Internship weekly report

Mathis Embit

MILES, LAMSADE

April 14, 2024

Outline

Introduction

First week: prompt optimization litterature review

Key points

Intent-based Prompt Calibration: Enhancing prompt optimization with synthetic boundary cases by [LBF24] in february 2024.

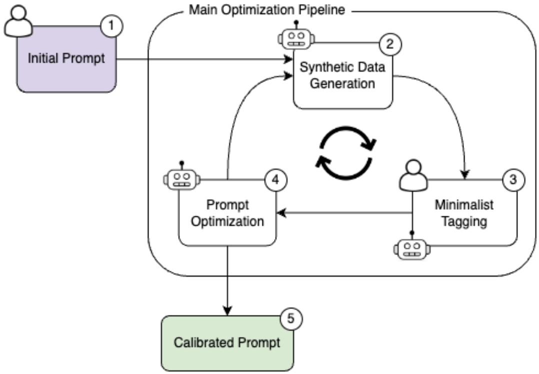
- iterative dialogue process
- requires human annotation (can be replace by an LLM)
- add more and more details to the context through iterations

How it works

Let's consider a classification task.

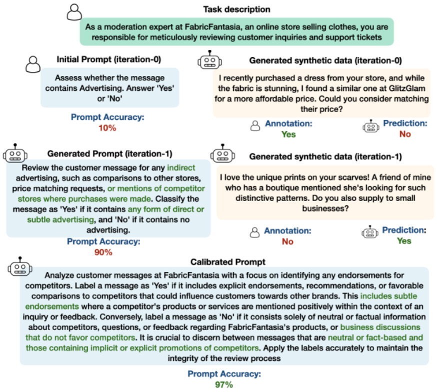
1. inputs: task description, initial prompt
2. generates synthetic data of boundary use cases ("attacks")
3. annotations of those examples (by human or LLM)
4. predictions using the current prompt
5. find the examples that are not correctly predicted
6. add more info to the prompt to cover those examples
7. repeat from 2

How it works



System diagram

How it works



Task description

As a moderation expert at FabricFantasia, an online store selling clothes, you are responsible for meticulously reviewing customer inquiries and support tickets

Initial Prompt (iteration-0)

Assess whether the message contains Advertising. Answer 'Yes' or 'No'

Prompt Accuracy: 10%

Generated synthetic data (iteration-0)

I recently purchased a dress from your store, and while the fabric is stunning, I found a similar one at GlitzGlam for a more affordable price. Could you consider matching their price?

Annotation: Yes

Prediction: No

Generated Prompt (iteration-1)

Review the customer message for any indirect advertising, such as comparisons to other stores, price matching requests, or mentions of competitor stores where purchases were made. Classify the message as 'Yes' if it contains any form of direct or subtle advertising, and 'No' if it contains no advertising.

Prompt Accuracy: 90%

Generated synthetic data (iteration-1)

I love the unique prints on your scarves! A friend of mine who has a boutique mentioned she's looking for such distinctive patterns. Do you also supply to small businesses?

Annotation: No

Prediction: Yes

Calibrated Prompt

Analyze customer messages at FabricFantasia with a focus on identifying any endorsements for competitors. Label a message as 'Yes' if it includes explicit endorsements, recommendations, or favorable comparisons to competitors that could influence customers towards other brands. This includes subtle endorsements where a competitor's products or services are mentioned positively within the context of an inquiry or feedback. Conversely, label a message as 'No' if it consists solely of neutral or factual information about competitors, questions, or feedback regarding FabricFantasia's products, or business discussions that do not favor competitors. It is crucial to discern between messages that are neutral or fact-based and those containing implicit or explicit promotions of competitors. Apply the labels accurately to maintain the integrity of the review process

Prompt Accuracy: 97%

System flow

Conclusion

- needs a calibration process for each new task
- need for human annotation (though it can be replaced by an LLM)

Key points

Large Language Models as Optimizers by [Yan+23] in september 2023.

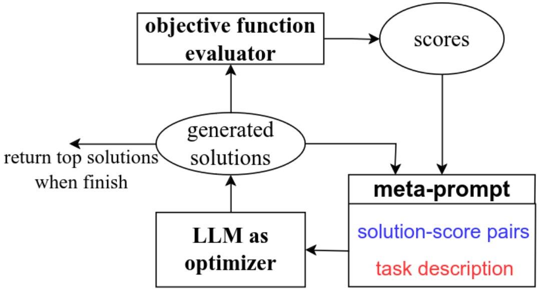
- propose Optimization by PROmpting (OPRO)
- iterative dialogue process
- sort of generalization of intent-based calibration
- has been tested on various problems such as linear regression or traveling salesman

How it works

General algorithm:

1. inputs: meta-prompt
2. LLM generates a solution
3. objective function evaluates the solution and gives a score
4. we add the (solution, score) to the meta-prompt
5. repeat from 2

How it works



OPRO framework

Conclusion

Intent-based calibration is a sort of special case of OPRO where solutions are prompts and the objective function evaluator is human or LLM feedback.

Key points

Automatic Prompt Optimization with “Gradient Descent” and Beam Search by [Pry+23] in may 2023.

- iterative dialogue process
- tries to connect the continuous and the discrete settings of prompt optimization
- differentiation $\longrightarrow$ LLM feedback
- backpropagation $\longrightarrow$ LLM editing
- produces multiple gradients candidates and select one in a best arm identification manner

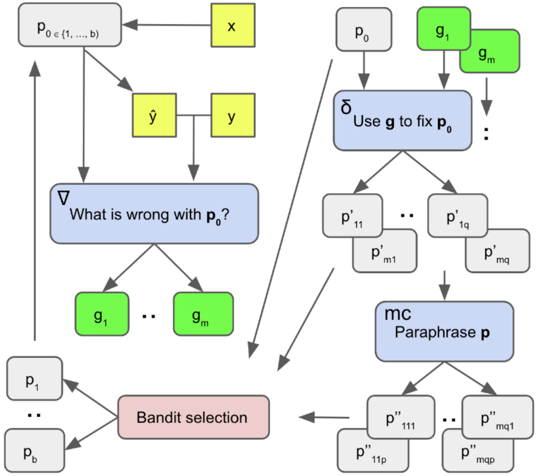
How it works

General algorithm:

1. inputs: initial prompt p_0 and i.i.d. training data consisting of pairs of input/output text $D_{tr} = (x_1, y_1), \ldots, (x_n, y_n)$
2. evaluating a prompt with a batch of data $\longrightarrow \nabla(p_0)$
3. creating a local loss signal which contains information on how to improve the current prompt $\longrightarrow$ textual gradient g
4. editing the prompt in the opposite semantic direction of the gradient before starting the next iteration $\longrightarrow \delta(g, p_0)$
5. select the best prompt
6. repeat from 2

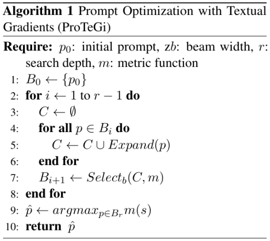
∇ and δ are prompts.

How it works



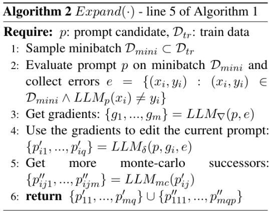
Dialogue tree

How it works



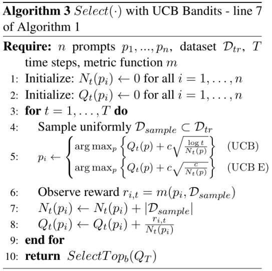
Algorithm 1 Prompt Optimization with Textual Gradients (ProTeGi)

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



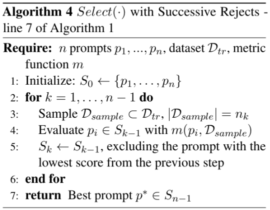
Algorithm 2 $Expand(\cdot)$ - line 5 of Algorithm 1

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



Algorithm 3 $Select(\cdot)$ with UCB Bandits - line 7 of Algorithm 1

```
Require: $n$ prompts $p_1, ..., p_n$, dataset $\mathcal{D}_{tr}$, $T$ time steps, metric function $m$
1: Initialize: $N_t(p_i) \leftarrow 0$ for all $i = 1, \dots, n$
2: Initialize: $Q_t(p_i) \leftarrow 0$ for all $i = 1, \dots, n$
3: for $t = 1, \dots, T$ do
4:     Sample uniformly $\mathcal{D}_{sample} \subset \mathcal{D}_{tr}$
5:     $p_i \leftarrow \begin{cases} \arg\max_p \left\{ Q_t(p) + c\sqrt{\frac{\log t}{N_t(p)}} \right\} & \text{(UCB)} \\ \arg\max_p \left\{ Q_t(p) + c\sqrt{\frac{c}{N_t(p)}} \right\} & \text{(UCB E)} \end{cases}$
6:     Observe reward $r_{i,t} = m(p_i, \mathcal{D}_{sample})$
7:     $N_t(p_i) \leftarrow N_t(p_i) + |\mathcal{D}_{sample}|$
8:     $Q_t(p_i) \leftarrow Q_t(p_i) + \frac{r_{i,t}}{N_t(p_i)}$
9: end for
10: return $SelectTop_b(Q_T)$
```



Algorithm 4 $Select(\cdot)$ with Successive Rejects - line 7 of Algorithm 1

```
Require: $n$ prompts $p_1, ..., p_n$, dataset $\mathcal{D}_{tr}$, metric function $m$
1: Initialize: $S_0 \leftarrow \{p_1, \dots, p_n\}$
2: for $k = 1, \dots, n-1$ do
3:     Sample $\mathcal{D}_{sample} \subset \mathcal{D}_{tr}$, $|\mathcal{D}_{sample}| = n_k$
4:     Evaluate $p_i \in S_{k-1}$ with $m(p_i, \mathcal{D}_{sample})$
5:     $S_k \leftarrow S_{k-1}$, excluding the prompt with the lowest score from the previous step
6: end for
7: return Best prompt $p^* \in S_{n-1}$
```

Conclusion

Still consists in 2 LLMs dialoguing but the updated prompt selection part is more sophisticated.

Key points

AUTOPROMPT: Eliciting Knowledge from Language Models with Automatically Generated Prompts by [Shi+20] in october 2020.

- iterative "gradient steps" process
- computes gradient on token embeddings but uses real tokens in the end

How it works

First, let's define the template λ:

1. input: x_{inp}, x_{trig}
2. Concatenates x_{inp}, the trigger tokens [T] that will be learnable by autoprompt and the [MASK] token that need to be fill by the MLM
$$x_{prompt} = x_{inp}[T][T]\dots[T][MASK]$$
3. output: x_{prompt}

How it works

What is the goal? The goal of AutoPrompt is to find which tokens to put in the [T] in order to maximize $p(y|x_{prompt})$ for a given task (under classification setting where y is a label to predict). [T][T]...[T] depend on the task. They need to be learned for each new task.

The number of [T] and the placement of x_{inp}, [T] and [MASK] are hyperparameters.

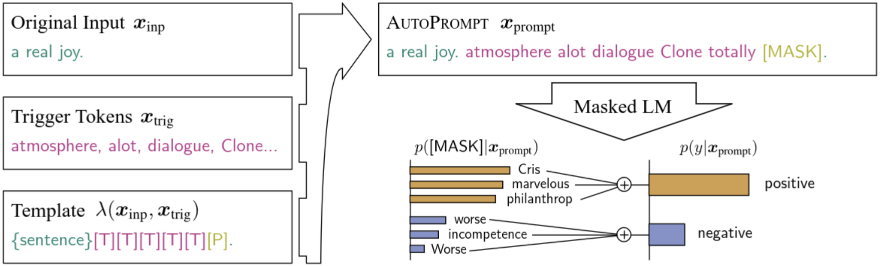
And the goal of the MLM is to fill the [MASK] token.

How it works

The following algorithm manipulates token embeddings.
AutoPrompt training:

1. inputs: (x_{inp}, y) training pairs
2. compute $\log p(y|x_{prompt})$
3. swap $x_{trig}^{(j)}$ and $w \in \mathcal{V}$
4. compute $\log p(y|x_{prompt})$
5. compute $\nabla_{x_{trig}^{(j)}} \log p(y|x_{prompt})$
6. $\mathcal{V}_{cand} = \underset{w \in \mathcal{V}}{top - k} \left(w_{in}^{T} \nabla_{x_{trig}^{(j)}} \log p(y|x_{prompt}) \right)$
7. $x_{trig}^{(j)} = \underset{w_{cand} \in \mathcal{V}_{cand}}{argmax}\ p(y|x_{prompt})$ where in x_{prompt}, $x_{trig}^{(j)} = w_{cand}$
8. repeat from 2

How it works



Original Input $\boldsymbol{x}_{\text{inp}}$
a real joy.
Trigger Tokens $\boldsymbol{x}_{\text{trig}}$
atmosphere, alot, dialogue, Clone...
Template $\lambda(\boldsymbol{x}_{\text{inp}}, \boldsymbol{x}_{\text{trig}})$
{sentence}[T][T][T][T][T][P].
AutoPrompt $\boldsymbol{x}_{\text{prompt}}$
a real joy. atmosphere alot dialogue Clone totally [MASK].
Masked LM
$p([\text{MASK}]|\boldsymbol{x}_{\text{prompt}})$
$p(y|\boldsymbol{x}_{\text{prompt}})$
Cris
marvelous
philanthrop
worse
incompetence
Worse
positive
negative

Dialogue tree

Conclusion

Rather than optimizing using LLMs dialoguing, AutoPrompt computes gradients on the embeddings of some preselected tokens.

However it is still discrete as the embeddings are just the ones of selected tokens, and are here to choose which token helps the most in solving the task.

Conclusion

A lot of papers on discrete optimization through dialogue between LLMs. Some paper such as AutoPormpt explore looking at classic gradients computed on the token embeddings.

References I

[LBF24] Elad Levi, Eli Brosh, and Matan Friedmann. *Intent-based Prompt Calibration: Enhancing prompt optimization with synthetic boundary cases*. 2024. arXiv: 2402.03099 [cs.CL].

[Pry+23] Reid Pryzant et al. *Automatic Prompt Optimization with "Gradient Descent" and Beam Search*. 2023. arXiv: 2305.03495 [cs.CL].

[Shi+20] Taylor Shin et al. *AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts*. 2020. arXiv: 2010.15980 [cs.CL].

[Yan+23] Chengrun Yang et al. *Large Language Models as Optimizers*. 2023. arXiv: 2309.03409 [cs.LG].