# MSRA PROJECT: LLM TEACHES ITSELF TO AUTOMATICALLY OPTIMIZE PROMPTS

**Yu Wang**[1*] **Nan Yang**[2†]
[1]University of Science and Technology of China [2]Microsoft
terencewang0809@gmail.com nanya@microsoft.com

**ABSTRACT**

Prompt engineering, which can help Large Language Model(LLM) achieve performance improvements, has been gaining significant traction lately. Compared to prompt tuning, prompt engineering offers greater interpretability and does not require model open-source. In particular, *automatic prompt optimization*, which utilizes LLM to self-reflect and refine prompts, has demonstrated good performance in recent studies. In this paper, we surveyed the existing methods of *automatic prompt optimization* and explored some fundamental reasons behind the effectiveness of *automatic prompt optimization*. Based on APO paradigm, we propose a new framework utilizing curriculum learning and devise a novel metric to select the best prompt. The code is available at https://github.com/Wloner0809/msra-project.

***K*eywords** Automatic Prompt Optimization · Curriculum Learning · LLM

## 1 Introduction

Nowadays, prompt has become a crucial factor in enabling LLM to perform effectively[1, 2]. Prompt can be categorized into *soft prompt* and *hard prompt*, where *soft prompt* represents continuous language embeddings that do not correspond to any human-readable tokens and *hard prompt* signifies manually handcrafted text prompts with discrete input tokens[3, 4]. Some studies have demonstrated the efficacy of tuning soft prompt, which we call prompt tuning[4]. However, such works assume that we have access to internal state variables of the LLM while we often utilize LLM through an API. In addition, soft prompt generally lacks interpretability compared to hard prompt. Therefore, we focus on optimizing hard prompt.

Prompt engineering, which aims to maximize the performance of LLM through discrete natural language, has become increasingly important in LLM applications[5]. Nevertheless, devising an optimal prompt involves numerous trials and a substantial investment of manpower, resulting in inefficiencies. This arises due to the acute sensitivity of LLM to the prompt[6, 7]. Even if two prompts are semantically similar, the result can be totally different. Fortunately, LLM has self-reflection[8, 9] and In-Context Learning ability[10], thus we can utilize LLM to automatically optimize the prompt to reduce manpower costs[11].

For *automatic prompt optimization*, there are numerous classification methods. According to this paper[12], the current methods can be categorized based on resampling and self-reflection. Resampling-based methods include APE[13], GPO[14], while reflection-based methods encompass APO[15], OPRO[16], PromptAgent[17], PE2[18]. Furthermore, as stated in this paper[19], LLM-based prompt optimizers and traditional model optimizers share many similarities. Hence we can also categorize based on 'update direction' and 'update method'. Table 1 in paper[19] gives us an example.

Despite the abundance of methods available, they are all quite similar. Therefore, we choose the APO algorithm[15], which was among the first to utilize self-reflection, as the framework for our optimization. Inspired by the effectiveness of Curriculum Learning in model training[20, 21, 22], as well as the similarity between APO and model training, we

[*]Yu Wang is the student of this project.
[†]Nan Yang is the mentor of this project.

| Prompt optimizer | Update direction | | Update method | |
|---|---|---|---|---|
| | Gradient form | Momentum form | Prompt variation | Prompt refinement |
| APE | P | None | None | Generation |
| APO | P+R | None | None | Editing |
| OPRO | P+M | Recency | None | Generation |
| PE2 | P+M+R | Recency | Fixed | Generation |

Table 1: 'P' refers to prompt, 'M' refers to performance, 'R' refers to reflection

propose a new framework utilizing Curriculum Learning and APO to automatically optimize the prompt. Besides, drawing inspiration from [23], we design a new metric based on perplexity to select the best prompt.

Due to financial constraints, we do not opt to use GPT-4[24] as our model. Instead, we use open-source models, such as LlaMA[25], Mixtral[26], Qwen[27]. We test our method on liar dataset[28], which was also used in APO[15], and achieve some performance improvements compared to the baseline. In addition, we also test our method on SST-2 dataset[29].

**Our main contributions are as follows:**

- We propose a new framework based on APO and Curriculum Learning and design a new metric based on perplexity.
- We demonstrate the effectiveness of our method for open-source LLMs.

## 2 Background

In this section, we first introduce the normal formulation of prompt engineering in section 2.1. Following that, we will discuss various approaches of *automatic prompt optimization* in section 2.2. Section 2.3 is dedicated to exploring pertinent concepts regarding Curriculum Learning. We introduce perplexity briefly in section 2.4. Building on the background outlined above, we present our own method in section 3.

### 2.1 Prompt Engineering

The objective of prompt engineering is to find a natural language prompt $p^\star$ that maximizes task performance on a given dataset $D$. Specifically, we assume that the dataset $D$ can be represented as $D = \{(x, y)\}$ and the whole dataset includes $D_{train}$, $D_{valid}$, $D_{test}$. Then the problem can be formulated as:

$$p^\star = \underset{p \sim M_{prompt}}{\arg\max}\ \mathbb{E}_{(x,y)\sim D_{valid}}[F(M_{task}(x; p), y)] \tag{1}$$

where $M_{prompt}$ represents the model that optimizes prompts, $M_{task}$ is the task model, $p$ denotes the prompt generated by $M_{prompt}$, $F$ stands for an evaluating function based on some metric.

### 2.2 Automatic Prompt Optimization

**APE[13]** APE(Automatic Prompt Engineer) is a resample-based method that utilizes LLM to generate semantically similar prompts and select the best prompt. Its core idea revolves around continuously filtering and manipulating high-quality prompts. Figure 1 shows how APE works.

**APO[15]** APO is the first framework that utilizes the self-reflection ability of LLMs to generate the 'text gradient' and edit the task prompt. The goal of APO is to continually prompt LLM to reflect and generate feedbacks for optimization. Figure 2 shows how APO works.

**OPRO[16]** OPRO(Optimization by PROmpting) makes use of the full optimization trajectory, which includes past solutions paired with their optimization scores. Based on the optimization trajectory, OPRO regenerates new prompts through implicit reflection for optimization. Figure 3 shows how OPRO works.

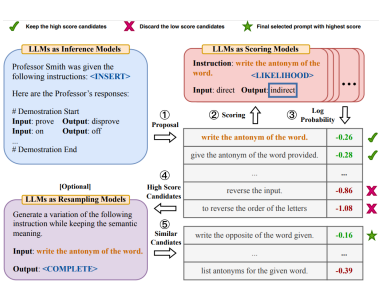


Figure 1: APE workflow.

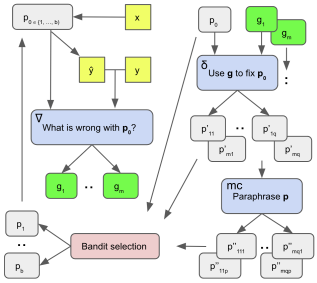


Figure 2: APO workflow.

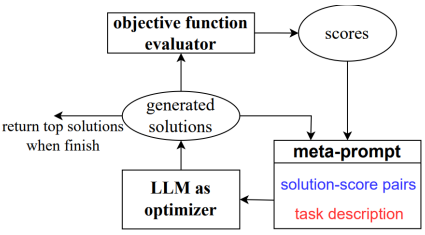


Figure 3: OPRO workflow.

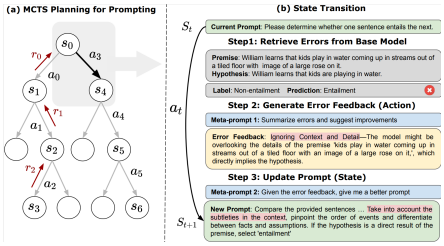


Figure 4: PromptAgent workflow.

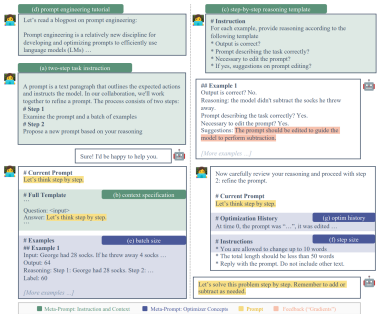

Figure 5: PE2 workflow.

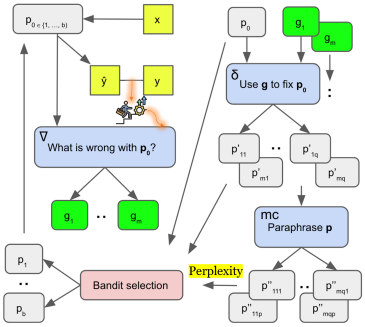


Figure 6: Our method workflow.

**PromptAgent[17]** PromptAgent leverages the self-reflection and planning ability of LLM, formulating the *automatic prompt optimization* problem as a MDP(Marcov Decision Process). It seamlessly incorporates MCTS(Monte Carlo Tree Search) to strategically optimize the prompting process. Figure 4 shows how PromptAgent works.

**PE2[18]** PE2(Prompt Engineering a Prompt Engineer) focuses on constructing a perfect 'meta-prompt' to guide LLM to generate better prompts. It incorporates two-step task description, context specification, step-by-step reasoning template into 'meta-prompt'. Figure 5 shows how PE2 works.

### 2.3 Curriculum Learning

Curriculum Learning is a training strategy that mimics the human learning process by training models from easy to difficult tasks. This idea can be seen as a particular continuation method.[20], which first optimizes smoother version of the problem and then gradually considers less smoothing version. This approach facilitates the identification of the global minimum.

The core issue in Curriculum Learning is to obtain a ranking function. Currently, most Curriculum learning approaches are designed based on the framework of *Difficulty Measurer + Training Scheduler*[21]. *Difficulty Measurer* decides the difficulty level of each data point, while *Training Scheduler* decides the sequence of data subsets throughout the training process based on the judgment from the *Difficulty Measurer*. In this work, We assign the role of the *Difficulty Measurer* to the LLM.

### 2.4 Perplexity

If we have a tokenized sequence $X = (x_0, x_1, \cdots, x_t)$, then the perplexity of $X$ is

$$PPL(X) = \exp\{-\frac{1}{t}\sum_{i}^{t}\log p_\theta(x_i|x_{<i})\} \tag{2}$$

where $\log p_\theta(x_i|x_{<i})$ is the log-likelihood of the $i_{th}$ token conditioned on the preceding tokens $x_{<i}$.

# 3 LLM teaches itself to automatically optimize prompts

The main framework(Algorithm 1) is based on [15]. In this section, we will demonstrate how to incorporate Curriculum Learning and perplexity metric into APO framework. Figure 6 shows the workflow of our method.

**Algorithm 1** APO

**Require:** $p_0$: initial prompt, $b$: beam width, $r$: search depth, $m$: metric function
1: $B_0 \leftarrow \{p_0\}$
2: **for** $i \leftarrow 1$ to $r-1$ **do**
3: $\quad C \leftarrow \emptyset$
4: $\quad$ **for all** $p \in B_i$ **do**
5: $\quad\quad C \leftarrow C \cup Expand(p)$
6: $\quad$ **end for**
7: $\quad B_{i+1} \leftarrow Select_b(C, m)$
8: **end for**
9: $\hat{p} \leftarrow argmax_{p \in B_r} m(s)$
10: **return** $\hat{p}$

### 3.1 Add Curriculum Learning into Expand

Due to LLM's exceptional general capabilities, it is regarded as a great zero-shot ranker[30]. Therefore, we adopt the LLM as the *Difficulty Measurer* in Expand algorithm, where LLM is utilized to output the ranking of each example's difficulty level. Thus, as shown in Algorithm 2, we rank the errors based on difficulty level before generating the gradients.

**Algorithm 2** $Expand(\cdot)$ - line 5 of Algorithm 1

**Require:** $p$: prompt candidate, $\mathcal{D}_{tr}$: train data
1: Sample minibatch $\mathcal{D}_{mini} \subset \mathcal{D}_{tr}$
2: Utilize LLM to rank $\mathcal{D}_{mini}$ based on the difficulty level
3: Evaluate prompt $p$ on minibatch $\mathcal{D}_{mini}$ and collect errors $e = \{(x_i, y_i) : (x_i, y_i) \in \mathcal{D}_{mini} \wedge LLM_p(x_i) \neq y_i\}$
4: Utilize LLM to rank errors $e$ based on the difficulty level
5: Get gradients: $\{g_1, ..., g_m\} = LLM_\nabla(p, e)$
6: Use the gradients to edit the current prompt: $\{p'_{i1}, ..., p'_{iq}\} = LLM_\delta(p, g_i, e)$
7: Get more monte-carlo successors: $\{p''_{ij1}, ..., p''_{ijm}\} = LLM_{mc}(p'_{ij})$
8: **return** $\{p'_{11}, ..., p'_{mq}\} \cup \{p''_{111}, ..., p''_{mqp}\}$

### 3.2 Add Perplexity Metric into Select

This metric is derived from the inspiration provided by [23]. We can conclude that prompts with lower perplexity tend to perform better in a given task. Therefore, we add perplexity as an additional metric to help select prompts. The whole process can be seen in Algorithm 3.

**Algorithm 3** $Select(\cdot)$ - line 7 of Algorithm 1

**Require:** $n$ prompts $p_1, ..., p_n$, dataset $\mathcal{D}_{tr}$, max expansion factor $num$, beam width $b$, metric function $m$
1: Initialize: $S_0 \leftarrow \{p_1, \ldots, p_n\}$
2: Compute the perplexity of each prompt of $S_0$
3: **if** $|S_0| > num$ **then**
4: $\quad$ Exclude the prompts with higher perplexity, get a new $S$
5: **end if**
6: Sample $\mathcal{D}_{sample} \subset \mathcal{D}_{tr}$
7: Evaluate $p \in S$ with $m(p, \mathcal{D}_{sample})$
8: Exclude the prompts with lower scores, if the scores of two prompts are the same, select the prompt with lower perplexity. Get a new $S$
9: return $S$

| Model | F1 | | |
|---|---|---|---|
| | Initial prompt | APO | Our Method |
| Llama-3-70B-chat-hf | 0.5033 | 0.5813 | **0.6356**↑ |
| Mixtral-8x22B-Instruct-v0.1 | 0.5054 | 0.5553 | **0.5900**↑ |
| Qwen1.5-110B-Chat | 0.5054 | 0.5727 | **0.6291**↑ |

Table 2: Liar Dataset Results

| Model | F1 | | |
|---|---|---|---|
| | Initial prompt | APO | Our Method |
| Llama-3-70B-chat-hf | 0.9427 | 0.9450 | **0.9484**↑ |
| Mixtral-8x22B-Instruct-v0.1 | 0.9530 | **0.9553** | 0.9472↓ |
| Qwen1.5-110B-Chat | **0.9587** | 0.9472 | 0.9404↓ |

Table 3: SST-2 Dataset Results

## 4 Experiments

Due to time and financial constraints, we only chose two benchmark NLP tasks to evaluate the effectiveness of our method.

### 4.1 Datasets

We chose to use Liar[28] dataset and SST-2[29] dataset. Liar is a publicly available dataset for fake news detection with 4000 statements, context and lie labels. SST-2 is a dataset for sentiment analysis of given sentences, with 67,350 examples in the training set, 873 examples in the development set, and 1,821 examples in the test set.

### 4.2 Baselines

We evaluated prompts with open-source models. Three types of prompts we tested are as follows.

- Initial prompt
- Prompt optimized by original APO[15] algorithm
- Prompt optimized by our method(APO + Curriculum Learning + Perplexity)

More details about prompts are deferred in Appendix 6.

### 4.3 Setup

**Models** For the models, we utilized three open-source large models' APIs provided by the together.ai* website, which are Llama-3-70B-chat-hf, Mixtral-8x22B-Instruct-v0.1, Qwen1.5-110B-Chat. $M_{task}$ and $M_{prompt}$ are the same model. Due to time and financial constraints, all of the reported results(F1 score as our final metric) came from single experimental trial.

**Hyperparameters** As for the hyperparameters, we used a minibatch size of $|D_{mini}| = 24$, beam size $b = 12$ and ran the algorithm for 2 optimization steps. For each step, we sampled groups of 4 errors at a time to generate the gradients. Besides, we set the temperature 0.0 during classification and 0.7 in all other contexts.

### 4.4 Results & Analysis

**Main results** As shown in table 2, our method outperforms the APO algorithm on more challenging datasets. However, when the task is relatively simple for the model, the sensitivity of the LLM to the prompt may decrease. As demonstrated in table 3, our method does not show effectiveness and even performs worse than the initial prompt. Therefore, we can conclude that our method works when the task is relatively difficult for the model.

*https://api.together.xyz/models

**Analysis** During the experiments, we found that the prompts with the lowest perplexity did not necessarily perform better. For example, as shown in figure 7, prompts with pretty low perplexity are actually not meaningful. Thus, we set a bound on perplexity when filtering or selecting prompts in practice.

```
# Task
‘ and ’

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:
```

Figure 7: prompt with lowest perplexity

## 5 Conclusion

Through this project, we reviewed numerous papers related to automatic prompt optimization, In-Context Learning, and various aspects of large language models (LLMs). This comprehensive review sparked the idea to integrate Curriculum Learning with APO framework to optimize prompts more effectively. During our experiments, we noticed that using the original metric often resulted in many prompts with identical metrics. To address this, we investigated and adopted perplexity as an auxiliary metric for prompt selection.

The experimental results demonstrate that our approach performs well on more challenging datasets. However, for simpler datasets, where the performance of LLMs is less influenced by prompts, our method appears less effective. This is likely because the inherent sensitivity of LLMs to prompts diminishes when the task is straightforward.

## 6 Limitations

- Due to network instability, we just ran a few rounds.
- Instead of using GPT-4, we chose to utilize the best available open-source models.
- The experiments have inherent randomness, and the number of repetitions was insufficient.
- Hyperparameters were not meticulously tuned; instead, a single set of hyperparameters was selected for all experiments.

## References

[1] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.

[2] Tianyu Gao. Prompting: Better ways of using language models for nlp tasks. *The Gradient*, 2021.

[3] Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Advances in Neural Information Processing Systems*, 36, 2024.

[4] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

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

[6] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR, 18–24 Jul 2021.

[7] Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*, 2023.

[8] Eric Jang. Can llms critique and iterate on their own outputs? *evjang.com*, Mar 2023.

[9] Noah Shinn, Beck Labash, and Ashwin Gopinath. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.

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

[11] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020.

[12] Ruotian Ma, Xiaolei Wang, Xin Zhou, Jian Li, Nan Du, Tao Gui, Qi Zhang, and Xuanjing Huang. Are large language models good prompt optimizers? *arXiv preprint arXiv:2402.02101*, 2024.

[13] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.

[14] Moxin Li, Wenjie Wang, Fuli Feng, Yixin Cao, Jizhi Zhang, and Tat-Seng Chua. Robust prompt optimization for large language models against distribution shifts. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1539–1554, Singapore, December 2023. Association for Computational Linguistics.

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

[16] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

[17] Xinyuan Wang, Chenxi Li, Zhen Wang, Fan Bai, Haotian Luo, Jiayou Zhang, Nebojsa Jojic, Eric P Xing, and Zhiting Hu. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*, 2023.

[18] Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*, 2023.

[19] Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. Unleashing the potential of large language models as prompt optimizers: An analogical analysis with gradient-based model optimizers. *arXiv preprint arXiv:2402.17564*, 2024.

[20] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[21] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.

[22] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

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

[24] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[25] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[26] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[27] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[28] William Yang Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[29] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 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, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[30] Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems, 2024.

# Appendix

### Optimal Prompts

Figure 8 shows the initial prompt and optimal prompts for liar dataset and APO framework. Figure 9 shows the initial prompt and optimal prompts for liar dataset and our method. Figure 10 shows the initial prompt and optimal prompts for sst-2 dataset and APO framework. Figure 11 shows the initial prompt and optimal prompts for sst-2 dataset and our method.

Initial prompt:
# Task
Determine whether the Statement is a lie (Yes) or not (No) based on the Context and other information.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Llama-3-70b-chat-hf:
# Task
Determine whether the given Statement is trustworthy or not. Label it as 'Deceptive' (Yes) if it intentionally provides false, inaccurate, or misleading information, or 'Accurate' (No) if it is a genuine and reliable statement. When making your judgment, take into account the provided Context, Job title, State, and Party of the speaker, as well as any external information that may be relevant to the claim.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Mixtral-8x22B-Instruct-v0.1:
# Task
Consider the Statement, Job title, State, Party, and Context. Determine whether the Statement is a lie (Yes) or not (No), taking into account the potential biases or motivations of the speaker. Keep in mind that politicians or public figures may make statements that are technically true but misleading or taken out of context.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Qwen1.5-110B-Chat:
# Task
Examine the credibility of the assertion by considering the historical backdrop, shifts in regulations or personal viewpoints through time, the individual's professional position, their alignment with a political State and Party, notable events referenced, and all relevant information. Judge whether the assertion is misleading (Yes) or appears valid (No), taking into account chronological changes and the development of situations.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Figure 8: optimal prompts for liar dataset and original APO algorithm

Initial prompt:
# Task
Determine whether the Statement is a lie (Yes) or not (No) based on the Context and other information.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Llama-3-70b-chat-hf:
# Task
Ignore the Job title, State, and Party affiliation, and focus solely on the Context and Statement. Classify the Statement as either a factual inaccuracy or misleading information (Yes) or a true and accurate claim (No).

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Mixtral-8x22B-Instruct-v0.1:
# Task
Evaluate the Statement in light of the Context, Job title, State, Party, and any other relevant information. Ascertain whether the Statement is a verifiable fact (No) or a subjective viewpoint (Yes) that could be swayed by the speaker's political leanings or ideology, thereby potentially incorporating bias or partisanship.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Qwen1.5-110B-Chat:
# Task
Evaluate the truthfulness of the Statement by considering the Context, the Job title of the speaker, their Party affiliation, and any historical or widely known facts. Determine if the Statement is a misrepresentation or aligns with reality, and label it as a lie (Yes) if it is factually incorrect or misleading based on these factors, and not a lie (No) otherwise.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Figure 9: optimal prompts for liar dataset and our method

Initial prompt:
# Task
Identify whether the sentiment of the following opinion is positive (Yes) or negative (No).

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Llama-3-70b-chat-hf:
# Task
Consider the nuances of language and the context in which words are used to determine the overall sentiment of the given text. Look for phrases or sentences that may convey irony, sarcasm, or contrast to accurately identify whether the sentiment leans towards being positive or negative. Respond with 'Yes' for positive and 'No' for negative.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Mixtral-8x22B-Instruct-v0.1:
# Task
Analyze the sentiment of the following statement, considering both positive and negative aspects, and determine if the overall sentiment is favorable (Yes) or unfavorable (No).

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Qwen1.5-110B-Chat:
# Task
Assess the prevailing emotional tone of the passage as favorable (Yes) or unfavorable (No), being mindful of any coexisting positive and negative sentiments or situational contradictions. Consider phrases that signal sentiment reversals, such as "on the other hand" or "despite that", to render an accurate evaluation that incorporates the text's nuanced emotional balance.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Figure 10: optimal prompts for sst-2 dataset and original APO algorithm

Initial prompt:
# Task
Identify whether the sentiment of the following opinion is positive (Yes) or negative (No).

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Llama-3-70b-chat-hf:
# Task
Determine whether the given text expresses a positive or affirmative stance (Yes) or a negative or disapproving stance (No) towards a specific topic or idea.

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Mixtral-8x22B-Instruct-v0.1:
# Task
Evaluate the sentiment conveyed in the subsequent statement, taking into account both overt and subtle hints like sarcasm or irony. Decide whether the sentiment is positive (Yes) or negative (No).

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Qwen1.5-110B-Chat:
# Task
Your task is to analyze the sentiment within a given text by considering the full context and subtle implications of the words used. Decide if the opinion conveyed is positive (respond with 'Yes') or negative (respond with 'No'). Be meticulous about identifying any words or expressions that might introduce a contrasting sentiment. ## Example 1
Text: "an otherwise delightfully comic narrative"
Label: Yes

# Output format
Answer Yes or No as labels

# Prediction
Text: {{ text }}
Label:

Figure 11: optimal prompts for sst-2 dataset and our method