Who Knows the Answer? Finding the Best Model and Prompt for Each Query Using Confidence-Based Search

Walter Gerych[1*], Yara Rizk[2], Vatche Isahagian[2], Vinod Muthusamy[2], Evelyn Duesterwald[2], Praveen Venkateswaran[2]

[1] MIT CSAIL
[2] IBM Research
wgerych@mit.edu, yara.rizk@ibm.com, vatchei@ibm.com, vmuthus@us.ibm.com, duester@us.ibm.com, praveen.venkateswaran@ibm.com

Abstract

There are increasingly many large language models (LLMs) available to the public. While these LLMs have exhibited impressive abilities on a variety of task, any individual LLM in particular may do well on some tasks and worse on others. Additionally, the performance of these models is heavily dependent on the choice of prompt template. For instance, they exhibit sensitivity to the few shot examples chosen or brittleness to the wording of instructions. Moreover, a prompt template that makes a model perform well for one input may not be the optimal template for another input. This necessitates an approach for adaptively selecting LLM and prompt template pairs for each input. Recent work has shown that the accuracy of an LLM's responses is correlated with the LLM's confidence in the response. Thus, a natural choice for selecting which model and prompt template to use is to select the pair that is most confident in its response. However, existing confidence metrics are expensive to calculate, necessitating multiple calls to each LLM and prompt pair. We thus propose an approach to predict the confidence of each pair using an auxiliary regression model that is inexpensive to run. Using this auxiliary model, we select the LLM and prompt template with the highest predicted confidence for a given input. Results on a range of benchmark datasets show that our confidence-based instance-level prompt search method consistently improves the performance of LLMs.

Introduction

Large language models (LLMs) have demonstrated impressive performance on a range of tasks such as question answering (Robinson, Rytting, and Wingate 2022), text summarization (Zhang et al. 2023), and search (Spatharioti et al. 2023). While some leading LLMs such as `GPT-4` (OpenAI 2023) show good performance on a wide variety of tasks, these models are closed-source. Fortunately, many open-source language models, such as `FLAN-T5` (Chung et al. 2022) and `BLOOM` (Scao et al. 2022), have been made freely available to the community. These open source models generally perform well on some tasks, or for some inputs, and poorly on others (Jiang, Ren, and Lin 2023). Additionally, the performance of LLMs in general has been shown to be highly sensitive to the prompt template or instructions that they are conditioned on (Zhao et al. 2021). These facts combine to lead to the central question that is the focus of this work: **Given an input for a user, how can we determine which LLM and which prompt template to use?**

A reasonable method for choosing which LLM and prompt template to use is to select the conditioned model that is most *confident* in its response. Ideally, we could determine the confidence of each model and prompt template pair and then select a return from the most confident pair. Recent works have proposed methods for measuring the confidence of LLMs (Kadavath et al. 2022; Lin, Hilton, and Evans 2022; Kuhn, Gal, and Farquhar 2023; Lin, Trivedi, and Sun 2023), using techniques such as measuring the semantic similarity between multiple samples from the model (Kuhn, Gal, and Farquhar 2023; Lin, Trivedi, and Sun 2023). However, existing confidence approaches require obtaining at least one (and often many) response from a model before confidence can be estimated. This is not feasible to do when choosing between multiple model and prompt template options; calculating the confidence of all pairs would incur a quadratic number of LLM calls for each user input. This is an unreasonable expense, in terms of time and compute resources.

In this paper, we tackle the problem of selecting the most confident LLM and prompt template pair using an estimated confidence that does *not* require observing the output of each model during inference. More specifically, we assume that we have access to (potentially unlabeled) datasets of examples for a set of tasks that we can compute model confidence on in an offline setting. Using this data, we aim to predict what the model confidences would be when given new instances from similar tasks.

There are several challenges that must be overcome to achieve this. First, we require that the confidence prediction must be inexpensive to calculate - as we need to make this confidence prediction for each LLM-prompt template pair for each input from the user. Second, the confidence calculation can not leverage the response from the model itself - again due to computational expense. Third, it is not trivial to compute confidence given only the input to the LLM - a natural choice, given that the output is not accessible. More specifically, the standard ways of computing features

[*] Walter performed this work during his internship at IBM Research.

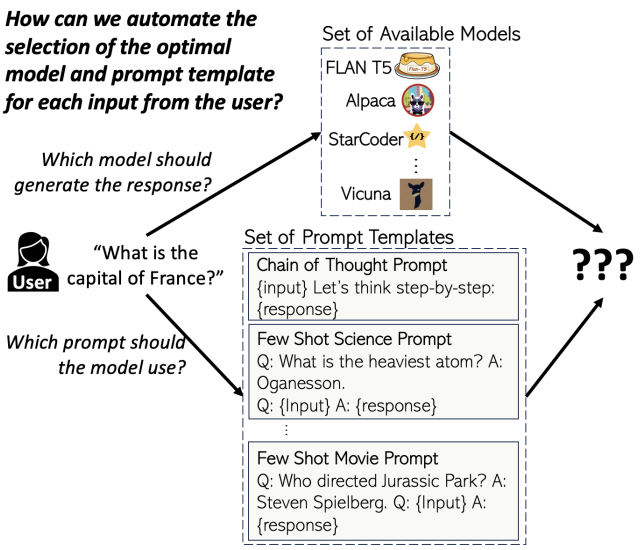

Figure 1: Given a set of LLMs and a set of prompt templates, how can we select the optimal LLM-prompt template pair for any given user input?

from variable length text - such as utilizing a text embedding from a Sentence Transformer (Reimers and Gurevych 2019) - result in high dimensional vectors. This leads to a sparse vector space unless incredibly large training sets are available, making these standard features inadequate for learning a mapping from input to confidence.

Hence, we propose Predicted-Instance-Confidence Search (PICS), a system for selecting a response from the LLM-prompt template pair that has maximum confidence. At a glance, PICS first precomputes the confidences of each model on the available training datasets. Then, training data is initially featurized by using a contextual text embedding model (Reimers and Gurevych 2019). Next, each datapoint is given a final feature representation based on the confidences and distances of its nearest neighbor points in the initial embedding space. This final feature representation is used to train an ensemble of small machine learning regression models to predict the confidence of each LLM and prompt template for each point in the available datasets. Then, when a new instance arrives, the regression models are used to obtain a predicted confidence for each LLM-prompt template pair. Running these regressors incurs minimal cost compared to the LLMs. Next, the LLM and prompt template pair with the maximum predicted confidence is selected. Finally, a set of responses are sampled from the selected pair, and the most confident response is returned.

Contributions. Our contributions are as follows.

- To the best of our knowledge, we are the first to propose selecting prompt templates based on model confidences.
- We are the first work to propose using predicted confidence from an auxiliary model for estimating LLM confidences.
- We propose a new featurization method for confidence regressors, based on neighbor confidence and similarity in an LLM embedding space.
- We propose returning the most confident response from a sample of model outputs, as opposed to returning a greedy response or the output of a beam search.

Problem Definition

Assumptions. Assume we are given access to a set of d_m LLMs, $\mathcal{M} = \{\mathbf{M}_i\}_{i=1}^{d_m}$, where $\mathbf{M}$ is an LLM. Additionally, we have a set of d_p *prompt templates*, $\mathcal{P} = \{\mathbf{P}_i\}_{i=1}^{d_p}$. For a given input query x, a formatted prompt is given by $\mathbf{P}(x)$. For instance, $\mathbf{P}$ may be a chain of thought template (Wei et al. 2022) such as `"{input}. Let's think it through {response}"`, where $\mathbf{P}(x)$ replaces `"{input}"` with the text of x. We also have access to n_D training datasets, $\mathscr{D} = \{\mathcal{D}_v\}_{v=1}^{n_D}$, $\mathcal{D}_v = \{d_i^v\}_{i=1}$, where d_i^v is the ith text in dataset $\mathcal{D}_v$.

Let $R : \mathscr{D} \times \mathcal{P} \rightarrow \{0, 1\}$ be a binary function, such that $R(\mathcal{D}, \mathbf{P}) = 1$ if prompt template $\mathbf{P}$ is "related" or "paired" with dataset $\mathcal{D}$, and returns 0 otherwise. The concept of relatedness is up to the user; for instance, a prompt template is "related" to a dataset if the prompt template uses few-shot examples from the dataset. Or, for example, $\mathcal{D}$ may be a dataset of QA examples and the prompt template may be an instruction designed to condition the LLM for question answering.

Defining model response. For a given input x, a response r from model $\mathbf{M}_i$ using prompt template $\mathbf{P}_j$ is sampled according to $r \sim P_{\mathbf{M}_i|\mathbf{P}_j(x)}$. Here, $P_{\mathbf{M}_i|\mathbf{P}_j(x)}$ is the conditional distribution parameterized by $\mathbf{M}_i(\mathbf{P}_j(x))$.

Goal. Our goal is to optimize the following equation independently for each input x:

$$\underset{\mathbf{M}\in\mathcal{M},\mathbf{P}\in\mathcal{P}}{\operatorname{argmax}} \; \mathbb{E}_{r\sim P_{\mathbf{M}|\mathbf{P}(x)}} conf(r|\mathbf{M}, \mathbf{P}), \tag{1}$$

where $conf()$ is a confidence measure, $conf : \mathcal{T} \rightarrow \mathbb{R}$, that assigns a real-valued confidence score to points in the space $\mathcal{T}$ of the LLM outputs. We provide a discussion on the choice of the confidence function in the following section.

Notably, we aim to optimize for Equation 1 *without* obtaining responses r for each $\mathbf{M} \in \mathcal{M}$ and each $\mathbf{P} \in \mathcal{P}$.

Methodology

The key idea to our proposed Predicted-Instance-Confidence Search is that for any given input x we *predict* the confidence of each LLM and prompt template pair, and then select the most confident response from the pair with the highest predicted confidence. We explain this approach in detail below.

Semantic Confidence

There are many ways to define the confidence score $conf(\cdot|\mathbf{M}, \mathbf{P})$ of an LLM conditioned on a prompt. For instance, we could prompt the model to express a verbalized confidence (Xiong et al. 2023), or estimate the number of distinct *semantic groups* (the number of semantically different responses) (Kuhn, Gal, and Farquhar 2023; Lin, Trivedi, and Sun 2023).

In this work, we utilize the *similarity vector*[1] formed from the similarity graph of a set of sampled LLM responses, as proposed by (Lin, Trivedi, and Sun 2023). More specifically, for a given input x, LLM $\mathbf{M}$ and prompt template $\mathbf{P}$ we first sample N responses from the conditioned LLM using a temperature τ.

Next, we compute the *similarity score* between each response. Let $s(r_i, r_j)$ be the similarity score between the ith and jth sampled responses. The functional form of $s(\cdot, \cdot)$ is a design choice; no matter the choice, it should be a score that is high for instances that are semantically similar (e.g., maximized for sentences with the same meaning) and low otherwise. For instance, it could be the score returned from an NLI-tuned LLM as initially proposed (Lin, Trivedi, and Sun 2023), a cosine similarity between the scores in an embedding space, or the RougeL score (Lin 2004) between the sentences. We chose RougeL in our experiments.

We then compute the similarity vector $\mathbf{s^{M,P,x}}$, where $\mathbf{s_i^{M,P,x}} = \sum_{\mathbf{j=1, j\neq i}}^{\mathbf{N}} \mathbf{s(r_i, r_j)}$. $\mathbf{s_i^{M,P,x}}$ is thus a vector where the ith entry is the sum of the similarities between the ith response and all other sampled responses.

Finally, the confidence score $conf(r_i|\mathbf{M}, \mathbf{P})$ is defined as follows:

$$conf(r_i|\mathbf{M}, \mathbf{P}) = \mathbf{s_i^{M,P,x}}/\mathbf{N}$$

Intuitively, the confidence of the LLM in its response r_i is high if r_i is similar to the other sampled responses from that model for the same input. This is because the similarity will be high when the model is producing semantically similar responses, indicating it is certain of its response.

Unfortunately, since this confidence is computed on the *output* of each LLM and prompt template pair - in fact, N responses from each pair - using this confidence directly for searching for the best pair to respond to each input x is too costly. Instead, we must develop a way of utilizing confidence *without* sampling from each pair during runtime.

Estimating the Confidence of LLM Responses

As computing $conf(r|\mathbf{M}, \mathbf{P})$ for each $\mathbf{M}$ and $\mathbf{P}$ is too costly for online inference, we propose to replace $conf(\cdot|\mathbf{M}, \mathbf{P})$ with an estimated confidence score $C^*_{|\mathbf{M},\mathbf{P}}(\cdot)$. This estimated confidence is computed using regression models trained to match the true confidence score.

The first design consideration we make when designing the estimated confidence is on what the input space of $C^*_{|\mathbf{M},\mathbf{P}}(\cdot)$ should be. Recall that $conf(\cdot|\mathbf{M}, \mathbf{P})$ is calculated using the response from the LLM; however, we cannot obtain this response without making a call to the LLM. Instead, we compute our estimated confidence using features of the *input* to the LLM: Given an input x that we want a response to, we estimate the confidence of $\mathbf{M}(\mathbf{P})$ using $C^*_{|\mathbf{M},\mathbf{P}}(f_{\mathbf{M},\mathbf{P}}(x))$, where $f_{\mathbf{M},\mathbf{P}}(x)$ is a featurized vector representation of x.

More specifically, we obtain an *initial* representation of x by first transforming x - which is in general a textual input of varying length - into a constant -size vector embedding $h(x)$. We utilized the average representation from `FLAN-T5-XL` (Chung et al. 2022) for our experiments. However, vectors in the embedding space $\mathcal{H}$ are not ideal representations for learning our confidence regressor $C^*_{|\mathbf{M},\mathbf{P}}(\cdot)$. $\mathcal{H}$ is not well suited to our needs as these embedding spaces are typically high dimensional; for instance, the embedding space from `FLAN-T5-XL` has over 1000 dimensions. We would need very large datasets to fit a confidence regressor on this space.

Rather than employing $h(x)$ as our featurized version of the input, we opt for computing a lower-dimensional representation that is specifically tailored for confidence prediction. First, we identify the k nearest neighbors of $h(x)$ in the embedding space $\mathcal{H}$, where these neighbors(representing the user inputs in the training data) are found using a search over the embedded texts from our training datasets $\{\mathcal{D}_v\}_{v=1}^{n_D}$. Let $d_k \in \bigcup_{v=1}^{n_D} \mathcal{D}_v$ be an instance in our training sets — e.g. an example input query, not a model's response — such that $h(d_k)$ is the instance with the kth highest cosine similarity to $h(x)$; then, we compute an intermediate representation:

$$g_{\mathbf{M},\mathbf{P}}(x) = \begin{bmatrix} conf(r(d_1)|\mathbf{M}, \mathbf{P}) \\ conf(r(d_2)|\mathbf{M}, \mathbf{P}) \\ \vdots \\ conf(r(d_k)|\mathbf{M}, \mathbf{P}) \end{bmatrix}^T,$$

where each neighbor confidence $conf(r(d_1)|\mathbf{M}, \mathbf{P})$ is precomputed in an offline setting. Specifically, we compute the confidences for all training data once, so this does not add to the runtime of our system once deployed.

Intuitively, this featurization allows the regressor to predict the confidence x based on a nonlinear combination of the confidences of similar queries. However, this representation $g_{\mathbf{M},\mathbf{P}}(x)$ alone does not provide any signal relating to *how* similar to the neighbor queries are to x; it could be that the k neighbors of $h(x)$ are very distant. Information on whether the neighbors are very similar (in which case their confidences should be more highly weighted) or distant (which implies their confidences are less predictive) is taken into account by a second intermediate representation $b(x)$, $b(x) = [\delta_x(d_1), \delta_x(d_2), \ldots, \delta_x(d_k)]$, where $\delta_x(d_k)$ is the cosine distance between $h(x)$ and its kth neighbor $h(d_k)$. We then obtain the final representation as:

$$f_{\mathbf{M},\mathbf{P}}(x) = g_{\mathbf{M},\mathbf{P}}(x) \oplus b(x).$$

Training the Confidence Regressors. Now that we have a representation $f_{\mathbf{M},\mathbf{P}}(x)$ that is tailored for confidence prediction, we can use this representation to obtain a predicted confidence. Specifically, we parameterize the predicted confidence $C^*_{|\mathbf{M},\mathbf{P}}(\cdot)$ using machine learning regression models. In our experiments, $C^*_{|\mathbf{M},\mathbf{P}}(\cdot)$ is given by an ensemble of linear regression, Random Forest (Ho 1995), and Gradient Boosting (Friedman 2001). The confidence regressor is fit by minimizing a loss L, given by:

$$L = \frac{1}{N} \sum_{d \in \bigcup_{v=1}^{n_D} \mathcal{D}_v} \left(\overline{conf}(r(d)|\mathbf{M}, \mathbf{P}) - C^*_{|\mathbf{M},\mathbf{P}}(f_{\mathbf{M},\mathbf{P}}(d)) \right)^2,$$

where $\overline{conf}(r(d)|\mathbf{M}, \mathbf{P})$ is the average confidence of a sample of responses from $\mathbf{M}(\mathbf{P}(x))$. Thus, when using the similarity vector $\mathbf{s_{i,i}^{M,P,d}}$ to define confidence, the average confidence is given by:

[1] Our similarity vector corresponds to the diagonal of the degree matrix in (Lin, Trivedi, and Sun 2023)

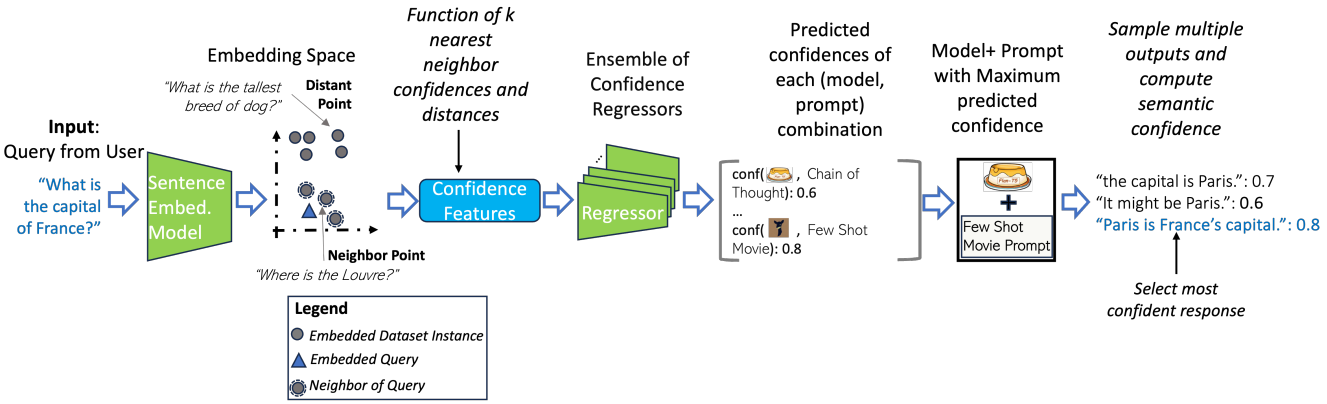

Figure 2: Overview of our proposed Predicted-Instance-Confidence Search pipeline.

$$\overline{conf}(r(d)|\mathbf{M},\mathbf{P}) = \sum_{i=1}^{N} \mathbf{s_i^{M,P,d}}/\mathbf{N^2}.$$

Predicted-Instance-Confidence Search

Now that we have described our approach for estimating confidences, we are equipped to describe our method for selecting the most confident LLM and prompt template pair with which to produce a response for each input x from the user. An overview of this pipeline is shown in Figure 2, and a pseudocode implementation of our proposed approach is given in Algorithm 1.

To begin, we once again utilize a text embedding model to produce the text's vector representation $h(x)$. We then select training datasets that have an average cosine similarity with $h(x)$ that is greater than a threshold α. Let $\mathscr{D}^{\alpha,x}$ be this subset, such that if $\mathcal{D} \in \mathscr{D}^{\alpha,x}$ then the average cosine similarity between $h(x)$ and the elements of $\mathcal{D}$ is above α.

We then select a subset of prompts $\mathcal{P}^{\alpha,x}$, such that $\mathbf{P} \in \mathcal{P}^{\alpha,x}$ iff $R(\mathcal{D},\mathbf{P})$ for some $\mathcal{D} \in \mathscr{D}^{\alpha,x}$. We filter the prompts in this way so that we estimate confidences only on settings where the corresponding regressors were trained on data similar to x.

This leads directly to the next step, which involves estimating the confidences for each LLM paired with each prompt template in the subset $\mathcal{P}^{\alpha,x}$. Thus, for each pair we featurize $h(x)$ as $f_{\mathbf{M},\mathbf{P}}(x)$ and compute the estimated confidence $C^*_{|\mathbf{M},\mathbf{P}}(x)$. We then select the LLM and prompt template with maximum predicted confidence as follows:

$$\mathbf{M}^*, \mathbf{P}^* = \underset{\mathbf{M}\in\mathcal{M},\mathbf{P}\in\mathcal{P}^{\alpha,x}}{\operatorname{argmax}} \widetilde{C}^*_{|\mathbf{M},\mathbf{P}}(x),$$

where $\mathbf{M}^*, \mathbf{P}^*$ is defined as:

$$\widetilde{C}^*_{|\mathbf{M},\mathbf{P}}(x) = \begin{cases} C^*_{|\mathbf{M},\mathbf{P}}(x) & \text{if } \eta > \gamma \\ \overline{C}_{|\mathbf{M},\mathbf{P}} & \text{otherwise} \end{cases}$$

In the equation above, η is the average cosine similarity of x's neighbor points in the embedding space, γ is a threshold, and $\overline{C}_{|\mathbf{M},\mathbf{P}}$ is the average confidence of $\mathbf{M}$ and $\mathbf{P}$. In our experiments we use $\gamma = 0.6$.

After obtaining the optimal model $\mathbf{M}^*$ and prompt template $\mathbf{P}^*$, we select a response $r*$. To this end, we sample n responses from $\mathbf{M}^*$ conditioned on $\mathbf{P}^*(x)$. We then compute the confidence of each sample, returning the response with maximum confidence. As this calculation needs to be done for only *one* model and prompt pair - i.e. the most confident pair - we opt to compute confidence using a semantic confidence approach rather than using a confidence regressor. Thus, given a set of n response $\{r_1, r_2, \ldots, r_n\}$, the final response r^* is found using:

$$r^* = \underset{r\in\{r_1,r_2,\ldots,r_n\}}{\operatorname{argmax}} conf(r|\mathbf{M}^*,\mathbf{P}^*).$$

Lastly, we return this final response r^* along with the model and prompt template pair with the highest confidence, $(\mathbf{M}^*, \mathbf{P}^*)$, to the user.

Algorithm 1: Predicted-Instance-Confidence Search

function PICS(x)
  Input: Query x, confidence regressors $C_{|\mathbf{M},\mathbf{P}}$ for $\mathbf{M} \in \mathcal{M}$ and $\mathbf{P} \in \mathcal{P}$, training dataset $\mathcal{D}$
  Output: Most confident response r^*, most confident model and prompt template pair $(\mathbf{M}^*, \mathbf{P}^*)$
  $f\_x \leftarrow confidence\_featurization(x, \mathcal{D})$
  $max\_conf \leftarrow 0$
  $\mathbf{M}^* \leftarrow \mathbf{M}_0$
  $\mathbf{P}^* \leftarrow \mathbf{P}_0$
  for $m \leftarrow 1$ to d_m **do**
    for $p \leftarrow 1$ to d_p **do**
      $\eta \leftarrow avg\_neighbor\_similarity(x, \mathcal{D})$
      if $\eta > \gamma$ **then**
        $predicted\_conf = C_{|\mathbf{M}_m,\mathbf{P}_p}(f\_x)$
      else
        $a \leftarrow avg\_conf(\mathbf{M}_m, \mathbf{P}_p)$
        $predicted\_conf \leftarrow a$
      if $predicted\_conf > best\_conf$ **then**
        $best\_conf \leftarrow predicted\_conf$
        $\mathbf{M}^* \leftarrow \mathbf{M}_m$
        $\mathbf{P}^* \leftarrow \mathbf{P}_p$
  $\mathbf{s}^{\mathbf{M}^*,\mathbf{P}^*,\mathbf{x}} \leftarrow \mathbf{similarity\_vector}(\mathbf{x}, \mathbf{M}^*, \mathbf{P}^*)$
  $i^* \leftarrow argmax\_i\ \mathbf{s_i^{M^*,P^*,x}}$
  $r^* \leftarrow \mathbf{s_{i^*}^{M^*,P^*,x}}$
  return $r^*, (\mathbf{M}^*, \mathbf{P}^*)$

Model	Number of Parameters	Instruction Tuned?
FLAN-T5-XL	11B	Yes
FLAN-UL2	20B	Yes
MPT-30B	30B	No
BLOOM	176B	No

Table 1: Models used when constructing model and prompt template pairs

Experiments

We now validate our proposed Predicted-Instance-Confidence Search (PICS) approach through a series of experimental evaluations.

Experimental Setup

We briefly describe our experimental setup below.

Compared Approaches

- **Average Performance of Model and Prompt Template Combinations (MPT Avg.)** We report the average accuracy over the set of valid model and prompt template combinations as "MPT Avg.".
- **Most Confident Model and Prompt Template Combination**. Instead of selecting the model and prompt template pair with the highest confidence individually for each instance, we could instead select the pair with the highest average confidence and use this for all inputs. We compare against this approach, which we refer to as "Most Confident".
- **Predicted-Instance-Confidence Search with Random Sampling (PICS-RS)** This is our proposed approach, but rather than the final semantic confidence calculation to select the most confident response out of a sampled set we instead simply sample a single response from the most confident model and prompt template pair.
- **Predicted-Instance-Confidence Search with Confidence Sampling (PICS-CS)** Our full proposed approach, including the final confidence sampling step.
- **Oracle Confidence Search with Random Sampling (OCS-RS)** This approach is similar to (PICS-RS), but instead of using the *predicted* confidence of each instance we instead use the calculated semantic confidence. This approach is too computationally expensive to be used in practice in most scenarios.
- **Oracle Confidence Search with Confidence Sampling (OCS-CS)** This approach is similar to (PICS-CS), but instead of using the *predicted* confidence of each instance we instead use the calculated semantic confidence. As was the case for OCS-RS, this approach is also too computationally expensive to be practically used in most scenarios.

Models We perform our analysis with each LLM shown in Table 1. We selected these models so as to have a mix of instruction tuned models and non-instruction tuned models, as well as models with a range of parameter counts (11B to 176B).

Datasets We used the following datasets in our analysis:

- **StrategyQA.** This dataset consists of input questions which require reasoning to answer, and the targets are responses that include the logical reasoning used to respond to the question (Geva et al. 2021).
- **TriviaQA**. The inputs in this dataset are trivia questions, and the ground truth responses are short factual answers (Joshi et al. 2017).

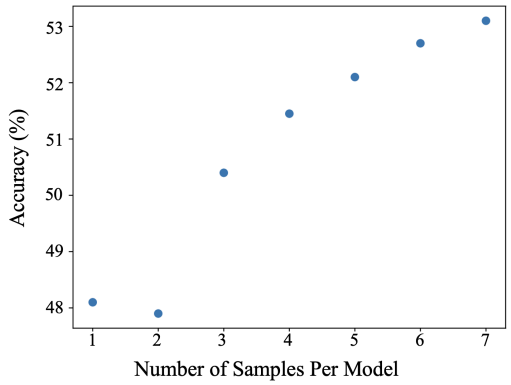

Figure 3: Accuracy vs. Number of Samples used in Confidence Sampling

- **Question Formation.** This dataset is taken from the *linguistic mapping*[2] Big Bench dataset (Srivastava et al. 2023). The inputs from this dataset are statements such as "The student reads the paper", and the target is the statement rephrased as a question; e.g., "Does the student read the paper?". We shorten the name of this dataset to "Q. Formation" in our result tables.

Prompt Templates The candidate prompt templates we utilized in our search consisted of few-shot templates taken from each dataset. For each dataset, we created six 3-shot prompt templates. The examples chosen for these prompt templates were found by embedding each dataset using a text embedding model, clustering the embeddings into six clusters, and then randomly selecting three examples from each cluster.

Metrics We follow the evaluation strategy of both (Kuhn, Gal, and Farquhar 2023) and (Lin, Trivedi, and Sun 2023) and use a *thresholded RougeL score*. Thus, if the RougeL score between a response and the ground truth is greater than a threshold γ then we say it is correct and has an accuracy of 1; otherwise, it is incorrect and has an accuracy of 0. We use $\gamma = 0.3$ for the StrategyQA and TriviaQA dataset (the value used in (Kuhn, Gal, and Farquhar 2023)) and $\gamma = 0.7$ for the Question Formation dataset (where this value was used in (Lin, Trivedi, and Sun 2023)). We use the higher threshold for Question Formation as it is a simpler task.

Confidence Search For Individual Models

In this first experiment, we consider the case where only a single model is available, alongside a set of prompt templates (described above). We are thus determining whether our Predicted-Instance-Confidence Search can improve performance by adaptively choosing the best prompt template to use for each input instance.

Table 2 show results for the FLAN-T5-XL, FLAN-UL2, MPT-30B, and BLOOM models, respectively. An important observation is that our proposed PICS-CS approach nearly always provides a performance increase of several percentage points when compared to the average performance of model and prompt template pairs, as well as a significant performance increase over the model and prompt template pair with maximum overall confidence ("Most Confident"). The cases where PICS-CS does not perform best both correspond to the case where inputs and the task are drawn from the Question Formation dataset. Specifically, in the cases where PICS-CS does not provide a significant boost, the Most

[2]https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/linguistic_mappings/question_formation_json

Model	Dataset	MPT Avg.	Most Confident	PICS-RS (Ours)	PICS-CS (Ours)	OCS-RS* (Ours)	OCS-CS* (Ours)
FLAN T5 XL	Strategy QA	48.68	48.55	49.20	**53.10**	50.90	54.25
	Trivia QA	21.09	21.40	21.40	**27.10**	25.25	27.30
	Q. Formation	61.48	62.90	66.25	**73.45**	70.20	75.40
FLAN UL2	Strategy QA	68.50	68.35	68.70	**69.30**	70.05	70.10
	Trivia QA	42.23	42.60	42.95	**46.10**	45.85	47.00
	Q. Formation	69.37	**87.30**	86.80	87.10	87.65	86.90
MPT 30B	Strategy QA	24.39	28.55	29.45	**38.25**	33.60	39.25
	Trivia QA	27.96	42.75	40.00	**49.00**	45.10	49.70
	Q. Formation	24.47	42.00	45.65	**79.50**	62.00	83.20
BLOOM	Strategy QA	25.22	30.55	30.00	**37.90**	31.60	36.10
	Trivia QA	43.02	45.35	45.80	**49.80**	45.80	51.25
	Q. Formation	98.48	**99.10**	98.35	98.85	97.20	97.00

Table 2: Performance of each search for each model. The approach with the best accuracy for a given model and dataset is given in bold. *We do not consider OCS-RS and OCS-CS when selecting the "best accuracy" due to these methods being too intractably expensive to run in practice.

Confident pair is already highly accurate (e.g., 99.10% accuracy for Most Confident with BLOOM), and PICS-CS is not significantly far off from this best performance; at worse, it is 0.30% off from the best performance.

Another very important thing to note is that on cases where every prompt template results in suboptimal performance for a given model (e.g., the Question Formation task with MPT-30B in Table 2), our PICS-CS approach increases performance by a large margin. For instance, MPT-30B increases from 42.0% accuracy on Question Formation to 79.50% accuracy when PICS-CS is used.

We draw attention to a result that at first glance may seem surprising: **in some cases, confidence search using *predicted* confidence (PICS) outperforms searches that use the calculated semantic confidence (OCS).** While intuitively it may seem like using the calculated confidence would be better than using predicted confidences, it is likely that predicting the confidence is acting as a kind of regularization for the confidence search. For example, for a given model and prompt template, if there is an instance that happens to have high confidence but is very similar to many training datapoints that have low confidence, then OCS will likely select this instance due to the fact that it has a high ground truth confidence. However, PICS would likely predict low confidence for this point, as it is most similar to low-confidence training instances. A high confidence point with all low confidence neighbors is likely to correspond to a point for which the model is *overconfident*. Thus, using predicted confidences are analogous to "smoothed over" confidences, and will be less sensitive to outliers.

Lastly, we note that the performance of the models does not always increase as model size increases. This is likely because the instruction tuned models, even if smaller, are more suitable for tasks in the dataset such as StrategyQA.

Confidence Search For Multiple Models

In this experiment, we perform searches over multiple models and multiple prompts; e.g., every model and prompt described above. Results are shown in Table 3.

We again see that PICS-CS usually improves performance by several percentage points. Notably, the performance of PICS-CS *increases* for 2 out of the 3 datasets; e.g., the best single-model performance for PICS-CS on TriviaQA is 49.80% (3, BLOOM model),

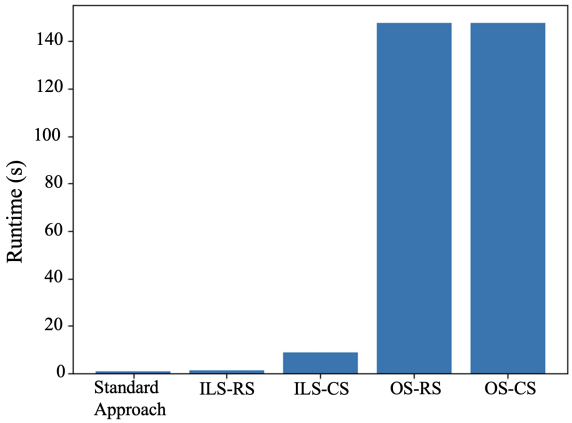

Figure 4: Runtime of each search method

while PICS-CS when searching over all models is 51.35% (Table 3).

Our approach does not improve performance for one dataset, Question Formation, in this setting. While it improves over the average performance of all model and prompt template pairs, it underperforms on the model and prompt template pair with the highest overall confidence. This is likely because some model and prompt template pairs perform very poorly for this task, and if a search method ever erroneously chooses these pairs then the overall performance is lowered significantly. Still, PICS-CS is only 2.30% off from the best performance.

Confidence Sampling

Figure 3 shows the effect that the number of samples drawn when doing the final confidence sampling step of PICS-CS. Results in this figure are taken from the experiment where FLAN-T5-XL is evaluated on inputs from the StrategyQA dataset. While increasing the number of samples improves performance, the computational cost increases as the LLM needs to be called more times.

Dataset	MPT Avg.	Most Confident	PICS-RS (Ours)	PICS-CS (Ours)	OCS-RS* (Ours)	OCS-CS* (Ours)
StrategyQA	41.70	68.35	69.00	**70.55**	61.60	61.75
TriviaQA	33.58	42.60	48.35	**51.35**	54.20	54.10
Q. Formation	65.85	**99.10**	97.50	96.80	94.75	94.65

Table 3: Accuracy for `All Models` for each dataset. The approach with the best accuracy for a given model and dataset is given in bold. *We do not consider OCS-RS and OCS-CS when selecting the "best accuracy" due to these methods being too intractably expensive to run in practice.

Runtime

We perform a runtime analysis of each approach in Figure 4. While PICS-CS has a somewhat longer run time than randomly choosing pairs, the increase in performance is likely worth the small increase in inference time. Additionally, using our predicted search is much more efficient than the oracle searches (last two bars). Note that OCS-RS and OCS-CS have equivalent run times; this is because multiple responses must be sampled from ever model and prompt template pair for methods. Thus, sampling from the most confident model and prompt template pair - required for confidence sampling (CS) - does not incur additional cost for the oracle method.

Related Works

The machine learning literature has shown us that a one-size-fits-all solution seldom exists in real-world applications. Often, different models should be adopted for different tasks and finding the right model(s) becomes the key problem to solve. Hyperparameter tuning, routing and ensembling techniques were researched extensively and many solutions relied on data similarity metrics (e.g., auto-AI (Arnold et al. 2020)) or model confidences that leveraged the probability distribution of model outputs. For LLMs, some works proposed approaches to calculate confidence more accurately in order to use routing and ensembling logic, e.g., using semantic similarity to group output phrases in natural language generation tasks (Kuhn, Gal, and Farquhar 2023; Lin, Trivedi, and Sun 2023). BlenderLLM (Jiang, Ren, and Lin 2023) selected the top-k models and merged their outputs using a sequence-to-sequence model.

Some opted to search for the best prompt and relied on a single LLM. (Rubin, Herzig, and Berant 2021) trained an input-output example retriever to construct the best possible prompt for a given input to the model. (Zhang, Feng, and Tan 2022) rely on reinforcement learning instead to select the best examples for in-context learning. Finally, (Zhang et al. 2022) automatically generated chain-of-thought prompts to obtain the best model output without the manual effort required by chain-of-thought prompting.

Conclusion

In this work, we proposed using the semantic *confidence* of LLMs to choose the best language model and prompt template pair to use for each input instance. Additionally, we proposed an efficient method for predicting these confidences without having to run each model for each instance. We also proposed the use of a final confidence sampling step to return a response with the highest confidence from the most confident model and prompt template pair. Our experimental evaluation on three disparate datasets showed that our sampling approach nearly always improves accuracy by several percentage points.

We hope that our work inspires more research into using confidence as a means for selecting the best model and/or prompt template to use for individual queries. Additionally, we hope our work on using predicted confidence rather than existing semantic confidences leads to more research on efficient yet robust confidence metrics for LLMs.

References

Arnold, M.; Boston, J.; Desmond, M.; Duesterwald, E.; Elder, B.; Murthi, A.; Navratil, J.; and Reimer, D. 2020. Towards automating the AI operations lifecycle. *arXiv preprint arXiv:2003.12808*.

Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, E.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Friedman, J. H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, 1189–1232.

Geva, M.; Khashabi, D.; Segal, E.; Khot, T.; Roth, D.; and Berant, J. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9: 346–361.

Ho, T. K. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, 278–282. IEEE.

Jiang, D.; Ren, X.; and Lin, B. Y. 2023. LLM-Blender: Ensembling Large Language Models with Pairwise Ranking and Generative Fusion. *arXiv preprint arXiv:2306.02561*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Kadavath, S.; Conerly, T.; Askell, A.; Henighan, T.; Drain, D.; Perez, E.; Schiefer, N.; Hatfield-Dodds, Z.; DasSarma, N.; Tran-Johnson, E.; et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Kuhn, L.; Gal, Y.; and Farquhar, S. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*.

Lin, C.-Y. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, 74–81.

Lin, S.; Hilton, J.; and Evans, O. 2022. Teaching models to express their uncertainty in words. *arXiv preprint arXiv:2205.14334*.

Lin, Z.; Trivedi, S.; and Sun, J. 2023. Generating with Confidence: Uncertainty Quantification for Black-box Large Language Models. *arXiv preprint arXiv:2305.19187*.

OpenAI, R. 2023. GPT-4 technical report. *arXiv*, 2303–08774.

Reimers, N.; and Gurevych, I. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Robinson, J.; Rytting, C. M.; and Wingate, D. 2022. Leveraging large language models for multiple choice question answering. *arXiv preprint arXiv:2210.12353*.

Rubin, O.; Herzig, J.; and Berant, J. 2021. Learning to retrieve prompts for in-context learning. *arXiv preprint arXiv:2112.08633*.

Scao, T. L.; Fan, A.; Akiki, C.; Pavlick, E.; Ilić, S.; Hesslow, D.; Castagné, R.; Luccioni, A. S.; Yvon, F.; Gallé, M.; et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Spatharioti, S. E.; Rothschild, D. M.; Goldstein, D. G.; and Hofman, J. M. 2023. Comparing Traditional and LLM-based Search for Consumer Choice: A Randomized Experiment. *arXiv preprint arXiv:2307.03744*.

Srivastava, A.; Rastogi, A.; Rao, A.; Shoeb, A. A. M.; Abid, A.; Fisch, A.; Brown, A. R.; Santoro, A.; Gupta, A.; Garriga-Alonso, A.; et al. 2023. Beyond the Imitation Game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837.

Xiong, M.; Hu, Z.; Lu, X.; Li, Y.; Fu, J.; He, J.; and Hooi, B. 2023. Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs. *arXiv preprint arXiv:2306.13063*.

Zhang, T.; Ladhak, F.; Durmus, E.; Liang, P.; McKeown, K.; and Hashimoto, T. B. 2023. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848*.

Zhang, Y.; Feng, S.; and Tan, C. 2022. Active example selection for in-context learning. *arXiv preprint arXiv:2211.04486*.

Zhang, Z.; Zhang, A.; Li, M.; and Smola, A. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Zhao, Z.; Wallace, E.; Feng, S.; Klein, D.; and Singh, S. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 12697–12706. PMLR.