1)-LLM rewards over the sub-partitions. As the example

[2]Our algorithm supports standard heuristic aggregation functions, such as Majority Voting and Weighted Majority Voting, but one could include other more advanced methods, backed by trained ML models.
[3]We define the reward as the number of correct predictions, i.e. hits.

illustrates, both values are 900, which suggests that there is no value add from using k+1 LLMs and that the most judicious yet uncompromising policy decision is to exit the cascade with the recommended Majority Voting aggregation function. The diagram also illustrates a contrasting example – orange partition at step k – where the max k-LLM reward of 500 is lower than the max (k+1)-LLM reward of 700, which forces the policy to recommend going down the cascade to k+1 LLMs.

As we previously stated, training and inference using the cascading ensemble policy requires a-priori defined bin edges for each LLM's scores. Optimizing these bin edges must be done in an outer optimization routine. We experiment with a simple application of Bayesian Optimization, with conditional search spaces, using popular RAY[TUNE] [14] and OPTUNA [15] packages. Prior to optimization, we fix the number of bins of each LLM. Bin edges are optimized over a large number of trials [4] on the objective of maximizing the F1 score, averaged over 10 repeated runs. During each run, we call the policy training [Algorithm 1] and the inference [Algorithm 2] in succession.

4 EXPERIMENTS

We fine-tuned Llama2-7B [16] and Flan-UL2 [17] models with 7.8M internal product classification data for 8.5K product classification tasks. We conduct LLM calibration experiments on 12 binary test product classification datasets which are not used in finetuning LLMs. Each test dataset has around 1000 products with classification labels provided by human annotator. We utilized our fine-tuned Llama2-7B and Flan-UL2 to do zero-shot classification on all the test products. Fig. 4 in the Appendix shows the example prompt we used for our finetuning Llama2-7B model for zero shot classification, and the model is expected to output either `yes` or `no`. Table 4 in the Appendix also includes more detailed information on the 12 test product classification datasets.

4.1 Calibration Results

To assess how well the confidence scores align with the LLM-based classification accuracy, we compute adaptive calibration error (ACE) [7], which gives us a single score measuring the average discrepancy between the LLM-based classification accuracy and the LLM confidence scores. ACE close to 0 indicates good alignment and low discrepancy between the confidence score and the accuracy. We compute ACE using the following formula:

$$ACE(\mathbf{s}, \mathbf{y}) = \frac{1}{|B|} \sum_{b \in B} \left| \bar{\mathbf{s}}_b - \frac{\#[\mathbf{y}_b = \text{“hit”}]}{\#[b]} \right| \tag{2}$$

In the above equation, we assume $B = \{b_i\}_{i=1}^{|B|}$ as the set of partitions created by binning score $\mathbf{s}$, with each partition defined as an array of indices. As the equation shows, ACE is based on two per-bin quantities: average score, $\bar{\mathbf{s}}_b$, and ratio of hits, $\frac{\#[\mathbf{y}_b=\text{“hit”}]}{\#[b]}$, where `hit` means the model makes the right classification decision.

We present the calibration results for fine-tuned Llama2-7B model. In our experiments, we calibrate each classification task separately, in order to better align with the fact that a deployed LLM will process each task in a silo, and the customer's experience depends on the system's performance on that task alone. Therefore, as we aggregate the ACE metrics across the 12 test tasks, we calculate and report measures of spread, in addition to measures of average. In particular, we report the following ACE statistics: mean, standard deviation (std), median and interquartile range (IQR). Together, these statistics allow us to assess the system's reliability in a more holistic manner.

[4]We limited this to 400 trials.

Table 1: Aggregated ACE statistics [lower is better] of fine-tuned Llama2-7B's calibrated confidence scores, spanning the two implemented out-of-task example selection strategies – ALL-OOT and SIM-OOT – and our two softmax scaling variants – SV and ST. We highlight the lowest ACE metrics in bold. Additionally, we mark with an asterisk the solution with smallest upper quartile (i.e. Q3 or 75th percentile) value.

Scenario	Method	Mean±Std	Median±IQR/2
Uncalibrated	–	0.070±0.039	**0.058**±0.027
ALL-OOT	SV	0.089±0.032	0.087±0.022
	ST	0.074±**0.023**	0.073±0.021
SIM-OOT	SV	0.102±0.073	0.087±0.037
	ST	**0.068**±0.027	0.062±**0.015***

Table 2: Aggregated ACE statistics of fine-tuned Llama2-7B's calibrated confidence scores in the in-task setting for each of the 3 few-shot settings. For each few-shot setting, the table reports only the best/lowest value of each ACE statistic, chosen from among all combinations of our in-task example selection strategies – U, SE, IE – and our two softmax scaling variants – SV and ST. In each cell, we report the statistics in the following format: `[value of statistic]/[selection strategy]/[softmax scaling variant]`.

# shots	Mean±Std	Median±IQR/2
0	0.068±0.027/SIM-OOT/ST	0.062±0.015/SIM-OOT/ST
20	0.063±0.023/IE/SV	0.059±0.020/U/SV
50	0.044±0.014/SE/SV	0.044±0.0095/SE/SV
100	0.038±0.015/IE/SV	0.036±0.011/IE/SV

4.1.1 Out-of-task Labeled Example Setting. Table 1 presents the ACE statistics for the uncalibrated scores and calibrated scores in the out-of-task regime, spanning our two softmax scaling versions and the two example selection strategies. We observe that the best calibration pipeline with out-of-task labeled examples uses the semantically similar selection strategy [SIM-OOT] and softmax temperature scaling [ST]. This pipeline has the lowest ACE IQR and mean compared with other pipelines, and achieves comparable median ACE with the baseline uncalibrated score. We highlight the finding that while calibrated scores do not exhibit significant improvement over uncalibrated scores in median and mean ACE, there is a significant improvement in consistency of ACE values across tasks, as evidenced by the much lower ACE IQR and Std for calibrated scores compared to uncalibrated ones.

4.1.2 In-task Labeled Example Setting. We present ACE metrics of experiments in in-task setting with fine-tuned Llama2-7B in Table 2. There is an unsurprising improvement in ACE metrics with increasing number of shots. The 100-shot calibration reduces the mean of ACE by 44% compared with the 0-shot calibration (out-of-task labeled example setting where no labeled example from target task is used for calibration) and by 46% compared with uncalibrated

Table 3: Comparisons (relative to Llama2-7B) of cost metrics and maximum performance metrics across different ensemble policies; cal. means calibrated confidence scores, and uncal. means uncalibrated confidence scores. The highest accuracy metric numbers are highlighted in bold.[5]

Policy	F1	prec.	recall	cost
Llama2-7B alone	F	P	R	X
Weighted maj. vote (uncal.)	1.023F	1.002P	1.034R	7X
Weighted maj. vote (cal.)[a]	1.046F	1.012P	1.066R	7X
Cascading (uncal.)[b]	1.016F	0.998P	1.025R	1.9X
Cascading (cal.)[c]	**1.058F**	**1.035P**	**1.068R**	3.2X

a: represents max performance across all calibration method and sampling strategy combinations.
b: represents max performance across all binning granularities.
c: represents max performance across all calibration method, sampling strategy, and binning granularity combinations.

scores. In each shot scenario, we report the best performing sampling and calibration strategy combination in terms of different ACE metrics in the table. We notice that Softmax Vector Scaling calibration (SV) appears most, indicating SV is superior to the other calibration methods. Similarly, we also observe that our proposed stratification-by-error sampling (IE), and importance-sampling-by-error sampling (SE) show up most of the time in the table, demonstrating these strategies consistently outperform conventional uniform sampling (U) in most cases. Table 5 in the Appendix shows that these strategies lead to 10% improvement in calibration accuracy over conventional uniform sampling.

4.2 Cascading Ensemble Results

We now summarize the findings of experiments on ensembles on the same 12 binary classification dataset described in section 4. Our experiments also use the three LLMs mentioned earlier: our fine-tuned Llama2-7B, fine-tuned Flan-UL2, and Claude-Instant model [4]. We access the Claude-Instant via bedrock API and the confidence score is not available. The majority voting policy requires confidence scores from all candidate models. Inspired by [18], we use our fine-tuned Llama2-7B as a verifier LLM and leverage its logits to measure the confidence score of Claude-Instant model output.

Table 3 shows various performance metrics (F1, precision, and recall) and inference costs across three policies: 1) trained a cost-aware cascading ensemble policy with 3 LLMs using calibrated/uncalibrated confidence score, 2) weighted majority voting ensemble with 3 LLMs using calibrated/uncalibrated confidence score, and 3) non-ensemble policy with fine-tuned Llama2-7B alone.

During experiments with weighted majority voting and cascading ensembles using calibrated confidence scores, we ran all calibration method and sampling strategy combinations described in sections 2.1 and 2.2.2. In the case of cascading ensemble policies, we explored the performance across two binning granularities, i.e. # of confidence bins for each LLM: 5 and 10. Table 3 reports the top-performing combinations. We provide more details and the winning configurations in the appendix section 7.1.It shows that the cascading ensemble policy with calibrated confidence scores not only achieves the best performance among all the benchmarked policies, but also reduces inference cost by more than 2 times compared with the majority voting policy, through significant reduction in the number of more expensive inferences (i.e., Flan-UL2 and Claude). We also observe that compared with uncalibrated confidence score, using calibrated confidence scores improves F1 by 2% and 3.6% for majority voting and cost-aware cascading ensemble policies, respectively.

5 RELATED WORK

LLM confidence score estimation: Existing literature on estimating LLM confidence score relies on 1) verbalized confidence that elicits the model to output confidence directly [19, 20], 2) consistency-based methods that instruct the LLM to generate multiple answers to a question and calculate the consistency of responses [8, 21], or 3) use of model output logits to estimate model confidence [18, 22]. Recent study [18] shows that model output logits are by far the most accurate technique. Therefore, in this study, we focus on leveraging token-level logits to estimate LLM confidence. Comparisons with LLM confidence estimation methods in the other two aforementioned categories are beyond the scope of this work.

LLM confidence score calibration: Calibration in deep neural networks, using logit-based confidence scores, has been an important and popular research problem in the past 10 years, with such noteworthy papers as [6] and [7]. More recently, works like [5, 23–25] have considered the problem of calibrating the outputs of generative LLMs. We also direct the reader to read a, yet unpublished, survey on calibration in LLMs [26].

LLM ensembles: Ensembling LLMs is an area that is gaining popularity in recent times, given that different LLMs excel at different tasks [27–30]. Recent work [27] proposes LLM-BLENDER, that jointly encodes pairs of LLM outputs using a cross-attention encoder, thereby ranking LLMs for each input. While this directly optimizes for classification accuracy, it incurs high cost of inference across all LLMs. Another work [28] employs a reward-guided routing method to route queries to the best LLM. While this work minimizes inference cost by querying only one LLM at a time, it does not truly ensemble LLMs, which is potentially sub-optimal. In contrast to these approaches, we adaptively query LLMs in a sequence, and stop when we are confident of the response we received, thereby striking a balance between cost and accuracy.

6 CONCLUSIONS AND FUTURE WORK

We developed token logit based confidence scores for the classification output from LLM. We proposed novel 'surrogate error' based sampling strategies to increase label efficiency, leading to a reduction in calibration error by 46%, compared with uncalibrated scores. Finally, we design a cost-saving cascading LLM ensemble policy based on LLM confidence score which achieves competitive performance, while reducing inference cost by more than 2 times compared with the conventional majority voting ensemble.

Our work is applicable to any use case of LLMs for data annotation. With more accurate LLM confidence estimation and calibration, we can filter out erroneous LLM labels based on LLM confidence score to ensure annotation quality, and build more cost-effective LLM ensembles. In the future, we plan to explore additional variants of softmax scaling to improve calibration accuracy, and expand the action-space of LLM ensemble to better trade-off speed and accuracy.

REFERENCES

[1] Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. Annollm: Making large language models to be better crowdsourced annotators, 2023.

[2] Parikshit Bansal and Amit Sharma. Large language models as annotators: Enhancing generalization of nlp models at minimal cost, 2023.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] anthropic. Introducing the next generation of claude.

[5] Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

[6] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 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, 2017.

[7] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[8] Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms, 2023.

[9] Ibomoiye Domor Mienye and Yanxia Sun. A survey of ensemble learning: Concepts, algorithms, applications, and prospects. *IEEE Access*, 10:99129–99149, 2022.

[10] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 694–699, New York, NY, USA, 2002. Association for Computing Machinery.

[11] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.*, 10, 06 2000.

[12] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[13] N. Littlestone and M.K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[14] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *ArXiv*, abs/1807.05118, 2018.

[15] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[16] etc Hugo Touvron. Llama 2: Open foundation and fine-tuned chat models, 2023.

[17] Google. google/flan-ul2. In *https://huggingface.co/google/flan-ul2*, 2023.

[18] Nihit Desai Dhruva Bansal. Labeling with confidence. In *https://www.refuel.ai/blog-posts/labeling-with-confidence*, 2023.

[19] Stephanie Lin, Jacob Hilton, and Owain Evans. Teaching models to express their uncertainty in words, 2022.

[20] Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher Manning. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5433–5442, Singapore, December 2023. Association for Computational Linguistics.

[21] Gwenyth Portillo Wightman, Alexandra Delucia, and Mark Dredze. Strength in numbers: Estimating confidence of large language models by prompt agreement. In *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*, pages 326–362, Toronto, Canada, July 2023. Association for Computational Linguistics.

[22] Vaishnavi Shrivastava, Percy Liang, and Ananya Kumar. Llamas know what gpts don't show: Surrogate models for confidence estimation, 2023.

[23] Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. On the calibration of large language models and alignment. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9778–9795, Singapore, December 2023. Association for Computational Linguistics.

[24] Shrey Desai and Greg Durrett. Calibration of pre-trained transformers. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302, Online, November 2020. Association for Computational Linguistics.

[25] Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. Calibrated language model fine-tuning for in- and out-of-distribution data. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1326–1340, Online, November 2020. Association for Computational Linguistics.

[26] Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models, 2024.

[27] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion, 2023.

[28] Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models, 2023.

[29] Mathieu Ravaut, Shafiq Joty, and Nancy Chen. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[30] Hongyi Wang, Felipe Maia Polo, Yuekai Sun, Souvik Kundu, Eric Xing, and Mikhail Yurochkin. Fusing models with complementary expertise, 2023.

7 APPENDICES

Table 4: Details of the SOP-ASIN dataset, on which all experiments in this paper were performed.

SOP Title	# ASINs/prompts	Match label (yes/no)
Bicycle Stem, Bicycle Crankset, Bicycle Cassette, Bicycle Fork, Bicycle Chaining	1118	569/549
Books Hardcover/ Notebook Hardcover	693	397/296
Cotton Bath and Kitchen Towels	1941	1119/822
Dried grains/Rice	938	560/378
Ebike Batteries, Ebike Chargers, Escooter Batteries, Escooter Chargers	764	370/394
Foam Sheet	901	359/542
Non-Gas Fire Pit	2510	1242/1268
SIM cards	703	312/391
Saws/Cutters/Milling Tools	1486	751/735
Ties/Tie Clips	1547	682/865
Window Fittings	876	460/416
Total	**14227**	

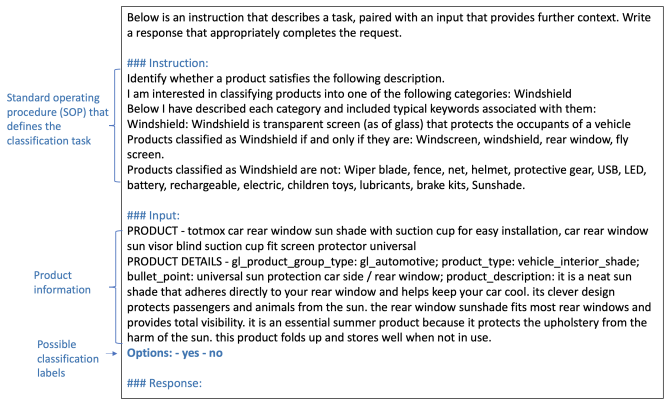

Figure 4: Example LLM prompt for the Llama2-7B model.

7.1 Details of experiments for reproducing results in Table 3

The results in Table 3 are based on a labeled dataset that merges all 12 binary product classification tasks together.

To compute the results in row 3, i.e. weighted majority voting ensemble policy with calibrated confidence scores, we applied formula (1) across all confidence score calibration configurations, spanning both calibration methods, SV/ST, and the three in-task example sampling methods, U/SE/IE. Since no model training is needed for this row, F1, precision, and recall were measured on the entire labeled dataset. The maximum performance across all these configurations, which is quoted in the table, is attained using Softmax Vector Scaling (SV) with 100-shot in-task example sampling using importance-sampling-by-error (IE).

For row 4 – cascading ensemble with uncalibrated confidence scores – the configuration for each experiment consists of one of two binning granularities: 5 or 10. Separately for each granularity, we ran 400 trials of Bayesian Optimization of bin edges [for both LLama2 and Flan-UL2 models][6] using Optuna and Ray packages. During each trial, we computed the computed the F1 score averaged across 10 80%/20% train/test splits, with Algorithm 1 applied to train segment, followed by applying Algorithm 2 to the test segment. The results quoted in the table correspond to the binning granularity of 10, i.e. 10 Llama2 bins and 10 Flan-UL2 bins.

Finally, the experimental design for results of row 5 – cascading ensemble with calibrated confidence scores – is identical to that of row 4, but with an expanded configuration. Each configuration consists of binning granularity, 5/10, calibration method, ST/SV, and in-task example sampling method, U/SE/IE. As for row 4, for each configuration, we perform outer Bayesian Optimization of bin edges, using 400 trials, with each trial consisting of evaluating the average F1 score of an optimized cascading ensemble policy. The max performance and cost in row 5 was attained with the following

Algorithm 1: Algorithm that trains the set of hierarchical policies, which are used by Algorithm 2. The algorithm proceeds in reverse order, from the full ensemble of K LLMs back to an ensemble composed of only the first/cheapest LLM. At each step $k \in \{K, K-1, ..1\}$, it groups all labeled training examples that have the same k-length sequences of bin indices, that contain the confidence scores of LLMs $1..k$. Within each such group, the algorithm picks the decision that maximizes some measure of average reward, e.g. the hit-rate (a.k.a accuracy) calculated from the labels of this group upon applying that decision. Among the possible decisions, all but one are distinct answer strategies, each taking the first k LLM answers, plus associated k scores, and outputting a single answer. The last decision is to go deeper in the cascade and involve more LLMs [referred to as *GoDeeper* in the algorithm]. The choices for answer strategies are also defined a-priori. The previously-described weighted majority voting strategy can be one of these choices. The algorithm first maximizes the average reward over the choices of answer strategies. This is compared with the maximum achievable reward with more LLMs, which has already been computed in the previous iteration $k+1$, using groups of samples with $(k+1)$-length bin indices that have the current k-length bin indices as prefix. Based on this comparison, the optimal decision for this group can be to go deeper in the cascade, or exit from the cascade with the k-LLM answer strategy that maximizes the average reward.

Input: $\mathbf{D} = \{\mathbf{a}_i, \mathbf{b}_i, y_i\}_{i=1}^{N}$: training dataset of triplets of (multi-LLM answer-score pairs, nested bins, and ground-truth answers) for a sample of N training examples; see Algorithm 2 for definitions of $\mathbf{a}$ and $\mathbf{b}$.
$\{\mathcal{A}_k\}_{k=1}^{K}$: see Algorithm 2
Output: $\{\pi_k\}_{k=1}^{K}$
for $k = K$ *to* 1 **do**
  forall $\tilde{\mathbf{b}} \in \mathcal{B}_1 \times \mathcal{B}_2 \cdots \mathcal{B}_k$ **do**
    forall $h \in \mathcal{A}_k$ **do**
      $\mu(h) = \sum_{\substack{i=1 \\ \mathbf{b}_i^{(1:k)}=\tilde{\mathbf{b}}}}^{N} \mathbb{1}[h(\mathbf{a}_i) = y_i]$
    $\pi_k\left(\tilde{\mathbf{b}}\right) = \arg\max_{h \in \mathcal{A}_k} \mu(h)$
    $m_k\left(\tilde{\mathbf{b}}\right) = \max_{h \in \mathcal{A}_k} \mu(h)$
    if $(k < K)$ *and* $\left(m_k\left(\tilde{\mathbf{b}}\right) < \sum_{b \in \mathcal{B}_{k+1}} m_{k+1}\left(\tilde{\mathbf{b}} \cup \{b\}\right)\right)$ **then**
      $\pi_k\left(\tilde{\mathbf{b}}\right) = GoDeeper$
      $m_k\left(\tilde{\mathbf{b}}\right) = \sum_{b \in \mathcal{B}_{k+1}} m_{k+1}\left(\tilde{\mathbf{b}} \cup \{b\}\right)$

Algorithm 2: Algorithm that applies cascading ensemble policy on an arbitrary question, with access to a quorum of LLMs.

Input: $\mathbf{a} = \{(a^{(k)}, s^{(k)})\}_{k=1}^{K}$: array of prediction-confidence score pairs, corresponding to each of the K LLMs in the ensemble, ordered from cheapest to costliest LLM.
$\{\mathcal{A}_k\}_{k=1}^{K}$: K collections of possible actions/answer strategies, where each $\mathcal{A}_k$ is a collection of answer strategies involving only the first k LLMs. Each answer strategy is a function that takes the answers provided by LLMs, and their scores, and outputs a single final answer.
$\mathbf{b} = \{b^{(k)}\}_{k=1}^{K}$: sequence of nested bin indices, where $b_k \in \mathcal{B}_k$ is the index of the bin containing k^{th} LLM's confidence score, conditioned on the $k-1$ previous bins.
$\{\pi_k(\mathbf{b}^{(1:k)})\}_{k=1}^{K}$: set of K policies, one for each sequence of nested bins of depth k, where each $\pi_k : \mathcal{B}_1 \times \mathcal{B}_2 \cdots \mathcal{B}_k \rightarrow \mathcal{A}_k \cup \{GoDeeper\}$ maps the sequence of bins up to depth k to an answer strategy involving the first k LLMs or to a call to go deeper in the cascade hierarchy and involve more LLMs.
Output: final prediction for the input question
for $k = 1$ *to* K **do**
  $h = \pi_k(\mathbf{b}^{(1:k)})$
  if $h \neq GoDeeper$ **then**
    return $h\left((a^{(1)}, s^{(1)}), (a^{(2)}, s^{(2)}), \cdots, (a^{(k)}, s^{(k)})\right)$

Table 5: Full label-assisted results for Llama2-7B. Expanded In-task aggregated ACE statistics, spanning our three in-task sampling strategies – Uniform [U], Stratification-by-error [SE], Importance-by-Error [IE] – and our two scaling variants – Softmax Vector Scaling [SV] and Softmax Temp Scaling [ST] variants. Per-shot best results are underlined, whereas overall optimal metric values are highlighted in bold.

Shots	Sampling Strategy	Method	Mean	Std	Median	IQR	Upper Quartile
10	U	SV	0.123	0.098	0.100	0.080	0.140
		ST	0.103	0.056	0.086	0.054	0.113
	SE	SV	0.098	0.045	0.095	0.023	0.107
		ST	0.103	0.041	0.099	0.044	0.121
	IE	SV	0.110	0.079	0.076	0.055	0.104
		ST	0.104	0.047	0.099	0.047	0.122
20	U	SV	0.069	0.025	0.059	0.020	0.070
		ST	0.078	0.023	0.072	0.033	0.089
	SE	SV	0.072	0.026	0.078	0.029	0.093
		ST	0.084	0.031	0.084	0.049	0.109
	IE	SV	0.063	0.023	0.057	0.036	0.075
		ST	0.080	0.018	0.078	0.022	0.089
50	U	SV	0.049	0.023	0.050	0.030	0.065
		ST	0.067	0.033	0.067	0.052	0.093
	SE	SV	0.044	**0.014**	0.044	**0.019**	0.054
		ST	0.061	0.026	0.058	0.040	0.078
	IE	SV	0.053	0.020	0.054	0.030	0.069
		ST	0.063	0.022	0.053	0.025	0.066
100	U	SV	0.042	0.019	0.040	0.026	0.053
		ST	0.058	0.029	0.053	0.033	0.069
	SE	SV	0.041	0.014	0.044	0.024	0.056
		ST	0.059	0.024	0.057	0.024	0.069
	IE	SV	**0.038**	0.015	**0.036**	0.022	**0.047**
		ST	0.060	0.026	0.052	0.024	0.065

configuration: binning granularity = 10, calibration method = SV, sampling method = IE.

[6]As an example, for binning granularity of 5, we would have 4 bin edges for Llama2 and 4 bin edges for Flan-UL2, for a total of 8 real-valued hyper-parameters.