Task
Utilize LLMs to be an Academic Chair for top ML conferences.

Experiments
As the focus of the paper is nonparametric algorithms with broad applicability, there was no hyperparameter search for the proposed ProTeGi algorithm. For this specific task, I experimented on a list of combinations of hyperparameters for the algorithm in order to achieve higher generalization performance. Prior work includes some human prompting results, shown below.

Biased Prompt :

- *'"Analyze the reviews provided, decide if the paper in question would be accepted at an academic conference. The vast majority of papers are accepted. About 0.05 of papers are rejected at the conference. Answer only ACCEPT or REJECT"'*

Test F1 score : 0.75

Neutral Prompts (Initial prompt sections for APO):

- *Opt 1: Given the following reviews (text), determine if a paper would be accepted (Yes) or not (No) by an academic conference.*
- *Opt 2: Given the following reviews, determine if the paper being reviewed would be accepted at an academic conference.*

Opt 1 - Test F1 score : 0.5
Opt 2 - Test F1 score : 0.5

Settings
Given the default settings from the paper, a baseline evaluation of test F1 score for this task is **0.58**. Below is a list of settings that are unique or changed to this task.

- Data
    - source : venue_id = 'NeurIPS.cc/2024/Conference'
    - Training dataset : 800 accept + 800 reject
    - Test dataset: 800 accept + 800 reject
    - Test data : 100 accept + 100 reject
- Setup
    - Hyperparameters (different from the paper)
        - "model": ["gpt-4o-mini", "gpt-4.1-nano",]
        - "eval_model": ["gpt-4o-mini", "gpt-4.1-nano",]

- "beam_size": 5
- "n_test_exs": 200,
- ["n_gradients", "errors_per_gradient", "gradients_per_error", "steps_per_gradient", "mc_samples_per_step", "max_expansion_factor"]
    - Combo 0: [4, 4, 1, 1, 2, 8] , from the paper
    - Combo 1: [4, 4, 3, 2, 0, 6] (referred to as '44320-6' hereafter)
    - Combo 2: [4, 4, 3, 1, 0, 6]
    - Combo 3: [4, 4, 3, 1, 1, 6]
    - Combo 4: [4, 4, 3, 1, 2, 6]
    - Combo 5: [6, 4, 3, 2, 0, 6]
    - Combo 6: [6, 4, 3, 2, 0, 8]
    - Combo 7: [6, 4, 3, 2, 0, 8]
    - Combo 8: [6, 6, 1, 1, 2, 6]
- "evaluator": ["bf", "ucb"]
- ["eval_rounds", "eval_prompts_per_round", "samples_per_eval"]
    - Combo 0 : [8, 8, 32], "eval_budget": 2048, from the paper
    - Combo 1 : [5, 3, 4], "eval_budget": 60
    - Combo 1 : [5, 3, 8], "eval_budget": 120
    - Combo 2 : [5, 6, 8], "eval_budget": 240
    - Combo 3: [5, 8, 16], "eval_budget": 640
    - Combo 4: [10, 5, 16], "eval_budget": 800

Results

- **Reflection on Eval_budget**

  Due to cost control, I started with GPT 4.1-nano and experimented on eval budget as the paper has shown that most of the algorithms improved as the budget increases.

```
config['eval_budget'] = config['samples_per_eval'] * config['eval_rounds'] * config['eval_prompts_per_round']
```

  Given eval_budget is decided by 3 hyperparameters, to get a quick reflection on how eval_budget can influence test performance (F1) for this task, I chose ‘bf’ evaluator instead of ‘ucb’ evaluator so that 'eval_budget' alone will be the only parameter for prompt selection. All of the reported test F1 scores are from majority voting on 5 trials, based on maxpooling over the final beam of candidates.

  As shown in Table 1, the eval_budget varies from 150 to 800 and the test F1 varies from 0.65-0.7, but there is no significant improvement in test performance (F1) as eval_budget increases. One observation from the experiment is that the F1 score of sampled train data can achieve over 90% performance at around 3 optimization steps, which aligns with the paper that the process can overfit on the train data.

Table 1 Exp 01 - evaluator - “bf”, model - GPT 4.1-nano

Eval_budget	150	240	360	600	800
Prompts for Eval	30	30	30	40	40
Test F1 Score	0.665	0.69	0.655	0.665	0.67

Given the best combo of expansion hyperparameters, proved by later experiments, using GPT 4o-mini alone and using UCB Bandits selection with an eval_budget as small as 60 could suffice the process for prompt selection, shown in Table 2 below.

Table 2 Exp 02 - evaluator - “ucb”, model - GPT 4o-mini

(Eval_budget)	5 x 3 x 4 (60)	5 x 3 x 8 (120)	5 x 6 x 8 (240)	5 x 8 x 16 (640)
Prompts for Eval	30	30	30	60
Test F1 Score	0.74	0.745	0.75/0.735	0.73
Peak Round	5	5	4	3

- **Combinations on Prompt Expansion**

Table 3 and Table 4 show combinations of hyperparameters during the prompt expansion step. Before filtering candidates from the expansion step, the total number of prompts, after getting gradients and synonyms, is decided by 5 hyperparameters. For example, '44112-8' indicates the 5 hyperparameters by '44112' and '8' after the dash line is the filtering hyperparameter for speeding up the process. Both tables have shown that combo '44320-6' is by far the best combo for achieving a 75% test performance with relatively small costs (API calls). Here, UCB Bandits does not show significant advantage as the eval_budget is relatively small. Also, generating more gradients seems more efficient than getting prompt synonyms in explorating high-quality candidates.

Table 3 Exp03 - "evaluator": "bf" , "model": GPT4o-mini, "eval_budget": 240

Expansion Combo	44112-8 (paper)	44310-6	44311-6	44320-6	64320-6	66112-6
Test F1 Score	0.715	0.72	0.73	0.76/0.74	0.75	0.725
Peak Round	4	3	4	6	4	4
Total API calls	4+1+4+5 (14)	4+1+12 (17)	4+1+12+13 (30)	4+1+12 (17)	6+1+18 (25)	6+1+6+7 (20)
Count of New Prompts	14	12	25	24	36	20

Table 4 Exp04 - "evaluator": "ucb" , "model": GPT4o-mini, "eval_budget": 240

Expansion Combo	44312-6	44610-6	48520-6	44320-6	64320-8	66112-6
Test F1 Score	0.73	0.745	0.725	0.75/0.735	0.725	0.735
Peak Round	3	5	5	4	4	4
Total API calls	4+1+12+13 (30)	4+1+24 (29)	4+1+20 (25)	4+1+12 (17)	6+1+18 (25)	6+1+6+7 (20)
Count of New Prompts	38	24	40	24	36	20

- **Learning Curve (GPT 4o-min VS GPT 4.1-nano)**

Hybrid model : utilize GPT-4o-mini for the reasoning part, including generating gradients and getting prompt synonyms, and utilize GPT 4.1-nano for scoring and evaluating prompts, which are F1 based.

Fig1 and Fig2 both show that GPT-4o-mini alone can improve test performance by around 7% as compared to GPT-4.1-nano alone. The performance of a Hybrid model could be better or worse than GPT-4.1-nano alone, which varies by the combinations of hyperparameters in the expansion step.

Fig 1 Test F1 on 4o-mini, 4.1-nano and Hybrid model - Expansion Combo-66112-6

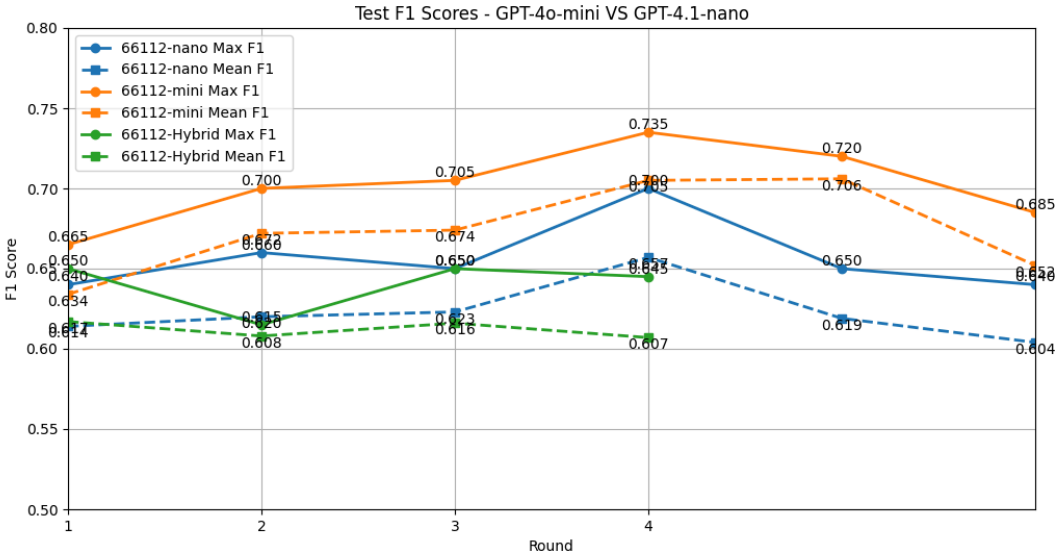
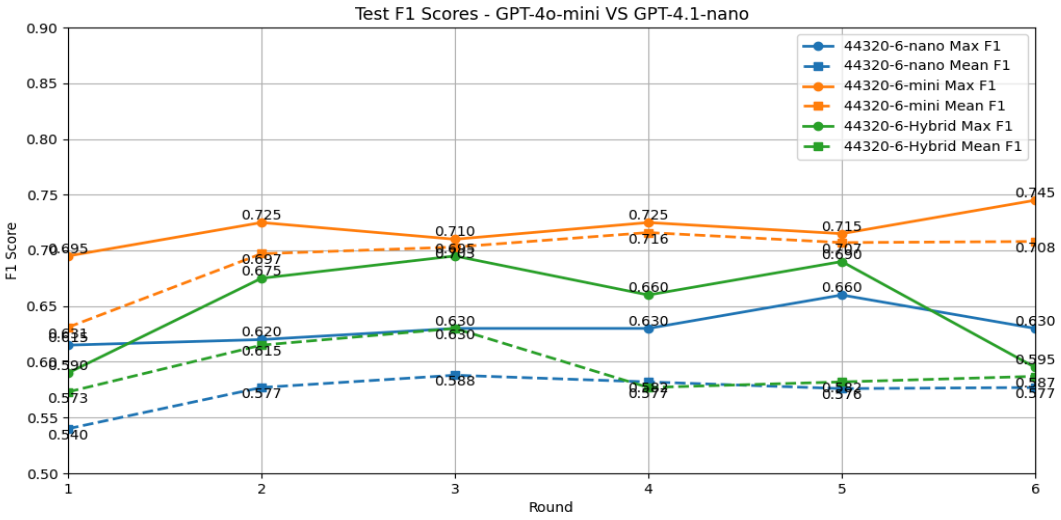


Fig 2 Test F1 on 4o-mini, 4.1-nano and Hybrid model - Expansion Combo-44320-6



- **Exploration Parameter c in UCB Bandits**

Table 5 comparison of c2.0 vs c1.0 - eval_budget-60

UCB-44320-6 (4o-mini)	5 x 3 x 4 - c2.0	5 x 3 x 4 - c1.0
Test F1 Score	0.74	0.725
Peak Round	5	4

- **Observations**
  - Gradient Prompt can be optimized and achieve better performance but has more instability. For example, if the variable *num_feedbacks = 1,* **Prompt 01** will randomly return more than 1 feedback during the run while **Ori Prompt** from paper can return the exact number of feedback whereas the input value of *num_feedbacks.*

Ori Prompt (from paper) :

gradient_prompt = f"""
I'm trying to write a zero-shot classifier prompt.

My current prompt is:
"{prompt}"

But this prompt gets the following examples wrong:
{error_string}

give {num_feedbacks} reasons why the prompt could have gotten these examples wrong.
Wrap each reason with <START> and <END>
"""

Prompt 01:

gradient_prompt = f"""
I'm trying to write a zero-shot classifier prompt.

My current prompt is:
"{prompt}"

But this prompt gets the following examples wrong:
{error_string}

give {num_feedbacks} different reasons why the prompt incorrectly classified these examples.
Wrap each reason with <START> and <END>
"""

Table 6 Comparison of Test F1/ Peak Step for Gradient prompts

Test F1/ Peak Step	Expansion Combo	
Gradient prompt	44112 - 8 (from paper)	44320 - 6
Ori	0.71 / R6	0.685 / R5
01	0.715 / R4	0.75 / R6