Beyond the Next Token: Towards Prompt-Robust Zero-Shot Classification via Efficient Multi-Token Prediction

Junlang Qian[1] **Zixiao Zhu**[1,2] **Hanzhang Zhou**[1,2] **Zijian Feng**[1,2] **Zepeng Zhai**[3]
Kezhi Mao[1,2*]
[1]Nanyang Technological University, Singapore
[2]Singapore-ETH Centre [3]Tencent, China
{junlang001,zixiao001,hanzhang001,feng0119}@e.ntu.edu.sg
zealzhai@tencent.com ekzmao@ntu.edu.sg

Abstract

Zero-shot text classification typically relies on prompt engineering, but the inherent prompt brittleness of large language models undermines its reliability. Minor changes in prompt can cause significant discrepancies in model performance. We attribute this prompt brittleness largely to the narrow focus on next-token probabilities in existing methods. To address this, we propose **P**laceholding **P**arallel **P**rediction ($\mathcal{P}^3$), a novel approach that predicts token probabilities across multiple positions and simulates comprehensive sampling of generation paths in a single run of a language model. Experiments show improved accuracy and up to 98% reduction in the standard deviation across prompts, boosting robustness. Even without a prompt, $\mathcal{P}^3$ maintains comparable performance, reducing the need for prompt engineering.

1 Introduction

Zero-shot text classification (Radford et al., 2019; Hu et al., 2021; Wei et al., 2022; Wang et al., 2023b; Liu et al., 2023; Yang et al., 2023) is among the most challenging applications of pre-trained large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023; Anthropic, 2024), as it aims to categorize text without additional data. In this context, prompt engineering has become a widely adopted approach to enhance accuracy. However, language models exhibit inherent **prompt brittleness** (Sclar et al., 2023; Zamfirescu-Pereira et al., 2023; Zhou et al., 2024a): their outputs are highly sensitive to minor modifications in prompt wording or format (as shown in Figure 1), leading to inconsistent performance. This issue makes crafting effective and reliable prompts difficult, particularly in zero-shot scenarios where prior information is lacking.

While some efforts have attempted to mitigate this issue through training (Tam et al., 2021), general methods for the inference stage are rare. Calibration (Zhao et al., 2021) is one of the few, but its primary benefit lies in enhancing accuracy rather than robustness, leaving the severe prompt brittleness problem not sufficiently resolved.

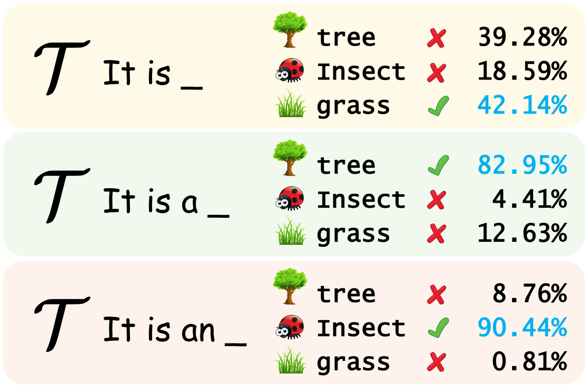


Figure 1: An example of prompt brittleness. The prompt "It is a _" yields a notably high score for "tree", while "It is an _" overwhelmingly favors "insect". The percentage scores are normalized for an arbitrary text unrelated to any class $\mathcal{T}$ = "knows grammar."

In this paper, we present a new perspective. We observe that current methods predominantly rely on next-token prediction (Bengio et al., 2000; Sutskever et al., 2014) to classify. We posit that this sole **reliance on the next token may contribute to prompt brittleness**, based on the following intuitions: (1) Since the next token directly follows the input, it would be inevitably impacted by the prompt, whereas later tokens are likely less sensitive[1]; (2) Language models may not immediately provide the answer as the next token. Instead, they often first generate non-discriminative words such as "so", "a", or "very", or engage in preliminary reasoning[2], which could hurt next-token performance for certain prompts.

*Corresponding author

[1]The Markov assumption implies that nearby words (or the preceding n-gram) have a stronger influence on predictions (Shannon, 1948, 1951; Brown et al., 1992; Almutiri and Nadeem, 2022).

[2]Some examples are in appendix A.1.

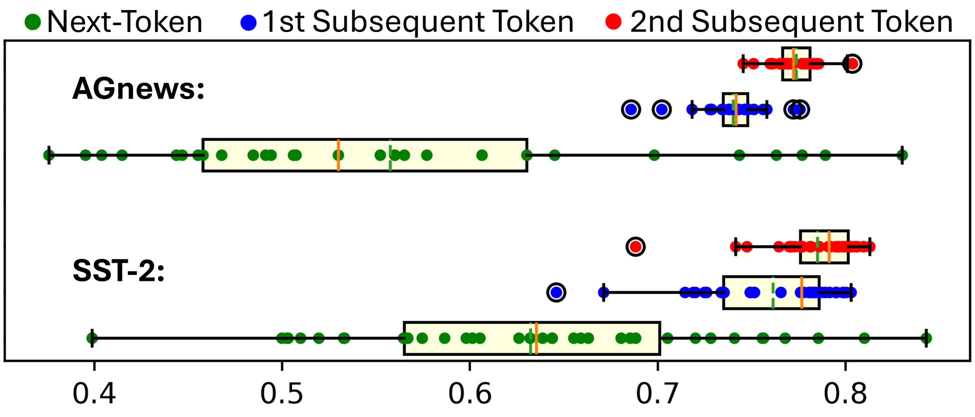


Figure 2: Accuracies of plausible prompts using tokens at the first three positions. Each row corresponds to a position: the ● green dots represent the next token (the 0th position), and the ● blue and ● red dots represent the 1st and 2nd tokens. Each dot denotes a prompt, and its horizontal coordinate indicates its performance.

Our experiments support our hypothesis. As illustrated in Figure 2, the accuracies across different prompts based on next-token prediction (●, green dots) are highly dispersed. In contrast, the accuracies for later tokens (●●, blue and red dots) are more tightly clustered together and demonstrate better overall performance. This suggests that subsequent token predictions are more robust to prompt variations and have the potential to improve performance.

However, to unlock the potential of subsequent token predictions, we must address a key challenge: current auto-regressive **language models predict only the next token**[3].

A naive solution is a token-by-token generation strategy, but it explores only a single generation path. Consequently, this strategy introduces error propagation, controllability issues, dependency on decoding algorithms, and randomness in generation path selection, all of which can hurt robustness[4]. In zero-shot natural language generation (NLG) tasks, a popular method to enhance this strategy is to sample multiple alternative generation paths and ensemble the results (Wang et al., 2023b; Lin et al., 2024; Zhang et al., 2024). Although extensive sampling could theoretically outperform next-token predictions in zero-shot classification[5], the computation of repeatedly generating numerous tokens many times is overly costly. Thus, the **inefficiency** stemming from token-by-token generation and path dependency limits our ability to leverage subsequent-token predictions to overcome prompt brittleness.

[3]An exception exists: one language model introduced by Gloeckle et al. can predict up to four tokens, but is still insufficient for our needs (more than ten tokens). Therefore, this model is not included in our study.

[4]In practice, generation-based strategies typically underperform next-token prediction in classification tasks (Puri and Catanzaro, 2019; Wei et al., 2021).

[5]If all possible generation paths were enumerated, token probabilities for all positions could be calculated, encompassing and exceeding the capabilities of next-token prediction.

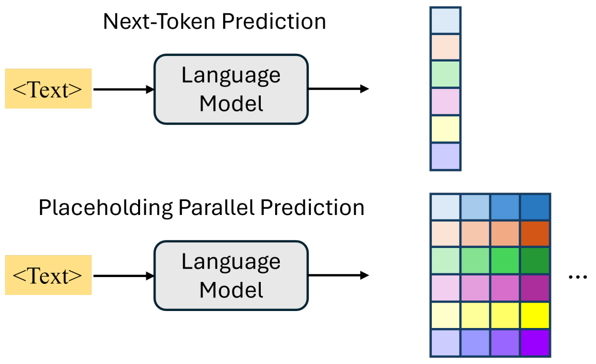


Figure 3: Next-Token Prediction versus Placeholding Parallel Prediction. Our proposed $\mathcal{P}^3$ obtains multiple token predictions in a single language model run.

To address this, we propose **Placeholding Parallel Prediction** ($\boldsymbol{\mathcal{P}^3}$), a novel and pluggable method. $\mathcal{P}^3$ appends placeholder tokens at the end of the input sequence to simulate comprehensive sampling of generation paths, thereby enabling multiple token predictions simultaneously within one model run. $\mathcal{P}^3$ introduces subsequent token predictions into zero-shot classification with high efficiency and without being affected by generation path dependency. This offers an effective solution to prompt brittleness. Through extensive experiments on seven public benchmarks, we demonstrate that $\mathcal{P}^3$ significantly alleviates prompt brittleness and surpasses SoTA accuracy. Notably, with $\mathcal{P}^3$, performance without any prompt instructions matches that with crafted prompts, significantly reducing the need for prompt engineering.

Our contributions are as follows:

- We propose a new perspective: the narrow focus on next-token prediction may be a key contributor to prompt brittleness in zero-shot classification.
- We introduce $\mathcal{P}^3$, the first method to obtain multiple subsequent token predictions for text classification within a single language model run. Our code is open-sourced.
- Extensive experiments demonstrate that $\mathcal{P}^3$ significantly reduces prompt brittleness and outperforms SoTA accuracy. Notably, we find that even without any prompt, the performance is comparable to that with crafted prompts, greatly reducing the reliance on prompt engineering.

2 Related Work

2.1 Prompt Brittleness

Prompt brittleness has emerged as a significant challenge in zero-shot classification (Krause et al., 2020; Schick and Schütze, 2021). This brittleness refers to the sensitivity of model performance to the specific wording and structure of prompts, where minor changes can lead to significant variations in output quality (Zhou et al., 2024b). Training-based solutions to address prompt brittleness often rely on labeled or unlabeled data. For instance, Logan IV et al. (2022) introduced strategies for simplifying prompt engineering through finetuning, but their methods require datasets for parameter optimization. Similarly, Wang et al. (2023a,c) leveraged contrastive self-training, which involves iterative pseudo-labeling using generated data, but this also depends on large amounts of unlabeled data to work. Such methods are therefore impractical for zero-shot learning.

Consequently, few attempts have focused on mitigating prompt brittleness directly during the inference phase, where the model operates without any further adjustments. Calibration methods (Zhao et al., 2021) represent one of the few efforts in this area. Zhao et al. propose calibration strategies to normalize model outputs based on contextual priors, improving consistency across different prompts without requiring external data. Another line of research aims to bypass prompt brittleness by automatically generating task-specific prompts (Pryzant et al., 2023; Gao et al., 2021; Holtzman et al., 2021; Jiang et al., 2020). Rather than fundamentally addressing the brittleness problem, these approaches serve as workarounds, sacrificing interpretability and cross-task consistency.

2.2 Zero shot Text Classification

Since class labels typically possess clear semantics, we can leverage the capabilities of language models for zero-shot classification (Radford et al., 2021; Zhou et al., 2023b).

Zero-shot text classification (Puri and Catanzaro, 2019; Brown et al., 2020; Liu et al., 2023) has evolved from earlier approaches like embedding-based semantic matching and natural language inference (NLI). Embedding-based methods (Cer et al., 2018; Reimers and Gurevych, 2019) suffer from semantic ambiguity, as inputs with subtle differences can receive similar embeddings. NLI-based methods (Bowman et al., 2015; Ma et al., 2021; Zhu et al., 2024) decompose a multi-class problem into binary tasks, but this often results in poor calibration across binary classifiers, with inconsistent performance between classes.

Recent methods focus on next-token prediction and generation-based techniques. In next-token prediction, the model assigns a score based on the generation probability of a class label token (Zhao et al., 2021). Although efficient, this method considers only a single token, limiting its capacity (Feng et al., 2024). Generation-based methods, on the other hand, prompt the model to generate text and analyze label presence (Radford et al., 2019; Brown et al., 2020; Wang et al., 2023b). While these approaches can capture richer semantics, they are constrained to one generation path at a time and face controllability issues (Krause et al., 2020; Ouyang et al., 2022).

Although these recent approaches address some shortcomings of earlier methods, they remain incomplete, as one focuses narrowly on a single token and the other only explores one possible output path. This leaves room for improvements in achieving more comprehensive classification strategies.

3 Methodology

3.1 Background

Text classification seeks to assign the correct label $c^\star$ to a given text $\boldsymbol{t}$, with $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ representing the predefined set of candidate categories. Current approaches to zero-shot text classification rely on the assumption that the correct class label

$c^\star$ or its corresponding tokens will exhibit higher generation probabilities under the language model than incorrect ones:

$$P(c^\star \mid \boldsymbol{t}) > P(c' \mid \boldsymbol{t}), \quad \forall c' \in \mathcal{C},\, c' \neq c^\star.$$

A language model $\mathcal{LM}$ predicts the generation probability of the next token given an input sequence $\boldsymbol{x}$. Specifically,

$$\mathcal{LM}(\boldsymbol{x})_{x_n} = P(x_n \mid (x_0, x_1, \ldots, x_{n-1})),$$

where the input $\boldsymbol{x}$ is constructed by integrating the given text $\boldsymbol{t}$ into a prompt template $p(\cdot)$:

$$\boldsymbol{x} = (x_0, x_1, \ldots, x_{n-1}) = p(\boldsymbol{t}).$$

The most commonly used approach is **based on next token prediction**, which assumes that the class probability $P(c \mid \boldsymbol{t})$ is approximated by the language model's next token probability:

$$P(c \mid \boldsymbol{t}) \approx \mathcal{LM}(\boldsymbol{x})_{x_n=c}.$$

Therefore, the class label is predicted by:

$$\hat{c} \leftarrow \arg\max_{c \in \mathcal{C}} \mathcal{LM}(\boldsymbol{x})_{x_n=c}.$$

An alternative approach **based on generation** leverages the continuation capability of language models to generate an output sequence:

$$\boldsymbol{x} \xrightarrow{\mathcal{LM}} (x_n, x_{n+1}, \hat{\ldots}, x_{n+m-1}).$$

The class label is then directly determined by which one appears in the generated sequence:

$$\hat{c} \leftarrow \{\hat{x}_n, \hat{x}_{n+1}, \ldots, \hat{x}_{n+m-1}\} \cap \mathcal{C}.$$

However, both methods remain incomplete: next-token prediction focuses on a single token position, while generation-based methods consider only one instance among many possible generation paths, limiting their robustness.

3.2 Placeholding Skipping Prediction ($\mathcal{PSP}$)

Let the i-th output token be x_{n+i} (where the next token is the 0-th token). All possible generation prefixes can be expressed as:

$$\{(x_n, x_{n+1}, \ldots, x_{n+i-1}) \mid x_j \in \mathcal{V},\, n \leq j < n+i\},$$

where $\mathcal{V}$ denotes the vocabulary, and $\boldsymbol{x}$ represents the known input.

The probability of each prefix is:

$$P(x_n, x_{n+1}, \ldots, x_{n+i-1} \mid \boldsymbol{x}).$$

Given this prefix, the probability of $x_{n+i} = c$ is:

$$P(x_{n+i} = c \mid \boldsymbol{x}, x_{n+1}, \ldots, x_{n+i-1}).$$

Thus, the probability of the entire generation path is:

$$\begin{aligned} &P(x_n, x_{n+1}, \ldots, x_{n+i} = c \mid \boldsymbol{x}) \\ &\quad = P(x_n, x_{n+1}, \ldots, x_{n+i-1} \mid \boldsymbol{x}) \\ &\qquad \times P(x_{n+i} = c \mid \boldsymbol{x}, x_{n+1}, \ldots, x_{n+i-1}). \end{aligned}$$

Next, by enumerating all possible prefixes, the overall probability for $x_{n+i} = c$ becomes:

$$\begin{aligned} &P(x_{n+i} = c \mid \boldsymbol{x}) \\ &= \sum_{\substack{(x_n, \ldots, x_{n+i-1}) \\ \in \mathcal{V}^i}} P(x_n, x_{n+1}, \ldots, x_{n+i-1} \mid \boldsymbol{x}) \\ &\quad \times P(x_{n+i} = c \mid \boldsymbol{x}, x_n, x_{n+1}, \ldots, x_{n+i-1}). \end{aligned}$$

This is equivalent to:

$$\begin{aligned} &P(x_{n+i} = c \mid \boldsymbol{x}) \\ &= P\left(x_{n+i} = c \,\middle|\, \boldsymbol{x}, \underbrace{x_n, x_{n+1}, \ldots, x_{n+i-1}}_{\text{unknown}} \right). \end{aligned}$$

We approximate this as:

$$\begin{aligned} &P(x_{n+i} = c \mid \boldsymbol{x}) \\ &\approx P\left(x_{n+i} = c \,\middle|\, \boldsymbol{x}, \underbrace{\texttt{<ph>}, \texttt{<ph>}, \ldots, \texttt{<ph>}}_{\text{i times}} \right). \end{aligned}$$

This can be rewritten using the language model as:

$$\begin{aligned} &P(x_{n+i} = c \mid \boldsymbol{x}) \\ &\approx \mathcal{LM}\left(\boldsymbol{x}, \underbrace{\texttt{<ph>}, \texttt{<ph>}, \ldots, \texttt{<ph>}}_{\text{i times}} \right)_{x_{n+i}=c}. \end{aligned}$$

As shown in Figure 4(b), we append i placeholder tokens, <ph>[6], to the input sequence $\boldsymbol{x}$

[6]In LLaMA2, we selected the unknown token <unk> as the placeholder <ph>. This token originally signifies "unknown", representing out-of-vocabulary (OOV) tokens, such as unrecognized language or unreadable characters, which can somewhat convey the intended meaning. Additionally, <unk> offers two key advantages: (1) it has length, and (2) it carries no semantic meaning.

and feed this extended sequence into the language model. This allows us to obtain the prediction for the i-th subsequent token, formally expressed as:

$$\mathcal{PSP}(\boldsymbol{x}, i) = \mathcal{LM}(\boldsymbol{x'}),$$

where:

$$\boldsymbol{x'} = (x_0, x_1, \ldots, x_{n-1}, \underbrace{\text{<ph>}, \text{<ph>}, \ldots, \text{<ph>}}_{i \text{ times}}).$$

We refer to this approximation of the i-th token prediction as Placeholding Skipping Prediction ($\mathcal{PSP}$).

3.3 Placeholding Parallel Prediction ($\mathcal{P}^3$)

In practical implementation, LLMs employ a transformer architecture with blocks stacked in both depth and sequence length, maintaining width-aligned input and output. Transformers with uni-directional attention produce an output for each prefix of the input sequence $\boldsymbol{x}$ during inference:

$$\text{transformer}(\boldsymbol{x}) = \begin{bmatrix} \mathcal{LM}(\varnothing), \\ \mathcal{LM}(x_0), \\ \mathcal{LM}(x_0, x_1), \\ \ldots, \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}) \end{bmatrix}.$$

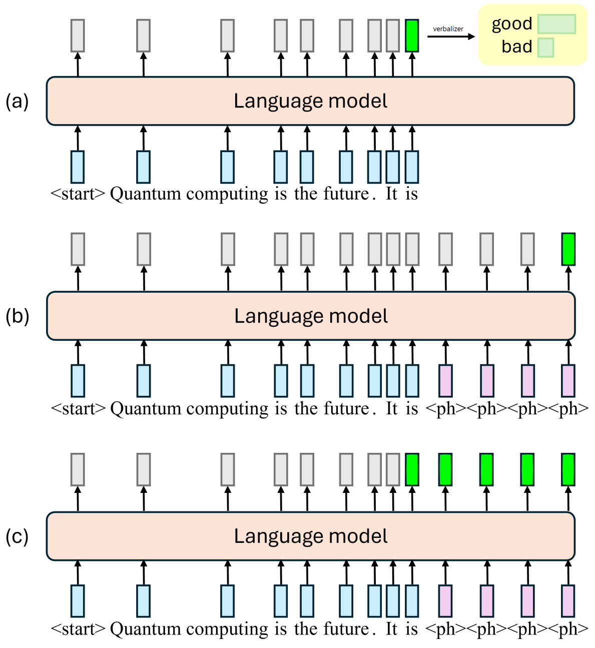

verbalizer
good
bad
(a)
Language model
<start> Quantum computing is the future. It is
(b)
Language model
<start> Quantum computing is the future. It is <ph><ph><ph><ph>
(c)
Language model
<start> Quantum computing is the future. It is <ph><ph><ph><ph>

Figure 4: (a) Next-Token Prediction. (b) Placeholding Skipping Prediction ($\mathcal{PSP}$). (c) Placeholding Parallel Prediction ($\mathcal{P}^3$). The small green rectangles indicate the output tokens to be used, and the grey ones indicate those not to be used. <ph> represents a placeholder token.

This property supports tasks such as computing text likelihood and parallel training. As depicted in Figure 4(a), current classification methods merely use the next-token prediction, $\mathcal{LM}(\boldsymbol{x}) = \mathcal{LM}(x_0, x_1, \ldots, x_{n-1})$, the last element of the transformer output.

Appending a series of <ph> tokens (subject to memory constraints) to the input sequence enables the transformer to compute all prefixes automatically:

$$\text{transformer}(\boldsymbol{x'}) = \begin{bmatrix} \mathcal{LM}(\varnothing), \\ \mathcal{LM}(x_0), \\ \mathcal{LM}(x_0, x_1), \\ \ldots, \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}), \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}, \text{<ph>}), \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}, \text{<ph>}, \text{<ph>}), \\ \ldots \end{bmatrix},$$

where:

$$\boldsymbol{x'} = (x_0, x_1, \ldots, x_{n-1}, \text{<ph>}, \text{<ph>}, \ldots).$$

As illustrated in Figure 4(c), extracting our desired elements, which correspond to the next-token and all subsequent tokens, we define:

$$\mathcal{P}^3(\boldsymbol{x}) = \begin{bmatrix} \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}), \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}, \text{<ph>}), \\ \mathcal{LM}(x_0, x_1, \ldots, x_{n-1}, \text{<ph>}, \text{<ph>}), \\ \ldots \end{bmatrix}$$

$$= \begin{bmatrix} \mathcal{LM}(\boldsymbol{x}), \\ \mathcal{PSP}(\boldsymbol{x}, 1), \\ \mathcal{PSP}(\boldsymbol{x}, 2), \\ \ldots \end{bmatrix} = \begin{bmatrix} \mathcal{PSP}(\boldsymbol{x}, 0), \\ \mathcal{PSP}(\boldsymbol{x}, 1), \\ \mathcal{PSP}(\boldsymbol{x}, 2), \\ \ldots \end{bmatrix}.$$

By leveraging the transformer's inherent capability to handle multiple token predictions[7] in a single run, this approach significantly enhances efficiency. We term this method $\mathcal{P}^3$.

4 Experiment

4.1 Datasets, Models, and Prompts

We conducted extensive experiments on two large language models and seven publicly available

[7]Note that our multi-token prediction does not generate a specific sequence of tokens; rather, it produces a probability distribution over the vocabulary for each token position.

Dataset			Gen	3w-SC	NTP	Cali1	Cali2	Cali3	Ours
IMDb	13B	acc (%)	–	–	68.11	79.36	77.92	75.08	**82.35 +14.24**
		std			0.1003	0.0899	0.0907	0.0962	**0.0020 -98.05%**
	70B	acc (%)	47.92	53.13	61.60	71.23	71.82	68.99	**75.86 +14.26**
		std	0.2053	0.3291	0.0729	0.0798	0.0749	0.0834	**0.0721 -7.92%**
AGnews	13B	acc (%)	–	–	55.75	60.40	61.70	61.15	**80.51 +24.76**
		std			0.1267	0.1361	0.1106	0.1053	**0.0068 -94.64%**
	70B	acc (%)	58.59	50.88	48.24	49.37	50.38	49.62	**80.14 +31.90**
		std	0.1180	0.3016	0.1544	0.1546	0.1398	0.1450	**0.0329 -78.71%**
DBpedia	13B	acc (%)	–	–	64.39	56.52	61.91	60.04	**64.47 +0.08**
		std			0.0965	0.2223	0.0922	0.0948	**0.0052 -94.64%**
	70B	acc (%)	66.41	56.35	70.03	67.11	67.23	65.74	**72.98 +2.95**
		std	0.0438	0.3288	0.0744	0.0694	0.0822	0.0865	**0.0469 -36.92%**
Amazon	13B	acc (%)	–	–	65.35	77.37	78.03	78.62	**81.62 +16.27**
		std			0.0986	0.1086	0.0886	0.0839	**0.0065 -93.45%**
	70B	acc (%)	49.22	53.32	61.14	60.50	61.02	61.41	**78.72 +17.58**
		std	0.1413	0.3315	0.0466	0.0766	0.0699	0.0772	0.0644 +38.39%
ISEAR	13B	acc (%)	–	–	36.79	27.48	41.07	37.73	**41.61 +4.82**
		std			0.0935	0.1771	0.0712	0.0960	**0.0182 -80.51%**
	70B	acc (%)	24.09	25.88	36.90	40.59	33.30	37.17	**43.22 +6.32**
		std	0.0208	0.1500	0.0935	0.0835	0.0716	0.0875	**0.0653 -30.18%**
SST-2	13B	acc (%)	–	–	63.21	73.15	69.52	67.60	**80.14 +16.92**
		std			0.0997	0.1145	0.0995	0.1131	**0.0178 -82.17%**
	70B	acc (%)	43.49	47.85	57.30	66.64	66.98	60.57	**71.07 +13.77**
		std	0.2419	0.3224	0.0612	0.1041	0.1007	0.0994	0.0843 +37.65%
Yahoo	13B	acc (%)	–	–	41.90	34.94	45.12	45.48	**50.50 +8.59**
		std			0.0967	0.1918	0.0797	0.0781	**0.0073 -92.43%**
	70B	acc (%)	36.98	31.93	37.74	38.85	40.30	39.07	**50.74 +13.00**
		std	0.0516	0.1931	0.1210	0.0980	0.1006	0.0844	**0.0544 -55.03%**
Avg	13B	acc (%)	–	–	56.50	58.46	62.18	60.81	**68.74 +12.24**
		std			0.1017	0.1486	0.0904	0.0953	**0.0091 -91.05%**
	70B	acc (%)	46.67	45.62	53.28	56.33	55.78	54.65	**67.53 +14.25**
		std	0.1175	0.2795	0.0899	0.0951	0.0914	0.0948	**0.0600 -33.26%**

Table 1: Results of accuracy and cross-prompt standard deviation (i.e., prompt brittleness) for each dataset. Gen refers to a vanilla generative method, and 3w-SC denotes three-way self-consistency. NTP represents the vanilla next-token prediction method, which serves as our baseline. Cali1, Cali2, and Cali3 correspond to calibration methods using "N/A," an empty string, and "<unk>*5" as the calibration text, respectively. Avg represents the average across datasets. For further details on hyperparameter settings, see appendix A.3. As shown, our method significantly improves stability while achieving the highest accuracy.

datasets, using over 30 plausible prompts for each dataset.

We employed seven publicly available text classification datasets: Amazon Review Polarity (Zhang et al., 2015a), SST2 (Socher et al., 2013), and IMDb (Maas et al., 2011) for sentiment classification (binary labels: positive/negative) of product and movie reviews; AGnews (Zhang et al., 2015b) and DBpedia (Bizer et al., 2009) for topic classification (four and fourteen categories, respectively) of news articles and Wikipedia content; Yahoo Answers (Zhang et al., 2015b) comprises questions and answers classified into ten categories. ISEAR (Scherer and Wallbott, 1994) consists of sentences labeled with seven emotions for emotion classification. To save time, we selected 10,000 random instances from the IMDb, Amazon, DBpedia, and Yahoo datasets for LLaMA2-70b, and from the Amazon dataset for LLaMA2-13b.

The two models are LLaMA2-13b, with 13 billion parameters for efficient NLP tasks, and LLaMA2-70b, with 70 billion parameters for more complex tasks and broader context handling (Touvron et al., 2023).

We employed a method similar to that of Gonen et al. (2022) to generate prompts, using ChatGPT-4 (OpenAI et al., 2024) to expand a set of seed prompts. A comprehensive list of all prompts used in our experiments is provided in the appendix A.9.

4.2 Comparison Methods

Our baseline method employs the straightforward next-token prediction for classification. Additionally, we implemented a calibration method (Zhao et al., 2021) to compare its performance with subsequent tokens. The calibration method adjusts the conditional generation probability (classification scores) of the next-token under different prompts by introducing meaningless text inputs and performing a single additional run of the model. This approach achieves state-of-the-art performance in zero-shot classification without requiring any manually collected external information, knowledge, or unlabeled data resources, thereby ensuring interpretability. The calibration method's enhancements to zero-shot classification focus on two critical aspects: average accuracy and stability, both of which are pivotal to our evaluation criteria. We used three different meaningless strings for calibration to ensure a comprehensive comparison. We also evaluated 256 samples per dataset using generative methods and three-way self-consistency to demonstrate our efficiency.

4.3 Experimental Details

We conducted the experiments on two A6000 GPUs.

In our experiments, for each data sample and prompt combination, we appended 512 additional <ph> tokens[8] to examine the effectiveness of our method across a broad range. To demonstrate that our approach is not reliant on this specific number, we introduced the hyperparameter η to control the range of tokens considered dynamically. Small η values focus on tokens close to the next position, while large η values consider tokens far down the sequence. Specifically, when $\eta = 0$, the method is equivalent to next-token prediction.

The two models exhibit behavior differences[9], prompting us to adapt our method accordingly. For LLaMA2-13B, classification is based on the probability distribution of the token located at a position proportional to the number of input tokens, determined by a slope of $\tan(\eta)$. We set η as the angle and map it into the tangent space for smoother transitions and scaling. For LLaMA2-70b, we define $[0, \eta)$ as a fixed range to vote and apply calibration to enhance the results.

5 Results Analysis

5.1 Overview

Aggregating results across seven datasets reveals that our $\mathcal{P}^3$ method substantially improves **efficiency**, **robustness**, and **accuracy**.

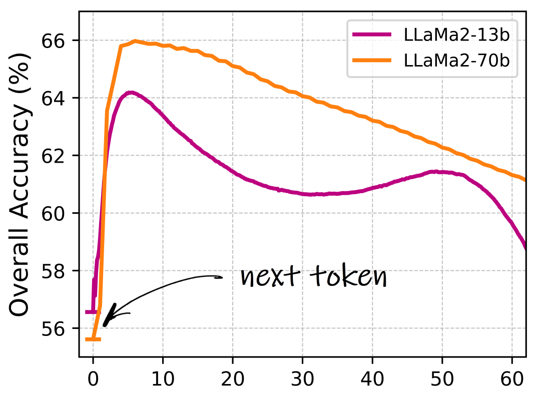
Dataset	**Number of Runs**			
	Gen	**3w-SC**	**NTP**	$\mathcal{P}^3$
Amazon	30.75	92.26	1	1
IMDb	28.03	84.09	1	1
AGnews	46.98	140.94	1	1
DBpedia	36.09	108.27	1	1
Yahoo	34.52	103.55	1	1
SST-2	25.80	77.39	1	1
ISEAR	26.86	80.58	1	1

Table 2: The average number of output tokens used to reach an answer (number of model runs). For outputs exceeding 50 tokens without matching any of the options, the sequence is truncated at 50. Gen denotes the vanilla generative approach, 3w-SC denotes 3-way self-consistency, NTP denotes next-token prediction, and $\mathcal{P}^3$ is ours.

Efficiency: Unlike generative approaches that require sequential token-by-token predictions through multiple model runs, $\mathcal{P}^3$ obtains multiple token predictions simultaneously in a single run. As evidenced in Table 2, $\mathcal{P}^3$ achieves time complexity on par with direct next-token prediction, ensuring high efficiency. As a trade-off for

[8]The use of so many <ph> tokens was initially intended to ensure redundancy but turned out to be unnecessary.

[9]We discuss the findings of behavioral differences between the two models in appendix A.5, but since they are orthogonal to this study, we do not elaborate on them here.



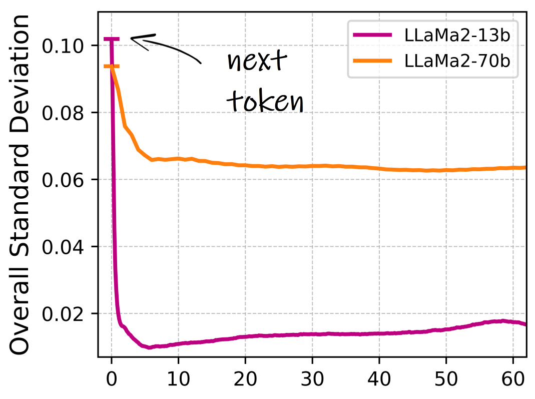


Figure 5: Average accuracy and average cross-prompt standard deviation (i.e., prompt brittleness) across seven datasets of $\mathcal{P}^3$. The horizontal axis represents the hyperparameter η, where $\eta = 0$ corresponds to the next-token prediction results, and larger η values indicate consideration of more distant token positions. $\eta > 0$ shows higher accuracy and lower standard deviation compared to next-token prediction.

multiple token prediction, $\mathcal{P}^3$ consumes slightly more floating-point operations (FLOP) compared to next-token prediction, as shown in Table 3.

Dataset	13B		70B	
	NTP	$\mathcal{P}^3$	NTP	$\mathcal{P}^3$
Amazon	2.82	3.05	14.26	14.91
IMDb	8.31	9.03	41.91	42.56
AGnews	1.62	1.75	8.20	8.85
DBpedia	2.16	2.33	10.94	11.59
Yahoo	3.67	3.98	18.56	19.21
SST-2	0.67	0.71	3.38	4.02
ISEAR	0.70	0.75	3.57	4.22

Table 3: Average numbers of FLOPs for next-token prediction (NTP) and $\mathcal{P}^3$ on different datasets for 13B and 70B models, all reported in Tflops, with $\eta = 5$. See appendix A.6 for the estimation method and FLOPs for different numbers of <ph> tokens.

Robustness and Accuracy: $\mathcal{P}^3$ significantly reduces prompt brittleness and achieves better performance.

Figure 5 illustrates that applying a unified hyperparameter η to all datasets, $\mathcal{P}^3$ **consistently outperforms** next-token prediction ($\eta = 0$) in overall accuracy and robustness within a broad range ($0 < \eta < 60$). Notably, the average cross-prompt standard deviation (i.e., prompt brittleness) reaches its minimum around $\eta = 5$ and stabilizes afterward, with accuracy peaking around the same point. This behavior implies that $\eta \approx 5$ can serve as a nice practical default setting when without task-specific prior knowledge.

Our method adapts effectively to diverse specific scenarios, and the results for individual datasets are presented in Table 1. $\mathcal{P}^3$ achieves high accuracy and robustness on the evaluated datasets, outperforming baseline and state-of-the-art approaches. Especially on LLaMA2-13b, our method achieved over 80% reduction in standard deviation on every dataset, with an average reduction of 91%. For AGNews, the LLaMA2-13b and LLaMA2-70b models achieved 25% and 32% increases in accuracy, along with 94% and 79% reductions in standard deviation, respectively. On IMDb, the LLaMA2-13B model not only improved accuracy by 14% but also yielded a standard deviation of only 0.002 (with a decrease over 98%), almost eliminating prompt brittleness.

5.2 Performance without Prompt

Dataset	13B		70B	
	Crafted	NoP	Crafted	NoP
IMDb	82.34	82.51	75.82	77.33
AGnews	80.52	80.26	80.15	79.85
Yahoo	50.47	51.06	50.84	48.04
Amazon	81.61	81.78	79.66	80.60
SST-2	80.13	80.45	71.23	65.28
DBpedia	64.50	63.52	72.98	72.96
ISEAR	41.61	41.66	43.23	42.66

Table 4: The average accuracy comparison between null prompts (i.e., using no prompt) and crafted prompts. Crafted represents crafted prompts, and NoP indicates using no prompt.

Given the significant reduction in standard deviation, the differences between various prompts have been largely minimized, making the choice of a specific prompt less critical. As shown in Table 4, using a null prompt (i.e., without any prompt) for zero-shot classification yields performance comparable to that of crafted prompts. This demonstrates that $\mathcal{P}^3$ effectively mitigates prompt brittleness and substantially reduces the reliance on prompt engineering.

6 Disscusion

What is the motivation of prompt engineering? The need for prompt engineering arises from the prompt brittleness (or cross-prompt instability), as not all prompts yield the same accuracy. Prompt engineering would become unnecessary if all prompts achieve consistent accuracy. Addressing the issue of prompt brittleness would save significant costs associated with prompt engineering, making it a worthwhile research topic.

Is prompt engineering necessary in zero-shot classification? We leverage the explicit semantic information contained within the classification labels to perform zero-shot classification. We assign a sample to the label with which it has the highest semantic alignment. We use the token generation probability as classification scores in zero-shot classification. This approach assumes that if the context is closely related to the semantics of a certain class label, tokens associated with that class will exhibit a higher co-occurrence probability compared to tokens related to other classes. The purpose of adding a prompt is to guide the model to generate class-related tokens as the next token. These class-related tokens may not necessarily appear at the next position but could emerge later in the text. The misalignment between the classification task and NLG, leading to the out-of-distribution (OOD) problem, undermines zero-shot performance by causing LLMs to generate non-discriminative words before the relevant class labels. Some studies, like Gonen et al. (2022) and Zhou et al. (2023a), alleviate the OOD issue by calculating prompt perplexity, but we address the core principle directly. Once subsequent tokens are predicted, the prompt's influence reduces. Logically, the text and the categories should suffice for classification. However, next-token predictions are often unstable, with accuracy fluctuations exceeding 10%, leading to the introduction of prompt engineering. Subsequent token predictions are significantly more stable and do not require such adjustments. As analyzed in Section 5.2, our $\mathcal{P}^3$ method allows for good classification scores without the need for a prompt.

Why are zero-shot classification capabilities not commonly used as an evaluation metric for large language models? Theoretically, zero-shot classification is an excellent, fair, and direct evaluation metric. However, the accuracy of next-token zero-shot classification is heavily influenced by the quality of the prompt. This reliance on prompt selection makes zero-shot classification a less reliable metric. Conversely, using the accuracy distribution of subsequent tokens obtained by our proposed $\mathcal{P}^3$, which is much more stable and less affected by the prompt, provides a fairer and more reliable evaluation metric.

7 Conclusion

Prompt brittleness is a critical challenge in zero-shot classification, undermining the reliability and consistency of language model outputs. In this work, we proposed $\mathcal{P}^3$, a novel method that introduces subsequent token predictions within a single model run to address this issue. Our extensive experiments across seven public benchmarks demonstrated that $\mathcal{P}^3$ significantly mitigates prompt brittleness while achieving accuracy beyond the SoTA. Notably, $\mathcal{P}^3$ performs equally well with or without prompts, greatly reducing the reliance on prompt engineering. These findings highlight $\mathcal{P}^3$'s potential as an effective solution for robust zero-shot classification, making classification more efficient and reliable.

8 Limitation

In the paper, we discussed using existing tokens as placeholders <ph>. Training soft tokens as placeholders may be another solution. For autoregressive language models, this approach involves training a soft token embedding of the ideal <ph> with a corpus. The formal expression of this token's properties is:

$$\mathcal{LM}(\boldsymbol{x} \oplus \text{<ph>})_y = \sum_{v \in \mathcal{V}} \mathcal{LM}(\boldsymbol{x})_v \cdot \mathcal{LM}(\boldsymbol{x} \oplus v)_y$$

where $\oplus$ denotes the concatenation of sequences.

Future work will focus on the behavior of tokens generated by our $\mathcal{P}^3$ method across different types of models and languages. Additionally, we plan to train a language model inherently equipped with <ph> tokens.

References

Talal Almutiri and Farrukh Nadeem. 2022. Markov models applications in natural language processing: a survey. *Int. J. Inf. Technol. Comput. Sci*, 2:1–16.

AI Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, volume 13. MIT Press.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. 2009. Dbpedia - a crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165. The Web of Data.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *Preprint*, arXiv:1803.11175.

Zijian Feng, Hanzhang Zhou, Zixiao Zhu, Junlang Qian, and Kezhi Mao. 2024. Unveiling and manipulating prompt influence in large language models. In *The Twelfth International Conference on Learning Representations*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A. Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *Preprint*, arXiv:2212.04037.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. *arXiv preprint arXiv:2104.08315*.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Jingang Wang, Juanzi Li, Wei Wu, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *arXiv preprint arXiv:2108.02035*.

Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. 2020. Can prompt-based learning be practical for low-resource tasks? *arXiv preprint arXiv:2001.07676*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. Gedi: Generative discriminator guided sequence generation. *Preprint*, arXiv:2009.06367.

Lei Lin, Jiayi Fu, Pengli Liu, Qingyang Li, Yan Gong, Junchen Wan, Fuzheng Zhang, Zhongyuan Wang, Di Zhang, and Kun Gai. 2024. Just ask one more time! self-agreement improves reasoning of language models in (almost) all scenarios. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3829–3852.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Robert L. Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2022. Cutting down on prompts and parameters: Simple few-shot learning with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3461–3476.

Tingting Ma, Jin-Ge Yao, Chin-Yew Lin, and Tiejun Zhao. 2021. Issues with entailment-based zero-shot text classification. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 786–796, Online. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll 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 F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Raul Puri and Bryan Catanzaro. 2019. Zero-shot text classification with generative language models. *arXiv preprint arXiv:1912.10165*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Preprint*, arXiv:1908.10084.

Klaus R Scherer and Harald G Wallbott. 1994. Evidence for universality and cultural variation of differential emotion response patterning. *Journal of personality and social psychology*, 66(2):310.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze questions for few shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

C. E. Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423.

Claude E Shannon. 1951. Prediction and entropy of printed english. *Bell system technical journal*, 30(1):50–64.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *arXiv preprint arXiv:2103.11955*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Fan Wang, Lei Zhao, and Li Chen. 2023a. Generation-driven contrastive self-training for zero-shot text classification with instruction-tuned gpt. *arXiv preprint arXiv:2304.11872*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Yau-Shian Wang, Ta-Chung Chi, Ruohong Zhang, and Yiming Yang. 2023c. Pesco: prompt-enhanced self contrastive learning for zero-shot text classification. *arXiv preprint arXiv:2305.14963*.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *Preprint*, arXiv:2306.02224.

JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Ziqi Zhang, Cunxiang Wang, Xiong Xiao, Yue Zhang, and Donglin Wang. 2024. Nash cot: Multi-path inference with preference equilibrium. *arXiv preprint arXiv:2407.07099*.

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.

Han Zhou, Xingchen Wan, Lev Proleev, Diana Mincu, Jilin Chen, Katherine Heller, and Subhrajit Roy. 2024a. Batch calibration: Rethinking calibration for in-context learning and prompt engineering. *Preprint*, arXiv:2309.17249.

Hanzhang Zhou, Zijian Feng, Zixiao Zhu, Junlang Qian, and Kezhi Mao. 2024b. Unibias: Unveiling and mitigating llm bias through internal attention and ffn manipulation. *arXiv preprint arXiv:2405.20612*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023a. Large language models are human-level prompt engineers. *Preprint*, arXiv:2211.01910.

Yunjiao Zhou, Jianfei Yang, Han Zou, and Lihua Xie. 2023b. Tent: Connect language models with iot sensors for zero-shot activity recognition. *arXiv preprint arXiv:2311.08245*.

Zixiao Zhu, Junlang Qian, Zijian Feng, Hanzhang Zhou, and Kezhi Mao. 2024. Edentail: An entailment-based few-shot text classification with extensional definition. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1124–1137.

A Appendix

A.1 LM Output Examples

Some examples of the language model's greedy output are in Table 5. As shown, the generation of language models does not ensure that the next token will produce the discriminative words necessary for classification.

Input:	My ACL paper is accepted. It is _
Output:	a very good paper.
Input:	My legs are broken. I am _
Output:	in a lot of pain.
Input:	I had my first job this year. It is _
Output:	a part-time job at a local restaurant. I am a waitress. I like my job very much.

Table 5: Greedy output examples from LLaMa2-13B. Blue text indicates non-discriminative expressions and red text indicates discriminative words for target categories {positive, negative}.

A.2 Zero-Shot Classification: Worth Studying

Zero-shot text classification remains a crucial research area, even as few-shot learning gains attention. Few-shot learning cannot fully replace zero-shot approaches, as it still relies on labeled data, which is not always feasible in many practical scenarios.

Broader Applicability and Fewer Constraints: Zero-shot learning is more aligned with real-world needs, especially when users cannot provide labeled examples, or when constructing even a single example for complex inputs is impractical. In contrast, few-shot learning assumes a shared distribution between provided and test examples, which is not always guaranteed.

Core Foundation for Research: Zero-shot learning serves as a more fundamental base for research, offering cleaner results without complexities like sample selection or ordering seen in few-shot studies, which can obscure core contributions.

Greater challenge and a benchmark for intrinsic model capabilities: Zero-shot classification directly tests the inherent capabilities of language models, free from the influence of examples. Our $\mathcal{P}^3$ method further enhances the reliability of zero-shot evaluations by mitigating prompt instability, making it a more stable and valuable metric.

A.3 Selected η

The hyperparameter η selections are in Table 6.

Dataset	13B	70B
IMDb	37	42
AGnews	59	497
Amazon	4	17
DBpedia	62	4
ISEAR	8	5
SST-2	6	7
Yahoo	65	4

Table 6: Hyperparameter settings for different datasets and models.

A.4 Dataset Size

The size of datasets we used in the paper is shown in Table 7.

IMDb	25000
AGnews	7600
Amazon	400000
DBpedia	70000
ISEAR	7666
SST-2	1821
Yahoo	60000

Table 7: The size of datasets.

A.5 Behavior Differences of Language Models

Although, as discussed in Section 5.1, the use of the $\mathcal{P}^3$ method consistently yields benefits, the classification performance of tokens at different positions varies. In this regard, the 13B and 70B

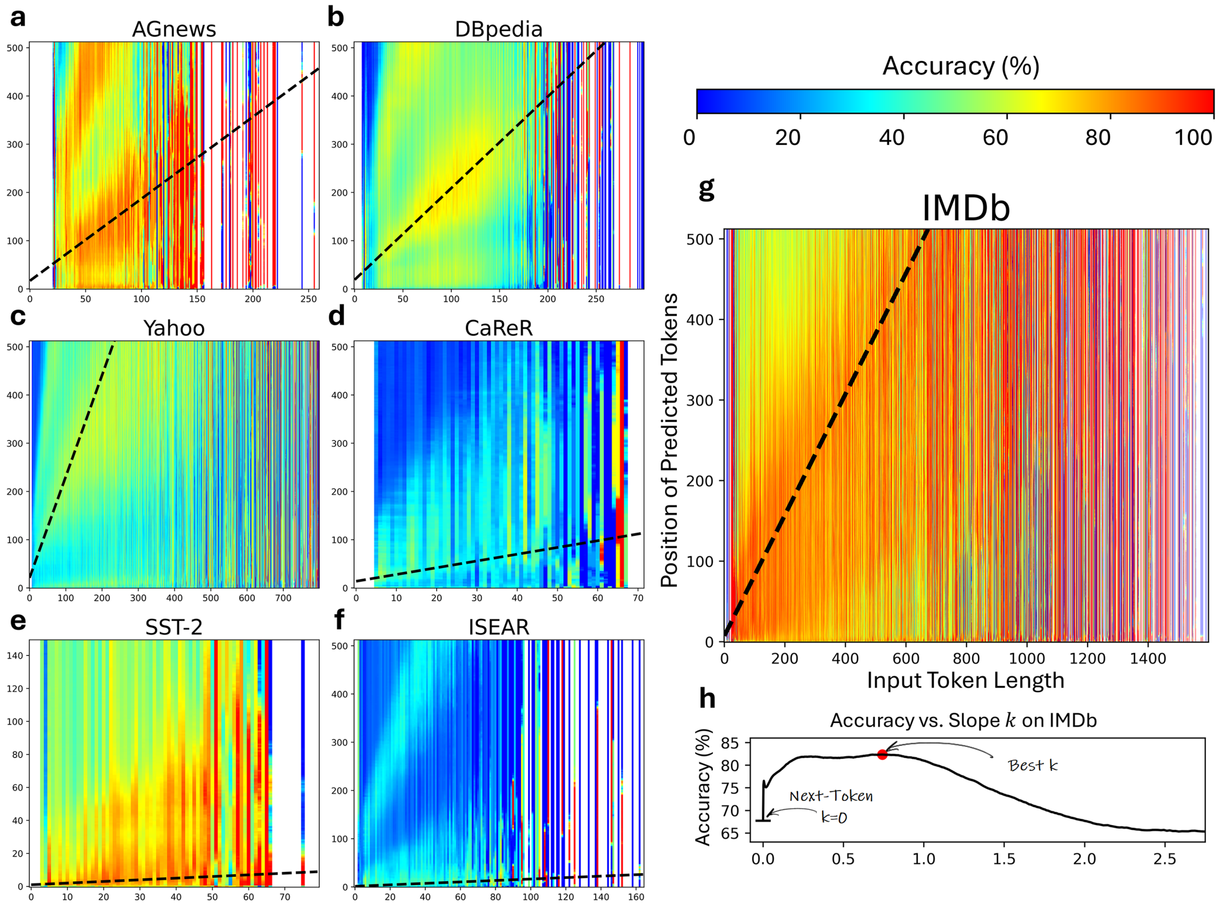


Figure 6: **abcdefg**: Accuracy heatmaps for each dataset on 13b. In each subplot, the x-axis represents the text length l_t (number of tokens in the text), and the y-axis represents the position of the token used for classification (0 represents the next-token). **h** shows the average accuracy of the IMDb dataset at different slopes k (with intercept $b = -10$).

models exhibit different behaviors. Specifically, the performance of the 13B model is not only position-dependent but also influenced by the number of input tokens, whereas the 70B model does not exhibit this phenomenon.

As shown in Figure 6, the distribution of well-performing (or poorly-performing) tokens in the 13B model follows a radial pattern, proportional to the number of input tokens. In contrast, the performance of tokens in the 70B model depends solely on their position and is independent of the input token length (exhibits vertical striping patterns).

In this paper, we treated it as an observed fact and leveraged it as a characteristic of language models. However, the exact underlying cause of this difference is challenging to determine due to the lack of transparency in LLaMA2's training details. We hypothesize that this distinction might arise from differences in the training process:

Stable positions in 70b: If the model underwent direct instruction-tuning or RLHF (Reinforcement Learning with Human Feedback) without distillation, the potential well-performing token positions would likely remain stable. This is because, for most classification tasks, answer lengths are consistent regardless of input variation.

Length-dependent positions in 13b: If the model underwent distillation from a larger model during instruction-tuning or RLHF, it might have inherited reasoning behaviors linked to input length, causing the observed correlation between input length and potential well-performing token positions. We would like to interpret this result as a possible consequence of discrepancies in training processes and parameter scales. While we have not confirmed this hypothesis, we believe deeper exploration could lead to a more rigorous explanation. Since this behavior difference does not impact our primary claims regarding the mitigation of prompt brittleness, we treated it as an existing fact in our experiments.

The two models exhibit behavior differences, prompting us to adapt our method accordingly. For LLaMA2-13b, we observed a radial pattern in its plot, prompting us to set η as the angle and use a line with slope $\tan(\eta)$ as our final result. For LLaMA2-70b, with a stable separability plot independent of input length, we define [0, η) as a fixed range for voting and applying calibration to enhance the results.

Furthermore, we found that using the votes of all 513 tokens obtained via $\mathcal{P}^3$ method as pseudo-labels also produces patterns similar to those of the true labels. However, we did not pursue further investigation into this matter.

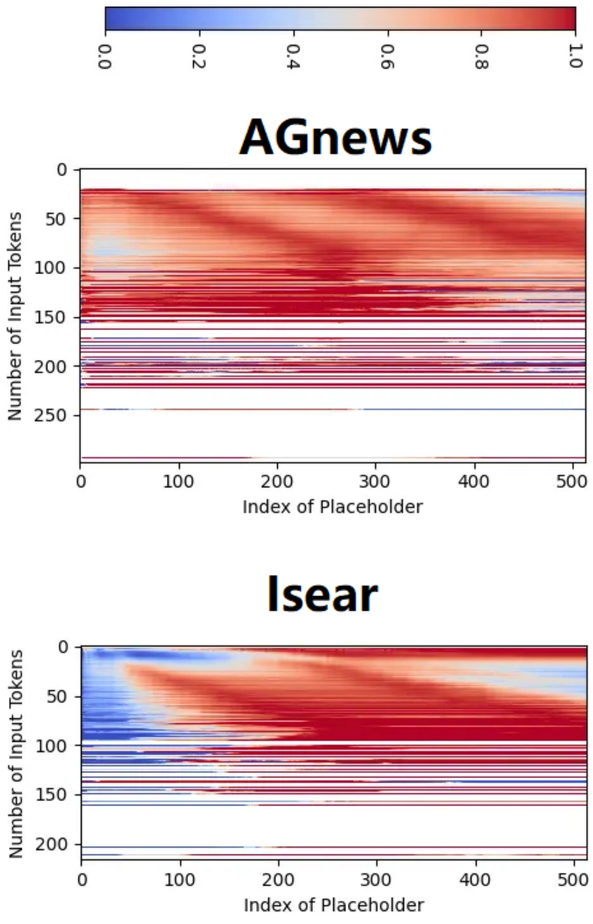


Figure 7: Consistency plot with pseudo-labels, where the pseudo-labels are determined by the voting results of 513 tokens.

A.6 FLOPs Estimation for Transformer Models

To estimate the FLOPs for the forward pass of a Transformer model, we consider the main computational components: attention and feed-forward networks (FFN).

For the attention mechanism, the input matrices Q, K, and V (with dimensions (B, s, h)) go through linear transformations to produce W_q, W_k, and W_v:

$$\text{QKV transformation FLOPs} = 6Bsh^2$$

Next, the attention matrix is computed by multiplying the query matrix Q with the transposed key matrix K^T, resulting in a matrix of size (s, s):

$$\text{Attention matrix computation FLOPs} = 2Bs^2h$$

Then, the attention values are computed by multiplying the attention matrix with the value matrix V:

$$\text{Attention over values FLOPs} = 2Bs^2h$$

Finally, a linear projection is applied to the resulting matrix:

$$\text{Post-attention linear projection FLOPs} = 2Bsh^2$$

For the FFN, where the input dimension h is transformed to $4h$ and then back to h, the FLOPs for this part are:

$$\text{FFN FLOPs} = 16Bsh^2$$

The language model head performs a linear transformation from the hidden size h to the vocabulary size V, resulting in the following FLOPs:

$$\text{LM head FLOPs} = 2BshV$$

The total FLOPs per layer, considering attention and FFN, is the sum of the components above:

$$\text{FLOPs per layer} = 24Bsh^2 + 4Bs^2h$$

Including the language model head, the total FLOPs per step for the forward pass in a model with l layers is:

$$\text{Total FLOPs per step} = l\left(24Bsh^2 + 4Bs^2h\right) + 2BshV$$

Based on the formula, we also computed the case using the selected ηs (Table 8), as well as the case where all 512 placeholder tokens were used in our experiments (Table 9).

As can be seen, the overall computational cost is proportional to the number of tokens, indicating that our method does not impose a significant overhead.

Dataset	13B		70B	
	NTP	$\mathcal{P}^3$	NTP	$\mathcal{P}^3$
Amazon	2.82	3.00	14.26	16.47
IMDb	8.31	14.73	41.91	47.41
AGnews	1.62	4.31	8.20	73.32
DBpedia	2.16	6.24	10.94	11.46
Yahoo	3.67	11.78	18.56	19.08
SST-2	0.67	0.72	3.38	4.28
ISEAR	0.70	0.79	3.57	4.22

Table 8: Total Tflops calculation for the selected η.

Dataset	13B		70B	
	NTP	$\mathcal{P}^3$	NTP	$\mathcal{P}^3$
Amazon	2.82	16.17	14.26	81.48
IMDb	8.31	21.84	41.91	109.69
AGnews	1.62	14.94	8.20	75.30
DBpedia	2.16	15.50	10.94	78.09
Yahoo	3.67	17.06	18.56	85.87
SST-2	0.67	13.96	3.38	70.37
ISEAR	0.70	13.99	3.57	70.57

Table 9: Total Tflops calculation for use of all 512 placeholder tokens in the experiment.

A.7 About Multi-Token Class Names

A common challenge in zero-shot classification, particularly when dealing with class names that span multiple tokens, is how to calculate the class score from the prediction of a single token. This issue arises due to the way current language models generate and calibrate token probabilities.

In typical zero-shot approaches based on token generation probabilities, directly multiplying probabilities for individual tokens within a sequence often fails to yield reliable classification scores. This problem is particularly pronounced when class descriptions consist of multiple tokens. Some studies address this challenge by integrating external resources, such as knowledge bases or validation datasets, which can help identify relevant tokens or synonyms associated with each class. For example, one common method is to sum the probabilities of all relevant tokens (e.g., "good" and "positive") to compute a cumulative score. However, such methods often require extensive preprocessing, which can limit their applicability in more dynamic or real-time scenarios.

In this work, we focus on exploring the robustness of subsequent token predictions. To ensure consistency with previous research and simplify the setup, we adopted the widely used practice of mapping each class to a single token (e.g., "good" for the positive class and "bad" for the negative class), as seen in standard datasets.

An interesting aspect of our $\mathcal{P}^3$ method, however, is its ability to naturally handle multi-token class names. By aggregating probabilities across subsequent tokens, $\mathcal{P}^3$ can provide a more coherent approach to multi-token class labels. Rather than relying on the prediction of a single token, $\mathcal{P}^3$ aggregates predictions across tokens, enabling it to better align with natural language. For instance, in cases where tokens like "ppi" or "ness" are unlikely to appear in the immediate context, $\mathcal{P}^3$ can still effectively aggregate across related tokens, such as "good" and "positive," to compute a more accurate class score.

This approach presents an important advantage over traditional methods, offering a more scalable and flexible framework for zero-shot classification tasks, particularly in scenarios involving complex or multi-token class names.

A.8 Position Plot

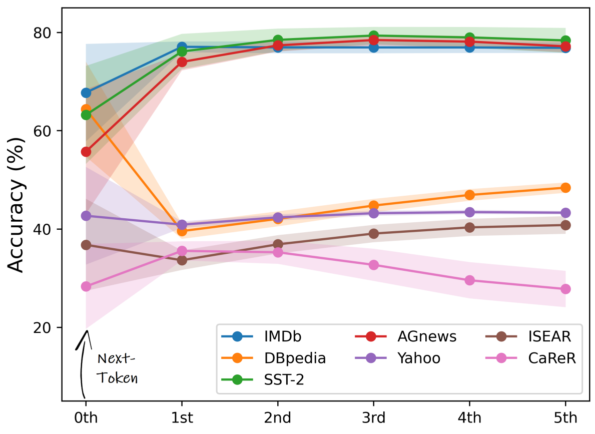


Figure 8: Accuracy and cross-prompt standard deviation for each dataset using predicted tokens at fixed positions on 13b. The x-axis shows token position. Dots indicate average accuracy and shaded areas represent standard deviation.

A.9 Prompts used in the experiment.

To evaluate the effectiveness and robustness of our Placeholding Parallel Prediction method, we conduct extensive experiments on seven datasets with various kinds of prompts, including empty prompts. The specific prompt settings for different datasets are shown in Table 10-16.

ID	Template
1	title\ntext\nMy feedback to it is
2	title\ntext\nOverall, my feedback to it is
3	title\ntext\nAfter considering all aspects, my feedback to it is
4	title\ntext\nAfter considering all aspects, my viewpoint is
5	title\ntext\nReflecting on the above, my viewpoint is
6	title\ntext\nOverall, my perspective on it is
7	title\ntext\nOverall, my takeaway is
8	title\ntext\nIn summary, I would say
9	title\ntext\nConsidering the details provided, my emotional reaction is
10	title\ntext\nTaking into account the experience shared, my viewpoint is
11	title\ntext\nReflecting on the content, my emotional stance is
12	title\ntext\nGiven the information above, my perspective on it is
13	title\ntext\nAnalyzing the feedback, my emotional assessment is
14	title\ntext\nBased on the review, my overall sentiment impression is
15	title\ntext\nWeighing up the insights, my sentiment conclusion is
16	title\ntext\nAfter thoroughly considering the review, my sentiment perspective is
17	Text: title text\nSentiment:
18	Text: title text\nSentiment Analysis: The overall sentiment is
19	title\ntext\nAll in all, it was
20	title\ntext\nIn summary, it was
21	title\ntext\nIn essence, it was
22	title\ntext\nIn conclusion, it was
23	title\ntext\nTo sum up, it's
24	title\ntext All in all
25	title\ntext Just
26	title\ntext It was
27	title\ntext It is
28	title\ntext That is
29	title\ntext That's
30	title\ntext But it is
31	title\ntext

Table 10: Prompt list for Amazon Review Polarity dataset.

ID	Template
1	[text] My feedback to the film is
2	[text] Overall, my feedback to the film is
3	[text] After considering all aspects, my feedback to the film is
4	[text] After considering all aspects, my viewpoint is
5	[text] Reflecting on the above, my viewpoint is
6	[text] Overall, my perspective on the film is
7	[text] Overall, my takeaway is
8	[text] In summary, I would say
9	[text] I think it is
10	[text] Overall, I think it is
11	[text] Considering everything, my feedback is
12	[text] Considering everything, I think it is
13	[text] After thinking about it, my feedback is
14	[text] Overall, I see it as
15	[text] Taking all factors into account, my assessment of it is
16	[text] Considering the details provided, my emotional reaction is
17	[text] Taking into account the experience shared, my sentiment is
18	[text] Reflecting on the content, my emotional stance is
19	[text] Given the information above, my sentiment evaluation is
20	[text] Analyzing the feedback, my emotional assessment is
21	[text] Based on the review, my overall sentiment impression is
22	[text] Weighing up the insights, my sentiment conclusion is
23	[text] After thoroughly considering the review, my sentiment perspective is
24	Text: [text] Sentiment:
25	Text: [text] Sentiment Analysis: The overall sentiment is
26	[text] All in all, the film was
27	[text] In summary, the film was
28	[text] In essence, the film was
29	[text] In conclusion, the film was
30	[text] To sum up, the film was
31	[text] All in all
32	[text] Just
33	[text] It was
34	[text] It is
35	[text] That is
36	[text] That's
37	[text] But it is
38	[text]

Table 11: Prompt list for IMDb dataset.

ID	Template
1	[text] My feedback to the film is
2	[text] Overall, my feedback to the film is
3	[text] After considering all aspects, my feedback to the film is
4	[text] After considering all aspects, my viewpoint is
5	[text] Reflecting on the above, my viewpoint is
6	[text] Overall, my perspective on the film is
7	[text] Overall, my takeaway is
8	[text] In summary, I would say
9	[text] I think it is
10	[text] Overall, I think it is
11	[text] Considering everything, my feedback is
12	[text] Considering everything, I think it is
13	[text] After thinking about it, my feedback is
14	[text] Overall, I see it as
15	[text] Taking all factors into account, my assessment of it is
16	[text] Considering the details provided, my emotional reaction is
17	[text] Taking into account the experience shared, my sentiment is
18	[text] Reflecting on the content, my emotional stance is
19	[text] Given the information above, my sentiment evaluation is
20	[text] Analyzing the feedback, my emotional assessment is
21	[text] Based on the review, my overall sentiment impression is
22	[text] Weighing up the insights, my sentiment conclusion is
23	[text] After thoroughly considering the review, my sentiment perspective is
24	Text: [text] Sentiment:
25	Text: [text] Sentiment Analysis: The overall sentiment is
26	[text] All in all, the film was
27	[text] In summary, the film was
28	[text] In essence, the film was
29	[text] In conclusion, the film was
30	[text] To sum up, the film was
31	[text] All in all
32	[text] Just
33	[text] It was
34	[text] It is
35	[text] That is
36	[text] That's
37	[text] But it is
38	[text]

Table 12: Prompt list for SST-2 dataset.

ID	Template
1	[title] [text] This topic is about
2	[title] [text] The label that best describes this news article is
3	[title] [text] This piece of news is regarding
4	[title] [text] The news article is about
5	[title] [text] Central themes of this news piece encompass
6	[title] [text] The central theme of this article revolves around
7	[title] [text] It can be labeled as
8	[title] [text] Its category is
9	[title] [text] In this article, it talks about
10	[title] [text] The content is a kind of
11	[title] [text] I think the news can be classified as
12	[title] [text] I would classify it as
13	[title] [text] Based on the description, its category is
14	[title] [text] In this context, the content falls into the category of
15	Text: [title] [text] Category:
16	Text: [title] [text] Topic Classification: The overall topic is
17	[title] [text] All in all, it was
18	[title] [text] In summary, it was
19	[title] [text] In essence, it was
20	[title] [text] In conclusion, it was
21	[title] [text] To sum up, it’s
22	[title] [text] All in all
23	[title] [text] Just
24	[title] [text] It was
25	[title] [text] It is
26	[title] [text] That is
27	[title] [text] That’s
28	[title] [text] But it is
29	[title] [text]

Table 13: Prompt list for AGnews dataset.

ID	Template
1	[title] [text] The category of [title] is
2	[title] [text] The label that best describes [title] is
3	[title] [text] So, [title] is
4	[title] [text] In this sentence, [title] is
5	[title] [text] [title] is a kind of
6	[title] [text] [title] can be classified as
7	[title] [text] [title] is an example of
8	[title] [text] [title] belongs to
9	[title] [text] I think [title] is
10	[title] [text] I would classify [title] as
11	[title] [text] Based on the description, its category is
12	[title] [text] In this context, [title] falls into the category of
13	Text: [title] [text] Category:
14	Text: [title] [text] Topic Classification: The overall topic is
15	[title] [text] All in all, it is
16	[title] [text] In summary, it is
17	[title] [text] In essence, it is
18	[title] [text] In conclusion, it is
19	[title] [text] To sum up, it's
20	[title] [text] All in all
21	[title] [text] Just
22	[title] [text] It was
23	[title] [text] It is
24	[title] [text] That is
25	[title] [text] That's
26	[title] [text] But it is
27	[title] [text]

Table 14: Prompt list for DBpedia dataset.

ID.	Template
1	[title] [text] This topic is about
2	[title] [text] The label that best describes this question is
3	[title] [text] This issue is regarding
4	[title] [text] This discussion is about
5	[title] [text] This discussion is regarding
6	[title] [text] This issue is about
7	[title] [text] The label that best describes this issue is
8	[title] [text] I would classify this question as
9	[title] [text] It can be labeled as
10	[title] [text] Overall, The most fitting category for this issue is
11	[title] [text] The content is associated with
12	[title] [text] I think it belongs to
13	[title] [text] I would classify it as
14	[title] [text] This issue falls into the category of
15	Text: [title] [text] Category:
16	Text: [title] [text] Topic Classification: The overall topic is
17	[title] [text] All in all, it was
18	[title] [text] In summary, it was
19	[title] [text] In essence, it was
20	[title] [text] In conclusion, it was
21	[title] [text] To sum up, it’s
22	[title] [text] All in all
23	[title] [text] Just
24	[title] [text] It was
25	[title] [text] It is
26	[title] [text] That is
27	[title] [text] That’s
28	[title] [text] But it is
29	[title] [text]

Table 15: Prompt list for Yahoo dataset.

ID	Template
1	[text] In summary, I would say
2	[text] I think it is
3	[text] Overall, I think it is
4	[text] Considering everything, I think it is
5	[text] Overall, I see it as
6	[text] In summary, I would say
7	[text] I feel
8	[text] Overall, I feel
9	[text] Overall, my feeling towards it is
10	[text] This text expresses
11	[text] It is a feeling of
12	[text] The sentiment is
13	[text] It is
14	[text] This conveys a sense of
15	[text] I am
16	[text] The overall impression is
17	[text] From my perspective, it is
18	[text] In my view, the feeling is
19	[text] This passage makes me feel
20	[text] It seems to evoke a feeling of
21	[text] This text primarily conveys
22	[text] From this, I sense an emotion of
23	[text] It can be interpreted as expressing
24	[text] The underlying emotion seems to be
25	[text] This narrative elicits
26	[text] Feeling-wise, this comes across as
27	[text] This evokes
28	[text] The emotional tone here is
29	[text] This story is imbued with
30	[text] One could interpret this as
31	[text] This text leaves the impression of
32	Text: [text] Emotion:
33	Text: [text] Emotion Recognition: The overall emotion is
34	[text] All in all, it was
35	[text] In summary, it was
36	[text] In essence, it was
37	[text] In conclusion, it was
38	[text] To sum up, it was
39	[text] All in all
40	[text] Just
41	[text] It was
42	[text] It is
43	[text] That is
44	[text] That’s
45	[text] But it is
46	[text]

Table 16: Prompt list for ISEAR dataset.