Report 2025-4-3

I have implemented `gpt-4o-2024-11-20` in Yamasaki's code. I got the results as follows:

1. Comparison with Original Results

Original Result (in Yamasaki's paper)

表 6.2　Is-in における LLM モデルの実験結果

手法	top1	top3	top5	plus0	plus1	plus3	plus5
dep.0to1(uttr)+adj	0.473	0.730	0.786	0.549	0.704	0.780	0.837
dep.0to2(uttr)+adj	0.473	0.727	0.786	0.549	0.704	0.783	0.839
dep.0to3(uttr)+adj	0.473	0.724	0.786	0.549	0.701	0.780	0.837
GPT-3.5	0.454	0.648	0.673	0.518	0.620	0.676	0.696
GPT-4	**0.727**	**0.808**	**0.839**	**0.769**	**0.820**	**0.839**	**0.845**

Result (is_in) of gpt-4o-2024-11-20

```
top_1_is_in: 0.72394
top_3_is_in: 0.81690
top_5_is_in: 0.83944
plus_0_is_in: 0.76338
plus_1_is_in: 0.81690
plus_3_is_in: 0.83944
plus_5_is_in: 0.84225

running time: 4.348min
```

It seems like there is **very limitted improvement** from `gpt4` to `gpt4o`.

2. Ranking or Not

In Yamasaki's code, the output of LLM is the predicted actions set **(no ranking)**. So when it comes to compute top-1 score, the top-1 score may become lower. Because the most possible (or the most appropriate) action **may not be the first element** in the predicted answer set.

So I try to make the model output the predicted actions sorted by their probabilities in descending order. And set the number of predicted actions to **5** by adding this sentence into **original prompt**.

> また、あなたが正解だと考える行動を5つ出力してください。行動の順番は、正解である確率が高いとあなたが判断する順に並べてください。

Result (top5 probability)

```
top_1_is_in: 0.65634
top_3_is_in: 0.83662
top_5_is_in: 0.89014
plus_0_is_in: 0.73803
plus_1_is_in: 0.82254
plus_3_is_in: 0.87887
plus_5_is_in: 0.89296

running time: 4.872min
```

We can notice that even though the `top-1` is lower than original, **the other metrics obtrained an improvement**. I think this is instereting. Why we got such a low `top-1` when we got improvement in `top-3` , `top-5` , `plus-1`, `plus-3`, `plus-5`.

The reason why we got this result, I think, is that the size of correct answer is not fixed in reference (content) answer we give the LLM. For example:

The reference in original paper

user	【ユーザの状況】 ユーザの発話: どこに片付けたかな？ ユーザの位置: キッチン ユーザが手にしている物: 大根 ユーザの近くにある物: 無し
assistant	[24] おろし器を持ってくる [回答終了]

Maybe we can modify the reference format?

3. CoT or Not

Also, I try to apply **CoT** by adding this sentence to the end of **original prompt**. There is almost no difference compared to original result.

> 日常生活の常識に基づいて、順を追って考え、答えを出してください。

Result

```
top_1_is_in: 0.70986
top_3_is_in: 0.83099
top_5_is_in: 0.85070
plus_0_is_in: 0.76620
plus_1_is_in: 0.81972
plus_3_is_in: 0.85070
plus_5_is_in: 0.85352

running time: 4.873
```

We can see that excluding `top_1_is_in`, there is an improvement in the other scores.

4. Eliminate `該当なし`

Next, I found that maybe we should not let LLM output `該当なし`, because **to do nothing** is always to be seen as wrong answer in Yamasaki's code. However, towards ambiguous utterances, **to do nothing** may sometimes be an appropriate action .

So, I add this sentence to the **original prompt**, delete the command that allows LLM to output `該当なし`, and finally get the result as follows.

```
また、少なくとも1つ以上の行動カテゴリを選択し、同じ行動カテゴリを重複して選択しないようにしてください。
```

Result (CoT)

```
top_1_is_in: 0.72394
top_3_is_in: 0.83662
top_5_is_in: 0.86197
plus_0_is_in: 0.77465
plus_1_is_in: 0.83099
plus_3_is_in: 0.86197
plus_5_is_in: 0.86479

running time: 4.403min
```

We can notice that all `is_in` got improvement excluding `top_1_is_in`. This result is expectable because the option of `to do nothing` does not exist.

The result without using **CoT** is shown below.

Result (Without CoT)

```
top_1_is_in: 0.74366
top_3_is_in: 0.82254
top_5_is_in: 0.84507
plus_0_is_in: 0.77746
plus_1_is_in: 0.81972
plus_3_is_in: 0.84789
plus_5_is_in: 0.85352

running time: 4.281min
```

CoT actually contributed to the results, excluding `top_1_is_in`.