Pareto Optimal Learning for Estimating Large Language Model Errors

Theodore Zhao
theodorezhao@microsoft.com

Mu Wei
mu.wei@microsoft.com

Joseph S. Preston
sam.preston@microsoft.com

Hoifung Poon
hoifung.poon@microsoft.com

Abstract

Large Language Models (LLMs) have shown impressive abilities in many applications. When a concrete and precise answer is desired, it is important to have a quantitative estimation of the potential error rate. However, this can be challenging due to the text-in-text-out nature of generative models. We present a method based on Pareto optimization that generates a risk score to estimate the probability of error in an LLM response by integrating multiple sources of information. We prove theoretically that the error estimator optimized in our framework aligns with the LLM and the information sources in an Pareto optimal manner. Experimental results show that the risk scores estimated by our method are well correlated with the true LLM error rate, thus facilitating error correction. By dynamically combining with prompting strategies such as self-verification and information retrieval, we demonstrate the proposed method can be utilized to increase the performance of an LLM, surpassing state-of-the-art task specific models.

1 Introduction

Large Language Models (LLMs) have evolved to become impressively powerful in recent developments (Zhao et al., 2023), with Generative Pretrained Transformer (GPT) models showing increasingly effective capabilities. The evolution from GPT-3 (Brown et al., 2020) to GPT-4 (OpenAI, 2023), along with the emergence of other LLMs such as PaLM (Chowdhery et al., 2022) and LLaMA (Touvron et al., 2023), has marked a significant leap in natural language understanding and problem-solving abilities. The generative nature of these models has led to their widespread adoption in numerous application fields. Despite their advanced capabilities, LLMs are capable of generating incorrect results (Ji et al., 2023), an issue that is particularly problematic in applications where precision and dependability are critical, like the biomedical and healthcare fields (Azamfirei et al., 2023; Nori et al., 2023).

Existing approaches for improving the correctness of LLMs include prompt engineering (White et al., 2023), retrieval methods (Chen et al., 2023), domain-specific tuning (Wu et al., 2023; Nguyen, 2023) among many others (Wang et al., 2023). While existing methods show varying degrees of improvement on different tasks, in general there lacks a systematic way to efficiently quantify the likelihood of errors in LLM outputs. One strategy involves querying the LLM in various ways to estimate the answer's correctness (e.g., Manakul et al., 2023). However, these approaches are computationally expensive and biased by the LLM itself.

In many LLM applications, including disease prediction (Han et al., 2023), medical diagnosis (Shea et al., 2023), and question answering (QA) (Moore et al., 2023), a concrete and precise answer is desired. For these mission critical tasks, a quantitative error estimation or confidence level for the response is equally important as giving a correct answer itself. However, the text-in text-out nature of the generative language models makes it challenging to estimate the error probability of the answer quantitatively. Although some LLMs provide internal probability scores for the generated tokens, they are poorly calibrated to the true error rate, particularly after applying reinforcement learning with human feedback (RLHF) (Ouyang et al., 2022; OpenAI, 2023).

Our goal in this paper is to address this issue by establishing a systematic way to quantitatively estimate the likelihood of error in LLM answer. We approach this through training an estimator model h via multi-objective optimization, leveraging extensive research in Pareto optimization (Pareto, 1964). Given the optimized model h and any LLM response, we can then directly estimate the LLM response error rate which we refer to as the Pareto optimal learning assessed risk (POLAR) score (Section 3.2). This framework leverages structured information retrieval from other information sources such as knowledge bases. We introduce a novel approach that trains a Pareto-optimal probabilistic model h to simultaneously optimize on LLM and align with the external information sources.

Our key contributions are as follows:

i) We propose a novel framework using Pareto optimization aligning to the LLM and multiple external information sources.

ii) The POLAR score from our framework is shown experimentally to be effective in estimating LLM error rate.

iii) We demonstrate that POLAR scores can be leveraged to boost an LLM's performance by easily combining with other popular strategies such as self-verification (Weng et al., 2023) and retrieval augmented generation (RAG) (Chen et al., 2023).

2 Related Work

Several heuristics have been proposed to reduce error and estimate the confidence level of an LLM response. Wang et al. (2022) used self-consistency to infer the reliability of the answer. Manakul et al. (2023) proposed SelfCheckGPT as a blackbox method to detect hallucination. The chain-of-thought method (Wei et al., 2022) has also been used to indicate potential errors. These approaches are not able to provide a quantitative error estimation that is calibrated to the true error rate, and susceptible to the issue that the model's self-assessment of confidence is inherently biased. The quality of the results is also highly dependent on the prompting strategy. Estimation of uncertainty in LLM outputs has been studied in Fadeeva et al. (2023); Kuhn et al. (2023); Lin et al. (2023), however, uncertainty does dot directly reflect the confidence score estimating error probability.

Producing confidence scores that are well correlated with a model's error rate has been well established in traditional machine learning. The study of model calibration dates back to the seminal work of Platt scaling (Platt et al., 1999), where a Logistic calibration model is fitted on top of the original model output. Various techniques have been developed afterwards for model calibration, including isotonic regression (Zadrozny and Elkan, 2002), temperature scaling (Guo et al., 2017), and Bayesian binning (Naeini et al., 2015). For LLM, a contextual calibration method for LLMs was proposed by (Zhao et al., 2021), which adjusts the class balance by taking an ensemble of LLM queries with content-free input. These methods rely on annotated calibration data and access to model's output probabilities that are not always available.

The problem of aggregating multiple sources of information or supervision sources is studied extensively in programmatic weak supervision (Zhang et al., 2022; Fu et al., 2020; Varma et al., 2019; Wang and Poon, 2018; Lang and Poon, 2021). Notable works include distant supervision (Hoffmann et al., 2011), crowd-sourcing (Raykar et al., 2010), data programming (Ratner et al., 2016, 2017) and MeTaL, also known as Snorkel, (Ratner et al., 2019). Most of these works weigh the supervision sources across all examples and combine the multiple sources into a single label per example. This approach have shown success but also exhibits significant limitations when applied to identifying LLM errors, primarily due to the weighting dilemma, where if the weight assigned to the LLM is too low, the aggregated result can be noisy, and if the LLM weight is too high, the output is dominated by the LLM, making detecting LLM error difficult. Rühling Cachay et al. (2021) mitigates the weighting problem with instance-dependent weighting, but the expectation maximization procedure demonstrates significant learning variance.

In this work we present a framework to systematically estimate error of an LLM output by simultaneously aligning to multiple information sources while circumventing the weighting dilemma through Pareto optimization.

3 Methodology

Our error estimation framework for LLM responses is a two-step process. In the first step, we iterate through a corpus of input instances to collect the corresponding LLM responses, while dynamically retrieving heuristic answers from other information sources. In this process, a probabilistic function h is learned that fits the multiple sources in a Pareto optimal manner. In the second step, the optimized h model is used to estimate the error rate of the LLM response on any new input instance, which we refer to as the POLAR score.

After the error estimation step, we also provide an optional third step that strategically re-prompt the LLM based on the POLAR score, and leverage the information retrieved from the information sources in an RAG manner (Chen et al., 2023). An overview of the framework is shown in Figure 1.

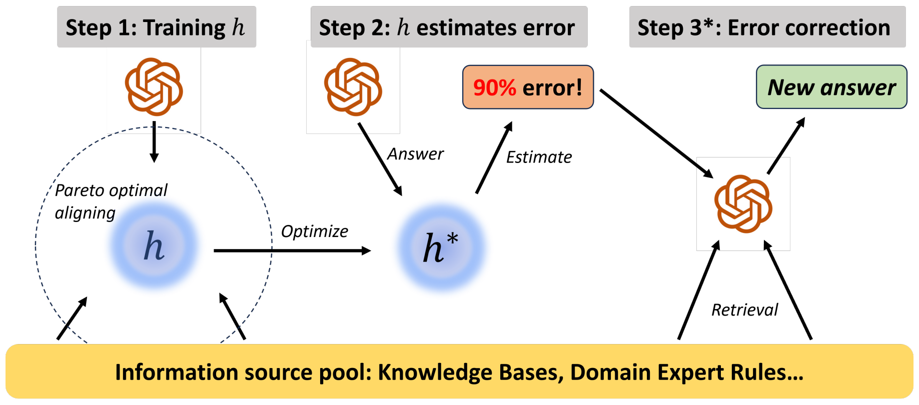

Step 1: Training h
Step 2: h estimates error
Step 3*: Error correction
90% error!
New answer
Answer
Estimate
Pareto optimal aligning
h
Optimize
h^*
Retrieval
Information source pool: Knowledge Bases, Domain Expert Rules...

Figure 1: Pareto optimal learning framework for LLM error estimation and correction.

3.1 Problem setup

Denote the LLM as a function $\mathrm{LLM}(x;p)$ where $x \in \mathcal{X}$ is the input text, and p is the user-defined prompt specific to some task. In order to quantify LLM error, we define $\mathcal{Y}$ as the quantized output space (e.g. answer choices in QA, disease type in diagnosis tasks, etc.). Any free-text LLM output is mapped to the quantized space $\mathcal{Y}$ through mapping $\mathcal{Q}$. Now we can define the LLM answer as

$$\Lambda(x) := \mathcal{Q}(\mathrm{LLM}(x;p)) \in \mathcal{Y}, \qquad (1)$$

where the output space $\mathcal{Y}$ is of cardinality K. Note that the LLM is allowed to state "unsure", but in error estimation we only consider the scenario when the LLM explicitly states the answer. Suppose the true answer for input x is $y \in \mathcal{Y}$, estimating the LLM error rate is to estimate the probability $\mathbb{P}(\Lambda(x) \neq y)$.

To account for other information sources, such as knowledge bases and expert rules, we introduce a pool of m sources. For $j = 1, \cdots, m$, define the triggering function $\mathbb{I}_j(x) \in \{0, 1\}$ indicating if the input x triggers a retrieval from information source j. Example triggering includes recognized entities, text patterns, and etc.. Multiple sources can be triggered at the same time.

Once x triggers retrieval from source j, i.e. $\mathbb{I}_j(x) = 1$, the retrieval function $\lambda_j : \mathcal{X} \to \mathcal{Y}$ represents the answer suggested by information source j for input x. Note that the LLM function Λ defined in (1) is also an information source that maps $\mathcal{X} \to \mathcal{Y}$. For simplicity, we denote $\lambda_0 := \Lambda$ and $\mathbb{I}_0(x) \equiv 1$. The answers retrieved from the multiple information sources can conflict with the LLM and among themselves, thus require a proper aggregation, which is the main challenge to solve.

3.2 Pareto Optimal Learning Assessed Risk

Step 1: Pareto Optimal Learning We propose to learn an optimal function $h : \mathcal{X} \to K$-simplex by simultaneously fitting it to the output of the LLM and other information sources. Incorporating multiple sources helps reduce error dependency on a single source, and corrects the bias from the LLM itself by aligning to external knowledge. The primary challenge here is to design a framework to resolve conflicts among different sources and with the LLM. We do not assume the sources to be independent, as it is often assumed in the weak supervision literature (Zhang et al., 2022). Instead we adopt the looser assumption that the sources are individually positively correlated with the correct output y, better than a random guess. Ideally, a reasonable h should align with each source j if it is triggered for retrieval, which is measured by

$$\ell_j(h;x) := \mathbb{I}_j(x) \cdot \mathcal{L}(h(x), \lambda_j(x)), \qquad (2)$$

where $\mathcal{L}$ is the cross-entropy loss of h against the information source. Note that $h(x) \in K$-simplex, which essentially estimates the probability distribution on the output space $\mathcal{Y}$ with cardinality K.

In order for h to align with the multiple information sources, mathematically, we want to solve the following multi-objective problem for $j = 0, \cdots, m$:

$$\min_{h \in \mathcal{H}} \quad \mathbb{E}_{x \sim \mathcal{X}}[\ell_j(h;x)]_{j=0}^{m}. \qquad (3)$$

As the objectives may conflict, we seek an $h^* \in \mathcal{H}$ that is *Pareto optimal*, following multi-objective learning theory (Hwang and Masud, 2012) and Pareto optimization (Pareto, 1964).

Definition 1 (Pareto optimal). $h^* \in \mathcal{H}$ is Pareto optimal to $\lambda_0, \cdots, \lambda_m$, if no $h \in \mathcal{H}$ exists that Pareto dominates h^* in (3). h^* must satisfy

$$\nexists\, h \in \mathcal{H},\ \text{s.t.} \begin{cases} \forall j = 0, \cdots, m, \\ \quad \mathbb{E}[\ell_j(h;x)] \leq \mathbb{E}[\ell_j(h^*;x)], \\ \exists j = 0, \cdots, m, \\ \quad \mathbb{E}[\ell_j(h;x)] < \mathbb{E}[\ell_j(h^*;x)]. \end{cases}$$

The Pareto optimization framework effectively manages dependencies between information sources. For example, a Pareto optimal h remains unaffected by the arbitrary duplication of a source. However, finding Pareto optimal solutions remains challenging in the multi-objective optimization literature. In this paper, we adopt one of the standard approaches to scalarize the multiple objectives (Hwang and Masud, 2012), and solve

$$\min_{h \in \mathcal{H}} \mathbb{E}_{x \sim \mathcal{X}}[G(\vec{\ell})], \tag{4}$$

where $\vec{\ell} = (\ell_0, \cdots, \ell_m)$ and G is a Pareto aggregator as defined below.

Definition 2 (Pareto aggregator). $G : \mathbb{R}_+^{m+1} \rightarrow \mathbb{R}_+$ is a *Pareto aggregator* if it satisfies:

- G is convex, and
- $G(\vec{\ell}'_j) < G(\vec{\ell})$ if $\ell'_j < \ell_j \ \forall\ j = 0, \cdots, m$, where $\vec{\ell}'_j = (\ell_0, \cdots, \ell'_j, \cdots, \ell_m)$.

In this study, we explore four different types of aggregators:

- Linear: $G(\vec{\ell}) = \sum_{j=0}^m \ell_j = \|\vec{\ell}\|_1$,
- Quadratic: $G(\vec{\ell}) = \left(\sum_{j=0}^m \ell_j\right)^2 = \|\vec{\ell}\|_1^2$,
- Euclidean norm: $G(\vec{\ell}) = \sqrt{\sum_{j=0}^m \ell_j^2} = \|\vec{\ell}\|_2$,
- Chebyshev: $G(\vec{\ell}) = \max_{j=0}^m \ell_j = \|\vec{\ell}\|_\infty$.

The nonlinear aggregator shapes the optimization in Equation (4) differently through Jensen's inequality. While the first three aggregators qualify as Pareto aggregators, the Chebyshev aggregator does not meet the definition criteria, serving as a comparative element in our experiment.

With Definitions 1 and 2, we propose finding an optimal solution h^* by solving Equation (4), which can be done via standard stochastic gradient descent algorithms such as Adam (Kingma and Ba, 2014). The solution is guaranteed by the following Theorem, with a detailed proof in Appendix A.

Theorem 1. *Suppose G is a Pareto aggregator as in Definition 2, solving the problem in Equation 4 approximates a Pareto optimum by minimizing the upperbound.*

Step 2: POLAR Score Estimation Once a optimal solution h^* is found, we can estimate the error rate of $\Lambda(x)$ for any new input $x \in \mathcal{X}$, by selecting the probability in $h^*(x)$ that corresponds to $\Lambda(x)$, denoted $h^*(x)[\Lambda(x)]$, and compute a risk score

$$\begin{aligned} \zeta(x, \Lambda(x); h^*) &= \mathbb{P}_{Y \sim h^*(x)}(\Lambda(x) \neq Y) \\ &= 1 - h^*(x)[\Lambda(x)], \end{aligned} \tag{5}$$

where $\mathbb{P}_{Y \sim h^*(x)}$ represents the probability distribution of Y as estimated by $h^*(x)$. We refer to $\zeta(x, \Lambda(x); h^*)$ as the Pareto optimal learning assessed risk (POLAR) score.

Algorithm 1 summarizes the entire process.

Algorithm 1 POLAR score estimation

```
Step 1: Training estimator $h^*$
 1: Input: LLM with prompt $p$, answer mapping $\mathcal{Q}$, retrieval triggering functions $\mathbb{I}_{1:m}$, information sources $\lambda_{1:m}$, training inputs $x_{1:n}$, Pareto aggregator $G$, and initialized $h \in \mathcal{H}$.
 2: for $i = 1$ to $n$ do
 3:     $\Lambda(x_i) = \mathcal{Q}(\text{LLM}(x_i; p))$
 4:     $\text{L}_{\text{LLM}} = \mathcal{L}(h(x_i),\ \Lambda(x_i))$
 5:     for $j = 1$ to $m$ do
 6:         $\text{L}_j = \mathbb{I}_j(x_i) * \mathcal{L}(h(x_i),\ \lambda_j(x_i))$
 7:     end for
 8:     Update $h$ with SGD iteration of $\min_h G(\text{L}_{\text{LLM}},\ \text{L}_1, \cdots, \text{L}_m)$.
 9: end for
10: Output: Optimized $h^*$.
Step 2: Estimation with $h^*$.
 1: Input: New input $x$, optimized $h^*$, LLM with prompt $p$, answer mapping $\mathcal{Q}$.
 2: LLM answer $\Lambda(x) = \mathcal{Q}(\text{LLM}(x; p))$
 3: Output: POLAR score $\zeta(x, \Lambda(x); h^*) = 1 - h^*(x)[\Lambda(x)]$ in Equation (5).
```

3.3 Step 3*: Error Correction with POLAR

Identifying LLM responses with a higher risk of error presents an opportunity to efficiently correct the errors and improve the final accuracy. In most of the applications, the POLAR score itself is sufficient to facilitate human-in-the-loop intervention. Here we provide an optional Step 3, which easily connects the POLAR score to other prompting strategies to correct the error automatically. In this setting, the information sources also serve as additional input to the LLM to enhance its answer. In this paper, we propose two dynamic prompting strategies to illustrate correcting LLM errors using the POLAR score $\zeta(x, \Lambda(x); h^*)$.

Dynamic self-verification Pick risk threshold δ. For any input x and LLM answer $\Lambda(x)$, if $\zeta(x, \Lambda(x); h^*) > \delta$, simply ask the LLM to self-verify its previous answer.

POLAR-assisted RAG Pick risk threshold δ. For any input x and LLM answer $\Lambda(x)$, if $\zeta(x, \Lambda(x); h^*) > \delta$, retrieve information from all sources that are triggered, i.e. $\mathbb{I}_j(x) = 1$. Provide the answer suggested by the information sources $\lambda_j(x)$ and the description of the sources to the LLM, and generate the revised answer. Algorithm 2 outlines the POLAR-assisted RAG. We provide detailed prompting design description in the Appendix D.

Algorithm 2 POLAR-assisted RAG

```
1: Input: Input $x$, optimized $h^*$, LLM with prompt $p$, answer mapping $\mathcal{Q}$, retrieval triggering functions $\mathbb{I}_{1:m}$, information sources $\lambda_{1:m}$, text description of the sources $d_{1:m}$, and user-defined POLAR threshold $\delta$.
2: Initial LLM answer $\Lambda(x) = \mathcal{Q}(\text{LLM}(x; p))$
3: Compute POLAR score
       $\zeta(x, \Lambda(x); h^*) = 1 - h^*(x)[\Lambda(x)]$
4: if $\zeta(x, \Lambda(x); h^*) > \delta$ then
5:     Initialize a followup prompt $p'$.
6:     for $j = 1$ to $m$ do
7:         if $\mathbb{I}_j(x) = 1$ then
8:             Get source answer $\lambda_j(x)$.
9:             Get source description $d_j$
10:            $p' = p' + \lambda_j(x) + d_j$.
11:        end if
12:    end for
13:    $\Lambda'(x) = \mathcal{Q}(\text{LLM}(x; p'))$
14: end if
15: Output: Updated answer $\Lambda'(x)$
```

4 Experiments

Dataset In order for the results to be reproducible, it is essential that both the datasets themselves and the associated extra information sources are fixed in the benchmarks. We leverage the publicly available datasets collected by (Zhang et al., 2021) and evaluate on four different NLP tasks: CDR (Li et al., 2016), ChemProt (Krallinger et al., 2017), SemEval (Hendrickx et al., 2019), and SMS (Almeida et al., 2011). The pre-defined supervision functions in these benchmarks serve as the information sources λ_j in addition to the LLM output Λ. We do not use any of the ground truth in the training sets, and only use the ground truth labels on test set for evaluation of the LLM error rate and error correction performance. Our dataset selection covers the following aspects:

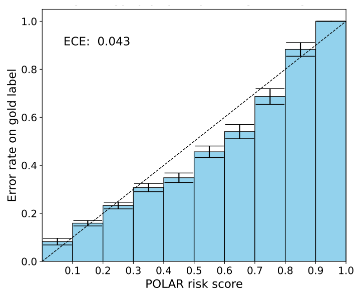
- Domain: General domain tasks(SemEval, SMS) that requires general instruction following and reasoning, as well as biomedical domain tasks (CDR, ChemProt) that requires domain knowledge.

- Difficulty: Tasks that are easy for advanced LLMs, like SMS (99% F1 for GPT-4), and tasks that are still challenging, like CDR (74% F1 for GPT-4), ChemProt (42% F1 for GPT-4), and SemEval (67% F1 for GPT-4). We illustrate that our framework can effectively estimate the error rate even when the LLM is already nearly perfect on the task.

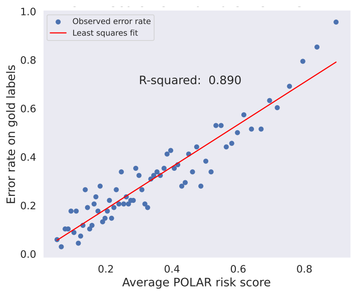
Prompt design To maximize LLM capabilities, we carefully design prompts for each problem, clarifying the problem setting, knowledge background, input/output structure, and instructions for stating "unsure". Detailed prompt information is provided in Appendix D.

Information sources The information sources in these datasets were created by human experts that utilize knowledge bases (e.g. the Comparative Toxicogenomics Database; Davis et al., 2021), textual patterns, and a combination of the two (Ratner et al., 2017; Yu et al., 2021; Zhou et al., 2020; Awasthi et al., 2020). We refer to the benchmark by (Zhang et al., 2021) for detailed description.

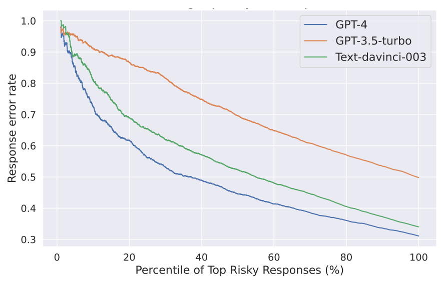
Optimization The Pareto optimal $h(x)$ is obtained by solving Equation 4, and we choose the quadratic aggregator for G and the BERT model (Devlin et al., 2018) for h. For the biomedical CDR and ChemProt, we choose BiomedBERT (Gu et al., 2020) for h. Ablations of model architectures and aggregators are examined in Section 5. Details of the training setup is described in Appendix C.



(a) Error calibration curve



(b) Correlation with error rate



(c) LLM error detection

Figure 2: LLM error estimation using the POLAR score. (a) The LLM response error rate vs ten equal-interval POLAR score bins. (b) The POLAR scores are sorted and then binned where each bin contains 100 examples. The average of the LLM errors and POLAR scores are plotted for each bin. The last bin with the top POLAR scores may have less than 100 examples. (c) shows the average LLM error rate vs top percentile POLAR score examples.

4.1 LLM Error Estimation

We present the LLM error estimation results using the POLAR score and compare them with existing methods as baselines.

4.1.1 Evaluation metrics

As the primary interest is in how well the LLM error rate is estimated, we report evaluation results measured in expected calibration error (ECE) (Naeini et al., 2015) that is widely used in assessing model confidence calibration as well as the coefficient of correlation R^2. An ECE of 0 indicates perfectly estimated error rates, or equivalently a perfectly calibrated model h in our framework.

For each dataset, the training split is used as the input examples $x_1, \cdots, x_n$ along with the predefined information sources λ_i to learn h^*. The test split and its ground truth labels are used obtain to $\Lambda(x)$, compute the true LLM error rate, and evaluate h^* measured by ECE and R^2.

Figure 2 displays POLAR score calibration for GPT-4 on the CDR chemical-disease relation extraction task. The calibration curve (Figure 2a) shows that the POLAR score reliably estimates the true probability of LLM error rates. Figure 2b demonstrates a high correlation between the POLAR score and the true error rate. Figure 2c reveals that responses with the highest POLAR scores are most prone to errors, with the top scores indicating nearly a 100% error rate.

4.1.2 Baseline methods

We compare the proposed POLAR score with the following baseline error estimation approaches. We divide them based on if they utilized the information sources that were used in our framework.

Without information sources We estimate the error rate through LLM ensemble:

1. Query the LLM multiple times with the same input x to sample responses.

2. Estimate the probability where the answer is different from the most frequent one, and use it as the estimation for error rate on input x.

As this approach is extremely expensive, we only evaluated this for GPT-3.5-turbo on the CDR dataset. The estimated ECE in this experiment is 0.4466, which is far worse than other approaches in Table 1. Therefore, we focus on comparison with other baseline methods in the rest of this section.

With information sources The following methods utilize the exact same information sources as in our framework. They differ in how the answers from the multiple sources and the LLM were combined together.

- Snorkel (Ratner et al., 2019): a weak supervision method combining multiple supervision sources via matrix completion. Snorkel model fitted on training set is use to give class probabilities for LLM error rate. We use the class probability given by the fitted model to estimate LLM error rate.

- WeaSEL (Rühling Cachay et al., 2021): A state-of-the-art weak supervision framework that fits a neural network using the LLM and weak labels in an end-to-end manner. We use the class probability given by the fitted model to estimate LLM error rate.

Table 1: LLM error rate estimation performance, using the POLAR score and other methods, measured in ECE and R^2. The best entries (low ECE, high R^2) from each row are highlighted in bold.

Task	LLM	**POLAR**		Snorkel		WeaSEL		Majority vote	
		ECE	R^2	ECE	R^2	ECE	R^2	ECE	R^2
CDR	GPT-4	**0.043**	**0.890**	0.167	0.299	0.146	0.387	0.145	0.348
	GPT-3.5-turbo	**0.046**	0.934	0.164	0.320	0.081	**0.942**	0.182	0.540
	Text-davinci-3	**0.055**	**0.907**	0.154	0.371	0.135	0.877	0.149	0.450
ChemProt	GPT-4	**0.035**	**0.934**	0.182	0.510	0.278	0.885	0.233	0.244
	GPT-3.5-turbo	**0.048**	**0.944**	0.228	0.625	0.219	0.922	0.282	0.031
	Text-davinci-3	**0.051**	**0.917**	0.218	0.700	0.213	0.846	0.279	0.307
SemEval	GPT-4	0.079	**0.916**	**0.068**	0.714	0.612	0.626	0.115	0.379
	GPT-3.5-turbo	**0.047**	**0.963**	0.150	0.821	0.345	0.890	0.277	0.208
	Text-davinci-3	**0.067**	**0.950**	0.119	0.796	0.455	0.784	0.242	0.396
SMS	GPT-4	**0.014**	**0.980**	0.244	0.089	0.409	0.345	0.588	0.091
	GPT-3.5-turbo	**0.041**	**0.963**	0.075	0.202	0.286	0.731	0.148	0.006
	Text-davinci-3	**0.023**	**0.943**	0.201	0.053	0.420	0.238	0.325	0.091

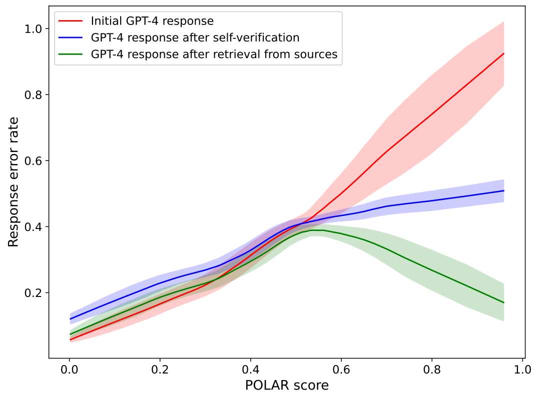
- Majority vote: A common method that estimates class probability according to the voted ratios among all information sources.

Table 1 compares POLAR score performance in LLM error calibration against baseline methods. We report the results spanning the four datasets and three LLMs (GPT-4, GPT-3.5-turbo, and text-davinci-003). The proposed POLAR score consistently outperforms other methods. Among the baseline methods, Snorkel, WeaSEL, and LLM distilled model can achieve top or close-to-top performance in some cases under specific metric, but lack the consistency to deliver stable calibration for different LLMs on different tasks. In comparison, the proposed POLAR score is consistently well-calibrated to the true error rate.

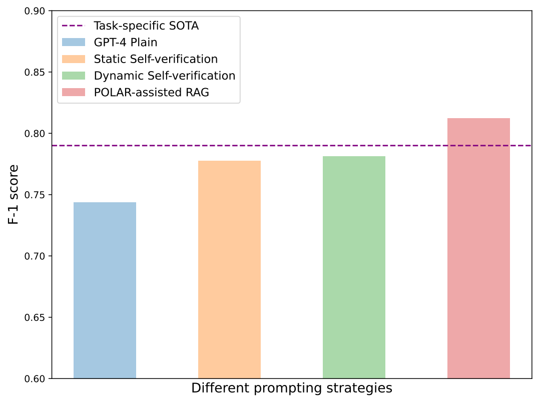
4.2 Improved LLM performance with POLAR-assisted dynamic prompting

We investigate the utility of the POLAR score through dynamic prompting (Section 3.3) to rectify LLM errors. In this experiment, we focus on the CDR dataset and GPT-4 and GPT-3.5-turbo.

To understand the advantage of dynamic prompting, we first examine the effect of LLM error rate using *static* prompting. That is, ignoring the POLAR score, we persistently follow up the initial LLM response with another prompt, either using the self-verification or providing suggested answers from the information sources. The results are shown in Figure 3a where the LLM error rate is



(a) Error rate reduction conditioning on POLAR score.



(b) Dynamic prompting performance

Figure 3: (a) shows the GPT-4 error rate before and after re-prompting, as plotted against the POLAR score. (b) shows the performance improvement using the two dynamic prompting strategies in Section 3.3.

Table 2: LLM error estimation performance with and without external information sources.

Dataset	CDR		ChemProt		SemEval		SMS	
	ECE	R^2	ECE	R^2	ECE	R^2	ECE	R^2
POLAR (with source)	**0.043**	**0.890**	**0.035**	**0.934**	0.079	0.916	0.014	**0.980**
Without Sources	0.164	0.592	0.216	0.766	**0.063**	**0.947**	**0.013**	0.977

Table 3: Average LLM error estimation performance for different loss aggregator and h modeling choices.

Aggregator	Linear		Quadratic		Euclidean norm		Chebyshev	
Architecture	ECE	R^2	ECE	R^2	ECE	R^2	ECE	R^2
BERT	0.0625	0.9273	**0.0458**	**0.9366**	0.0549	0.9003	0.0711	0.8260
MLP	**0.0555**	**0.9392**	0.0974	0.9188	0.0691	0.9302	0.0775	0.8934
LR	**0.0641**	**0.9360**	0.1072	0.9020	0.0766	0.9288	0.0948	0.8813

plotted with respect to the POLAR score. We observe that the LLM error rate decreases with static follow-up prompts when the POLAR score is high, around > 0.5. However, the LLM error rate increases slightly with follow-up prompting when the POLAR score is low, around < 0.5. This suggests the utility of dynamic prompting – follow up the initial LLM response with a prompt only when the LLM is more likely to make a mistake, i.e. when the POLAR score is high.

We apply dynamic prompting to the same problem and set $\delta = 0.5$. The results are shown in Figure 3b. We see that the POLAR-assisted dynamic prompting increases the GPT-4 performance. The POLAR-assisted RAG strategy has a larger increase due to incorporating additional information sources λ_j in the follow-up prompt. We note that GPT-4 with POLAR-assisted RAG outperforms state-of-the-art supervised task-specific model (Xiao et al., 2021).

5 Ablations

External sources of information To illustrate the importance of these additional sources, we examine the performance without them by removing all λ_j for $j = 1, \cdots, m$, and only take $\lambda_0 := \Lambda$ as the LLM itself in the framework. Note that in that sense, solving Equation (3) becomes a single-objective learning problem, which essentially is equivalent to distilling a small model h from Λ. We show the error estimation comparison for GPT-4 in Table 2. We can see that the absence of additional information sources generally leads to inconsistent error estimation capabilities, as they are essential to prevent h from overfitting to LLM responses. This is especially obvious for the biomedical domain tasks CDR and ChemProt, where domain knowledge is critical in error estimation.

Pareto loss scalarizer and harmonizer The effectiveness of different loss aggregators and h modeling choices is encapsulated in Table 3, where we present the ECE and R^2 measures averaged across three LLMs (GPT-4, GPT-3.5-turbo, and text-davinci-003) for the four tasks. We observe that the nonlinear quadratic loss aggregator, when combined with BERT finetuning, delivers superior error estimation performance. Conversely, simpler models, such as Multi-Layer Perceptron (MLP) and Logistic Regression (LR), achieve optimal results with a basic linear aggregator. Notably, the Chebyshev aggregator consistently underperforms across various scenarios. This empirical evidence lends support to Theorem 1, validating the necessity of a Pareto loss aggregator.

6 Conclusion

We presented a novel framework for LLM error estimation using Pareto optimal learning. The error estimator learned in our framework aligns with the LLM and other information sources Pareto optimally. We showed experimentally that the proposed POLAR score is well calibrated with the LLM error rate evaluated on ground truth, ensuring reliable error estimation. We proposed two POLAR-assisted dynamic prompting strategies, and showed that POLAR-assisted RAG enhances GPT-4's performance, surpassing state-of-the-art task-specific model. This development marks a substantial advancement in the application of LLM, providing an effective method to both estimate and reduce LLM errors.

Limitations

According to Theorem 1, we approximate a Pareto optimum by minimizing its upperbound. However, due to the complexity of modeling the errors present in an LLM response and other information sources, we cannot offer further theoretical guarantees of how close an approximated solution is to a Pareto optimum. Nonetheless, we empirically demonstrate in Section 4 that the approximated solution h^* closely estimates the LLM error rate.

In this work, we require external information sources for retrieval. While they are essential for injecting knowledge in addition to the LLM, we admit that not they are not available for every applications. Our framework also focuses on clearly defined error cases where there is a concrete answer. We envision future work to expand into areas where the LLM output is more ambiguous.

Ethics Statement

The motivation of this work is to reduce errors produced by LLMs so that LLMs can be beneficial to users and applications by providing factual and accurate responses. Potential risk includes improper use of the error estimator.

References

Tiago A Almeida, José María G Hidalgo, and Akebo Yamakami. 2011. Contributions to the study of sms spam filtering: new collection and results. In *Proceedings of the 11th ACM symposium on Document engineering*, pages 259–262.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. *arXiv preprint arXiv:2004.06025*.

Razvan Azamfirei, Sapna R Kudchadkar, and James Fackler. 2023. Large language models and the perils of their hallucinations. *Critical Care*, 27(1):1–2.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2023. Benchmarking large language models in retrieval-augmented generation. *arXiv preprint arXiv:2309.01431*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

Allan Peter Davis, Cynthia J Grondin, Robin J Johnson, Daniela Sciaky, Jolene Wiegers, Thomas C Wiegers, and Carolyn J Mattingly. 2021. Comparative toxicogenomics database (ctd): update 2021. *Nucleic acids research*, 49(D1):D1138–D1143.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ekaterina Fadeeva, Roman Vashurin, Akim Tsvigun, Artem Vazhentsev, Sergey Petrakov, Kirill Fedyanin, Daniil Vasilev, Elizaveta Goncharova, Alexander Panchenko, Maxim Panov, et al. 2023. Lm-polygraph: Uncertainty estimation for language models. *arXiv preprint arXiv:2311.07383*.

Daniel Fu, Mayee Chen, Frederic Sala, Sarah Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *International Conference on Machine Learning*, pages 3280–3291. PMLR.

Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan Naumann, Jianfeng Gao, and Hoifung Poon. 2020. Domain-specific language model pretraining for biomedical natural language processing.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR.

Changho Han, Dong Won Kim, Songsoo Kim, Seng Chan You, Jin Young Park, SungA Bae, and

Dukyong Yoon. 2023. Evaluation of gpt-4 for 10-year cardiovascular risk prediction: insights from the uk biobank and koges data. *iScience*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid O Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2019. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. *arXiv preprint arXiv:1911.10422*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 541–550.

C-L Hwang and Abu Syed Md Masud. 2012. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Martin Krallinger, Obdulia Rabal, Saber A Akhondi, Martın Pérez Pérez, Jesús Santamaría, Gael Pérez Rodríguez, Georgios Tsatsaronis, Ander Intxaurrondo, José Antonio López, Umesh Nandal, et al. 2017. Overview of the biocreative vi chemical-protein interaction track. In *Proceedings of the sixth BioCreative challenge evaluation workshop*, volume 1, pages 141–146.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Hunter Lang and Hoifung Poon. 2021. Self-supervised self-supervision by combining deep learning and probabilistic logic. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4978–4986.

Jiao Li, Yueping Sun, Robin J Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J Mattingly, Thomas C Wiegers, and Zhiyong Lu. 2016. Biocreative v cdr task corpus: a resource for chemical disease relation extraction. *Database*, 2016.

Zhen Lin, Shubhendu Trivedi, and Jimeng Sun. 2023. Generating with confidence: Uncertainty quantification for black-box large language models. *arXiv preprint arXiv:2305.19187*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Steven Moore, Huy A Nguyen, Tianying Chen, and John Stamper. 2023. Assessing the quality of multiple-choice questions using gpt-4 and rule-based methods. In *European Conference on Technology Enhanced Learning*, pages 229–245. Springer.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29.

Ha-Thanh Nguyen. 2023. A brief report on lawgpt 1.0: A virtual legal assistant based on gpt-3. *arXiv preprint arXiv:2302.05729*.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. 2023. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Vilfredo Pareto. 1964. *Cours d'économie politique*, volume 1. Librairie Droz.

John Platt et al. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Alexander Ratner, Braden Hancock, Jared Dunnmon, Frederic Sala, Shreyash Pandey, and Christopher Ré. 2019. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4763–4771.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.

Vikas C Raykar, Shipeng Yu, Linda H Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. 2010. Learning from crowds. *Journal of machine learning research*, 11(4).

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. 2021. End-to-end weak supervision. *Advances in Neural Information Processing Systems*, 34:1845–1857.

Yat-Fung Shea, Cynthia Min Yao Lee, Whitney Chin Tung Ip, Dik Wai Anderson Luk, and Stephanie Sze Wing Wong. 2023. Use of gpt-4 to analyze medical records of patients with extensive investigations and delayed diagnosis. *JAMA Network Open*, 6(8):e2325000–e2325000.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Paroma Varma, Frederic Sala, Shiori Sagawa, Jason Fries, Daniel Fu, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, et al. 2019. Multi-resolution weak supervision for sequential data. *Advances in Neural Information Processing Systems*, 32.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, Yidong Wang, Linyi Yang, Jindong Wang, Xing Xie, Zheng Zhang, and Yue Zhang. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity.

Hai Wang and Hoifung Poon. 2018. Deep probabilistic logic: A unifying framework for indirect supervision. *arXiv preprint arXiv:1808.08485*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575.

Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. 2023. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance.

Yuxin Xiao, Zecheng Zhang, Yuning Mao, Carl Yang, and Jiawei Han. 2021. Sais: supervising and augmenting intermediate steps for document-level relation extraction. *arXiv preprint arXiv:2109.12093*.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077.

Bianca Zadrozny and Charles Elkan. 2002. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699.

Jieyu Zhang, Cheng-Yu Hsieh, Yue Yu, Chao Zhang, and Alexander Ratner. 2022. A survey on programmatic weak supervision. *arXiv preprint arXiv:2202.05433*.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, pages 12697–12706. PMLR.

Wenxuan Zhou, Hongtao Lin, Bill Yuchen Lin, Ziqi Wang, Junyi Du, Leonardo Neves, and Xiang Ren. 2020. Nero: A neural rule grounding framework for label-efficient relation extraction. In *Proceedings of The Web Conference 2020*, pages 2166–2176.

A Proof of Theorems

A.1 Proof of Theorem 1

Proof. For convenience, let's denote

$$u_j(h) := \mathbb{E}[\ell_j(h(x), \lambda_j(x))], \quad j = 0, 1, \cdots, m.$$

We first show that any h^* minimizing $G(u_0, u_1, \cdots, u_m)$ is Pareto optimal.

Proof by contradiction. Suppose h^* is not Pareto optimal. Then there must exist some $h' \in \mathcal{H}$ Pareto dominating h^*. Without loss of generality, let's assume $u_j(h') < u_j(h^*)$, and $u_k(h') \le u_k(h^*)$, $\forall k \neq j$. Then according to Definition 2 of Pareto aggregator,

$$G(u_0(h'), \cdots, u_j(h'), \cdots, u_m(h')) \tag{6}$$
$$\le G(u_0(h^*), \cdots, u_j(h'), \cdots, u_m(h^*)) \tag{7}$$
$$< G(u_0(h^*), \cdots, u_j(h^*), \cdots, u_m(h^*)), \tag{8}$$

which contradicts the assumption that h^* is the minimizer for

$$G(u_0(h), \cdots, u_j(h), \cdots, u_m(h)).$$

Therefore, the original statement is true, and minimizing the objective

$$\min_{h \in \mathcal{H}} \quad G(\mathbb{E}\ell_0(h), \cdots, \mathbb{E}\ell_j(h), \cdots, \mathbb{E}\ell_m(h)) \tag{9}$$

gives a Pareto optimum.

Next, we use Jensen's inequality to upperbound this objective with the objective in problem 4. Using the fact that G is convex, we apply Jensen's inequality and get

$$G(\mathbb{E}\ell_0(h), \cdots, \mathbb{E}\ell_j(h), \cdots, \mathbb{E}\ell_m(h)) \tag{10}$$
$$\le \quad \mathbb{E}[G(\ell_0(h), \cdots, \ell_j(h), \cdots, \ell_m(h))]. \tag{11}$$

Therefore, solving the problem in Equation 4 approximates Pareto optimal harmonizer by upperbounding Equation 9. □

B Weights for Rebalancing the Sources

In our experiments, we explored four different types of scalarization functions, namely:

- Linear aggregator: $G(\ell_0, \ell_1, \cdots, \ell_m) := \sum_{j=0}^{m} w_j \ell_j$.
- Quadratic aggregator: $G(\ell_0, \ell_1, \cdots, \ell_m) := \left(\sum_{j=0}^{m} w_j \ell_j\right)^2$.
- Euclidean norm aggregator: $G(\ell_0, \ell_1, \cdots, \ell_m) := \| (w_0\ell_0, w_1\ell_1, \cdots, w_m\ell_m) \|$.
- Chebyshev aggregator: $G(\ell_0, \ell_1, \cdots, \ell_m) := \max_{j=0}^{m} w_j \ell_j$.

The weights $w_j \in \mathbb{R}$ are parameters of G. In the main text of the paper, we fixed to equal weights $\vec{w} = \vec{1}$. Here we introduce three approaches to determine the weighting if necessary.

Equal Weight The simplest weighting scheme of

$$w_0 = w_1 = \cdots = w_m = \frac{1}{m+1}$$

gives nice performance in practice, and is the method we used for the results in the main body of the paper. The nonlinear Pareto aggregators have the ability to balance the sources even under equal weights. Practically, We recommend starting with equal weight.

In the case that the supervision sources are highly correlated, or when the quality of the sources varies a lot, we propose the following two approaches utilizing the correlation in prediction residual.

Maximal Eigenvalue of Residual Correlation Suppose we have a pilot harmonizer $h_0 \in \mathcal{H}$, which can usually be obtained from minimizing a Pareto aggregator with equal weight, it gives predicted distribution $\vec{p}(x) \in \mathbb{R}^{|\mathcal{Y}|}$ for any input $x \in \mathcal{X}$, where

$$p_c(x) := \mathbb{P}(h_0(x) = c).$$

For any source $0 \le j \le m$, denote the one-hot vector $\vec{\lambda}_j(x) \in \mathbb{R}^{|\mathcal{Y}|}$ as:

$$\lambda_{j,c}(x) = \begin{cases} 1 & \text{if } \lambda_j(x) = c, \\ 0 & \text{otherwise.} \end{cases}$$

The prediction residual is defined as

$$\vec{r}_j(x) := \vec{\lambda}_j(x) - \vec{p}(x),$$

which accounts for the information source label for x that is unexplained by the harmonizer $h_0(x)$.

In order to rebalance the sources, we consider the correlation matrix C between the prediction residual $\vec{r}_j$'s. Specifically, let the covariance be

$$\Sigma_{ij} := \mathbb{E}_{x\sim\mathcal{X}}[\vec{r}_i(x) \cdot \vec{r}_j(x)] - \mathbb{E}_{x\sim\mathcal{X}}[\vec{r}_i(x)] \cdot \mathbb{E}_{x\sim\mathcal{X}}[\vec{r}_j(x)].$$

The correlation variance is denoted as

$$C_{ij} = \frac{\Sigma_{ij}}{\sqrt{\Sigma_{ii}\Sigma_{jj}}}.$$

We rebalance the sources according to the eigenvector $\vec{v}_{max} \in \mathbb{R}^{m+1}$ corresponding to the largest eigenvalue of C. In order to get reasonable weights, we first normalize $\vec{v}_{max}$ such that the sum of the entries equals to one. Then we project $\vec{v}_{max}$ to the weights simplex with minimal values $\frac{\epsilon}{m+1}$:

$$w_{excess,j} = \left(\vec{v}_{max,j} - \frac{\epsilon}{m+1}\right)^{+}$$
$$\vec{w}_{max} = \frac{\epsilon}{m+1} + \frac{w_{excess}}{\|w_{excess}\|_1} \cdot (1-\epsilon).$$

This projection ensures that the weight on each source is at least ϵ portion of the value from equal weight, with the minimal ratio threshold $\epsilon \in [0, 1]$.

Maximal eigenvalue method is recommended when the sources are relatively independent, and when the quality of the sources differ a lot. Intuitively, suppose two sources tends to agree with each other when they are not fitted well by the harmonizer, because there isn't intrinsic dependency between the sources, it is likely that the true label is given by the sources. Therefore, a maximal eigenvalue rebalancing scheme puts higher weights on the sources to encourage the harmonizer to fit to the unexplained examples.

Minimal Variance of Residual Correlation The same as in the maximal eigenvalue method, we consider the correlation matrix C between the prediction residual $\vec{r}_j$'s. Instead of finding the maximal eigenvalue of C, we consider solving the following minimal variance problem:

$$\min_{v} v^T C v, \quad \text{s.t. } \mathbf{1}^T v = 1.$$

This problem admits the closed form solution of

$$v_{min} = \frac{C^{-1}\mathbf{1}}{\mathbf{1}^T C^{-1}\mathbf{1}}.$$

Again, we project $\vec{v}_{min}$ to the weights simplex with minimal values $\frac{\epsilon}{m+1}$:

$$w_{excess,j} = \left(\vec{v}_{min,j} - \frac{\epsilon}{m+1}\right)^{+}$$
$$\vec{w}_{min} = \frac{\epsilon}{m+1} + \frac{w_{excess}}{\|w_{excess}\|_1} \cdot (1-\epsilon),$$

which ensures that the weight on each source is at least ϵ portion of the value from equal weight, with the minimal ratio threshold $\epsilon \in [0, 1]$.

The minimal variance method is a classical portfolio rebalancing strategy in financial mathematics. The intuition behind the algorithm is minimizing the risk by diversification. This rebalancing scheme is useful when there are intrinsic dependency between the sources. Suppose two sources are duplicates and always tend to give the same label, their residuals should also be highly correlated. Minimal variance optimization automatically avoid putting too much weights on the duplicating sources.

While the equal weight method typically delivers good results in the simplest way, the other two rebalancing schemes are designed to address the specific concern such as source dependency and quality. For best performance, we recommend checking against the labels on a validation set if available.

C Training details

We explored different configurations of Pareto optimal learning below:

- Harmonizer model: we experiment 1. BERT (Devlin et al., 2018) (PubMedBERT (Gu et al., 2020) for biomedical datasets CDR and ChemProt), 2. multi-layer perceptron (MLP), 3. Logistic regression (LR). The last two are built on top of the last layer embedding of the corresponding BERT model.

- Pareto loss scalarizer: we experiment all four loss scalarization functions as defined in Section 3.2, namely linear, quadratic, Euclidean norm, and Chebyshevy scalarization.

- Optimizer: We use AdamW (Loshchilov and Hutter, 2017) optimizer with learning rate $[10^{-4}, 10^{-5}, 10^{-6}]$, weight decay $[10^{-4}, 10^{-5}]$, batch size 16. All hyperparameters are optimized on held out dev set.

- Computation: We trained on Azure Standard NC12s v3 with 1 Nvidia V100 GPU.

D LLM Prompting Details

In this section we will describe the details of the prompts used to query the LLMs.

D.1 Out-of-the-box prompt

- Setting: describe the role of the LLM in the task, and the overall problem setting.
- Background: necessary background knowledge for domain specific tasks, including information from annotation guidelines for human annotators.
- Data structure: for relation extraction tasks, explain the definition of each entity.
- Desired output: describe the list of the desired output. For each of the categories, provide explanation and optionally some examples.
- Chain of thought (CoT): instruction to encourage the LLM to think step-by-step, articulate point-by-point, and give the response in the desired structure.
- Confidence: ask the model to state “unsure” if it is not confident about the answer.
- Example: state the example and ask the model to perform the task.

Each prompt for out-of-the-box (zero-shot) prediction contains:

- A problem setting part that depends on the specific dataset.
- A response regularization part that encourages chain-of-thought (CoT) and confidence check, and specifies proper response format.
- A task instance part that contains the input instance and restates the task to perform.

Problem setting prompt

- CDR: “You are an intelligent assistant to extract chemical-disease relations from academic literature. Your job is to determine if in the given piece of text, the drug (entity 1) induces the disease (entity 2) or not. Negative means the drug does NOT induce the disease. Positive means the drug induces the disease. Please use your judgement to the best of your knowledge. Your answer should be classified into the following categories: [Negative, Positive]. ”

- ChemProt: “You are an intelligent assistant to extract chemical-protein interaction from academic literature. Your task is to identify the chemical-protein interactions (CHEMPROT) between entity 2: Chemical Entities Mentions (CEMs) and entity 1: Gene and Protein Related Objects (named as GPRO in the instruction below) in the given piece of text. In brief, the chemical-protein interactions include direct interactions (when a physical contact exits between a CEM and a GPRO, in most cases this GPRO being a protein or protein family and alters its function/activity) as well as indirect regulatory interactions between CEMs and GPROs (including genes, gene products (proteins, RNA), DNA/protein sequence elements and protein families, domains and complexes) that alter either the function or the quantity of the GPRO. The guidelines below provide curation rules to evaluate if the given sentence contains a description of a chemical-protein interaction; in particular, if sufficient detail/evidence is provided for comentioned CEMs and GPROs. Additionally, it describes curation rules and definitions to assign each identified chemical-protein interaction to any of the 10 classes, with detailed description listed below:

  0. Part of: CEM that are structurally related to a GPRO: e.g. specific amino acid residues of a protein.

  1. Regulator: CEM that clearly regulates a GPRO, but for which there is no further information on whether the regulation is direct or indirect.

  2. Upregulator: CEM that increments a GPRO signal, without any insight on the mechanism.

  3. Downregulator: CEM that decreases a GPRO signal, without any insight on the mechanism.

  4. Agonist: CEM that binds to a receptor and alters the receptor state resulting in a biological response.

  5. Antagonist: CEM that reduces the action of another CEM, generally an agonist. Many antagonists act at the same

receptor macromolecule as the agonist.

6. Modulator: CEM that acts as allosteric modulator, compound that increases or decreases the action of an (primary or orthosteric) agonist or antagonist by combining with a distinct (allosteric or allotropic) site on the receptor macromolecule.

7. Cofactor: CEM that is required for a protein's biological activity to happen.

8. Substrate/Product: CEM that is both, substrate and product of enzymatic reaction.

9. NOT: This class should be used to define the NEGATIVE occurrence of a chemical-protein interaction, without providing any further information on the specific negative CHEMPROT class or class.

Please identity the CHEMPROT interaction to the best of your knowledge. Your answer should be classified into the following categories: [Part of, Regulator, Upregulator, Downregulator, Agonist, Antagonist, Modulator, Cofactor, Substrate/Product, NOT]. "

- SemEval: "You are an intelligent assistant to help recognize semantic relations between pairs of nomimals. For example, tea and ginseng are in an ENTITY-ORIGIN relation in "The cup contained tea from dried ginseng.". You will be given a piece of text, and Entity 1 and Entity 2 in the text for you to classify their semantic relation. The semantic relations are in the format of "entity1-entity2". The complete semantic relation inventory is given below:

  0. Cause-Effect: An event or object (entity 1) leads to an effect (entity 2). Example: those cancers (entity 2) were caused by radiation exposures (entity 1)

  1. Component-Whole: An object (entity 1) is a component of a larger whole (entity 2). Example: my apartment (entity 2) has a large kitchen (entity 1)

  2. Content-Container: An object (entity 1) is physically stored in a delineated area of space (entity 2). Example: a bottle (entity 2) full of honey (entity 1) was weighed

  3. Entity-Destination: An entity (entity 1) is moving towards a destination (entity 2). Example: the boy (entity 1) went to bed (entity 2)

  4. Entity-Origin: An entity (entity 1) is coming or is derived from an origin (entity 2) (e.g., position or material). Example: letters (entity 1) from foreign countries (entity 2)

  5. Instrument-Agency: An agent (entity 2) uses an instrument (entity 1). Example: phone (entity 1) operator (entity 2)

  6. Member-Collection: A member (entity 1) forms a nonfunctional part of a collection (entity 2). Example: there are many trees (entity 1) in the forest (entity 2)

  7. Message-Topic: A message (entity 1), written or spoken, is about a topic (entity 2). Example: the lecture (entity 1) was about semantics (entity 2)

  8. Product-Producer: A producer (entity 2) causes a product (entity 1) to exist. Example: a factory (entity 2) manufactures suits (entity 1)

  Please determine the semantic relation between entity 1 and entity 2 in the given text to the best of your knowledge. Your answer should be classified into the following categories: [Cause-Effect, Component-Whole, Content-Container, Entity-Destination, Entity-Origin, Instrument-Agency, Member-Collection, Message-Topic, Product-Producer]. "

- SMS: "You are an intelligent assistant to determine if a text message is spam or not spam (ham). Your answer should be classified into the following categories: [ham, spam]. "

Response regularization prompt "You may think step by step, articulate point by point, or make conclusion from multiple evidences, but please always state the most likely label as your answer at the very begining of your response. You are encouraged to reflect on your response, but please keep in mind that a clear answer is always desired. Try to give a clear answer at your best guess even when you are not very sure, in which case any of your conserns or explanations should go after the most likely answer to the best of your knowledge.

If you are very unsure about the answer and are not willing to explicitly state any label, please say 'unsure' at the very begining of your response. "

Task instance prompt

- Classification (for SMS):

  "Please classify the following example into the most likely category: [TEXT] "

- Relation extraction (for CDR, ChemProt, SemEval):

  "Please classify the following example into the most likely category: [TEXT] Entity 1 [ENTITY 1] Entity 2: [ENTITY 2] "

The complete prompt for querying the LLM is

$$\textbf{Problem setting prompt} \\ +\textbf{Response regularization prompt} \\ +\textbf{Task instance prompt}$$

D.2 Dynamic prompting

In dynamic prompting, we query another follow-up prompt after the LLM gives the initial out-of-the-box response. As this is an extension to our main experiments, we only implemented for the CDR relation extraction task. The follow-up prompts for the two dynamic prompting strategies are:

Dynamic self-verification "Are you sure about your previous answer? If not, please give a new answer. Otherwise, please restate your previous answer. "

POLAR-assisted RAG "It is possible that the answer could be something else. Here are some evidences to help you figure out the right answer.

$$\text{InformationSourceRetrival}(x,\ \vec{\lambda}(x))$$

Are you sure about your previous answer? If not, please give a new answer. Otherwise, please restate your previous answer. "

InformationSourceRetrival$(x,\ \vec{\lambda}(x))$ contains evidences from all the information sources $\lambda_j(x) \neq 0$ that are triggered by the input instance x. Examples of evidence from the information sources are shown below. Note that each evidence will be provided only when the corresponding information source is triggered.

- "According to the Comparative Toxicogenomics Database, the relation between the given chemical-condition pair is listed, confirming the answer. "
- "According to the Comparative Toxicogenomics Database, the given chemical-condition pair "[ENTITY 1]-[ENTITY 2]" is listed that the chemical actually treats the condition, so the answer that [ENTITY 1] does not induce [ENTITY 2] is confirmed. "
- "According to the Comparative Toxicogenomics Database, the given chemical-condition pair "[ENTITY 1]-[ENTITY 2]" is listed that the chemical is typically present with the condition, which may confirm the answer if [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [INDUCE PATTERN], it is likely that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [NOT INDUCE PATTERN], it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [C TREATS D PATTERN], [ENTITY 1] actually treats [ENTITY 2]. , so it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [CLOSE MENTION PATTERN], [ENTITY 1] is closely mentioned with [ENTITY 2], so they should be closely related. "
- "Based on the expression [DISEASE IMPROVE PATTERN], the disease [ENTITY 2] is actually improved, so it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [INITIAL CONDITION PATTERN], [ENTITY 2] is the initial condition of the patient(s), so it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [UNCERTAIN PATTERN], it is uncertain that [ENTITY 1] induces [ENTITY 2]. "
- "Based on the expression [INDUCED BY OTHER PATTERN], [ENTITY 2] is induced by other factors, so it is not likely that [ENTITY 1] induces [ENTITY 2]. "

- "[ENTITY 1] and [ENTITY 2] are not closely mentioned in the text, so it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "According to phrases like [WEAK EXPRESSION], there is no strong signal that [ENTITY 1] induces [ENTITY 2]. "
- "According to the text, another chemical is mentioned closer to [ENTITY 2] than [ENTITY 1], so it is not likely that [ENTITY 1] induces [ENTITY 2]. "
- "According to the text, another disease is mentioned closer to [ENTITY 1] than [ENTITY 2], so it is not likely that [ENTITY 1] induces [ENTITY 2]. "