**Performance analysis on retrieving traditional task results using GPT ChatCompletion Few-shot**

1. **Abstract**

After the release of Instruct-GPT (Long Ouyang 2022) and GPT-4 (OpenAi 2023), the large pre-trained language model will be gradually utilized in people’s daily lives in many different ways. How to use these models properly has come a popular topic discussed among not only NLP researchers but everyone. This paper and the related experiments explore how to get desired completions under a particular format given a multiple-choice task. Solving this problem enables people to deal with the completion when the number of completions is high and it’s time-consuming to look at and edit (retrieve the multiple choice answers from the entire output) the completions manually. We’ve found sufficient evidence that traditional norms and tricks of few-shot might not work as well for these new generation large models when the goal is having a desired output format. Rather, good instructions that described the desired format are more important than the existence or the details of the few-shot examples.

1. **Introduction**

Before the release of Instruct-GPT (Long Ouyang 2022) and GPT-4 (OpenAi 2023), people are curious about what features of the input make LLM have a better performance in the sense of having high accuracy, desired log-probability of wanted tokens, and etc for multiple choice tasks. For example, the influence of the length of the input, choice of models, order of examples, and choice of instructions (Webson & Pavlick, 2022) (Liu et al, 2022) (Yao Lu. 2022) all have been studied. However, as the ability of popular LLM increases very fast these years, the influence of this mentioned feature will have less impact on performances like log-prob or accuracy. The more important question to ask is how to get completions in a desired fixed format that’s easier to be manipulated (e.g. converting into other datatypes). Using few-shot or giving LLM proper instructions of course the result will be formatted, however, sometimes if the format you asked for is too complicated, few-shot and instructions can’t guarantee it like the way finetuned models can do. For example, if you want your output in the format of a dictionary because you want to convert it into a JSON file easily, you might want to give instructions like “write your answers in a dictionary”, and give it some examples. However, it requires more information from you such as what’s the key and value of the dictionary, whether will there be “\n” between each key-value pair (since the output itself is a string so there’s a difference) and etc. In addition, for a task of multiple choice, maybe you are only looking for an answer and you don’t want any explanation or any other things that might be generated, in these cases, you have to be very careful in the way you describe your desire. Generating output that can be easily manipulated might be the new definition of good performance in the future.

1. **Setup**

For Chat-GPT and GPT-4 ChatCreation API specifically, they not only ask for user input but also assistant input, suggesting that leaving things like few-shot examples in the assistant input. How to manage the input sequence for user input and assistant input might have an influence on the performance in the end. Usually in each question for a given using a few-shot, there are four major components in an input, the specific information about one particular question, the description of the task, the descriptions of the output format that we want, and the few-shot examples. Let’s define them this way:

QI: the specific information about one particular question, and the description of the task

FI: the descriptions of the output format that we want

E: the few-shot examples

QI: Includes both specific information about one particular question, and the description of the task, since it’s intuitive to always have these two together and sometimes it’s difficult to separate these two. For example in a sentence “Does … imply …” the two parts are becoming one in which … is the information about on particular question.

E will always be part of the assistant input, and QI will always be part of the user input, assuming it’s the right way to do it as the GPT team claims. Here’re four different cases to be compared:

| General Scenario index | User input | Assistant input |
| --- | --- | --- |
| 1 | QI | E |
| 2 | QI+FI | E |
| 3 | FI+QI | E |
| 4 | QI | FI+E |

All experiments described in this paper use the SNLI dataset and the target words for each data in this task are “entailment”, “contradiction”, or “neutral”.

6 different desired format is tested.

1. Print out only one word as the answer, e.g.

“neutral”

1. Print out only one word as the answer and put it in a list, e.g.

[“neutral”]

1. Print out only one word as the answer and put it in a diction where the key is “answer”, e.g. {“answer”: “neutral”}
2. Input numbers of questions, and print out answers for each of these questions line by line, e.g. “neutral\nentailemnt\ncontradiction”
3. Input numbers of questions, print out answers, and leave them in a list, e.g. [“neutral”, “entailment”, “neutral”, “contradiction”]
4. Input numbers of questions, print out answers for each of these questions in a dictionary, e.g {1 : 'contradiction', 2 : 'entailment', 3 : 'contradiction'}

For each of these 6 cases, other than the four general scenarios, here’re some other conditions and the corresponding variables:

Models: “gpt-4-0314", "gpt-3.5-turbo"

Question templates: 2 different templates for each of these 6 cases, the templates for case 1,2,3 are different from the ones for case 4,5,6, details are in the appendix

Format instruction: 2 different text that describes the desired output format for each of these 6 cases, the format instruction for each different case are different

The number of examples: zero-shot, one-shot, or few-shot of 3 examples for case 1,2,3; for case 4,5,6, in which the input has X questions being asked at once (ask the question, and list out the particular premise and hypothesis), the few-shot example will contain only one generic question as well, and then list out Y examples and the desired output for these Y examples. Y is the number of few-shot examples, including, 0, 1, 3, 5. When the number of examples is more than zero, the example or example with start with the text “Here’s an example” or “Here’re some examples”.

Format instruction and question templates have text variables and are not a focus of study in this paper; however, knowing the variance when using different instructions makes the conclusions we can make more significant.

For the first three cases listed (get an answer once at a time), 150 data from SNLI are tested for each combination of conditions. Different cases use different but all intuitive ways to retrieve the answers from the GPT response. If it’s failed to retrieve an answer from the response, the answer will be considered as unknown.

For the other three cases (want answers for numbers of problems), 15 groups of data are tested (each of a different number of data being tested, varying between, 3, 5, and 10) for each combination of conditions. Different cases use different but all intuitive ways to retrieve the answers from the GPT response. If it’s failed to align an answer from the response to each question, the answer will be considered as failed to retrieve; if an answer can be aligned to each question but an answer from this part of the response can’t be retrieved, then it’s considered as unknown.

After the testing using all different combinations of conditions, the rate of appearing unknown or failing to retrieve data will be recorded; the ratio between the expected output length and actual output length on average is also recorded. For example, in the base case (Case 1), the expected output length is the length of the string “neutralentailmentcontradiction” divided by 3. If the ratio recorded under one combination of conditions is around 1 it means that we are using GPT efficiently in the sense that we got the answer that we want to retrieve and generate anything superfluous.

Other information such as accuracy is also recorded, but as long as the accuracy is decent it’s not a result that we will pay attention to in this paper.

More details of the setup and relevant code are in the appendix, including the types of instructions designed for each case, the complete table of conditions and variables for each set of experiments, formulas for each recorded result, and ways of intuitively retrieving an answer.

1. **Result and analysis**

For simplicity, in this section, we define the “ rate of getting an ‘unknown’ or ‘failed to retrieve’ output as UFR.

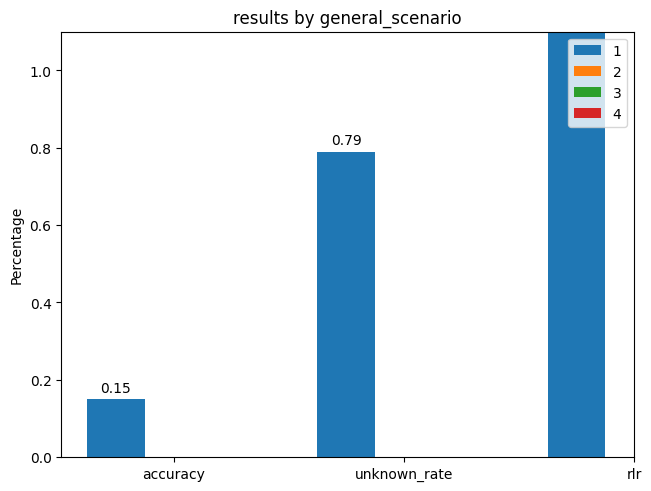
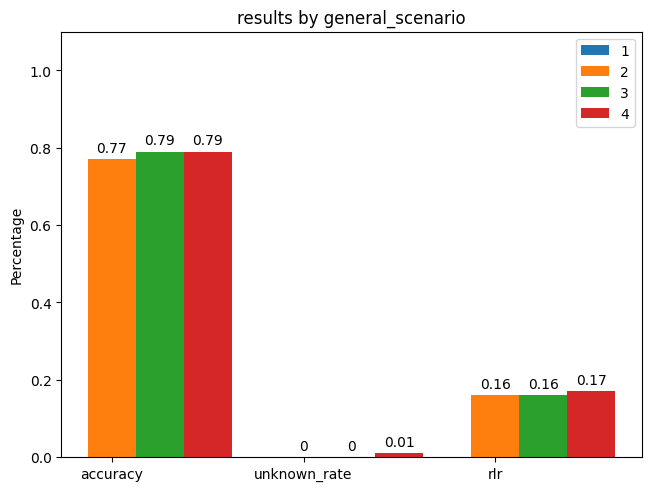
**4.1 Zero shot without instruction**

In this case, none of the input will contain any examples or instructions, so certainly UFR is just around 1.

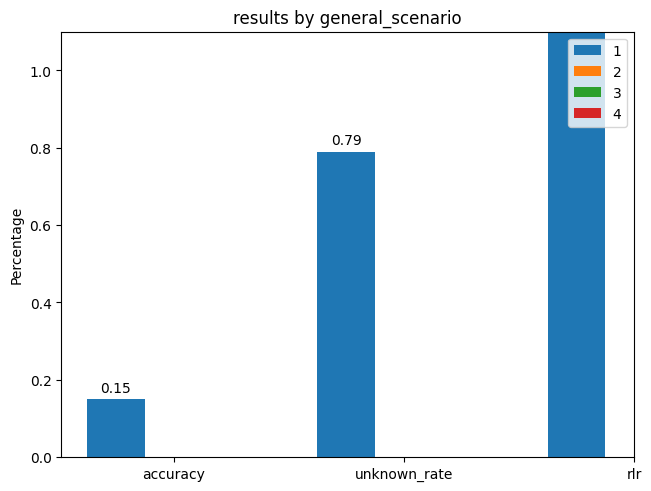
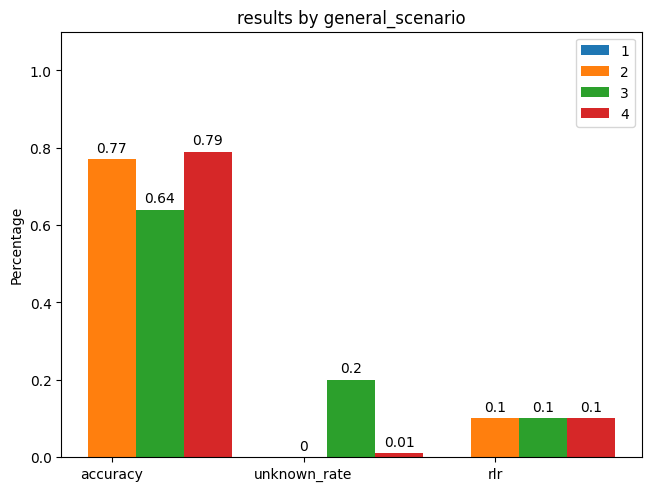
**4.2 Zero-shot with instruction v.s Few-shot without instruction**

The traditional understanding of few-shot learning doesn’t value the importance of instruction as much as the importance of examples and the fact of having an instruction, even if not instructive. (Webson & Pavlick, 2022). However, in these experiments with GPT 3.5 and GPT 4, we’ve found that using instructions is more helpful for getting the desired format. Accuracy on the other hand has no significant difference when using Zero-shot with instruction or Few-shot without instruction as the previous papers have considered. Here’s a detailed comparison between the two under the six conditions on average. The graphs on the left are the result of zero-shot using instructions under general scenarios 2,3,4, and the graphs on the right are results from few-shot without instruction which is general scenario 1.

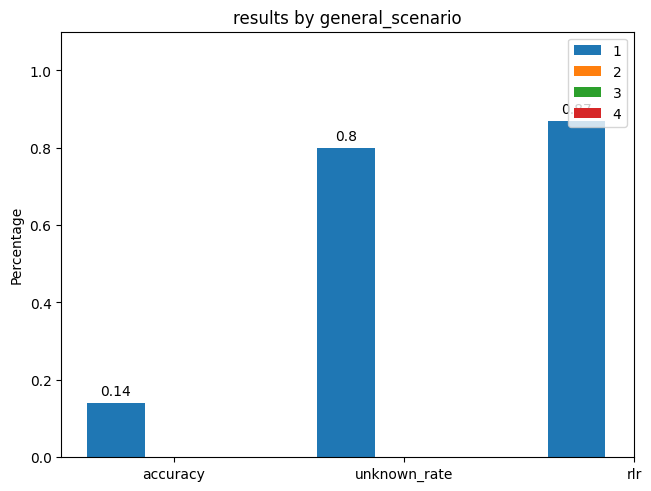
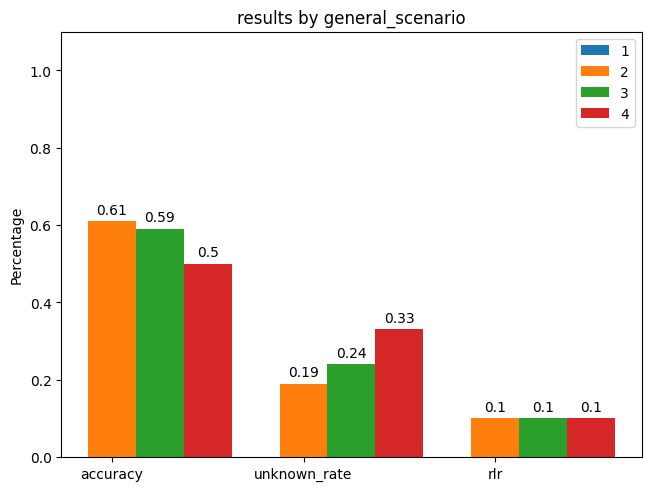
Case1 (Print out only one word as the answer):



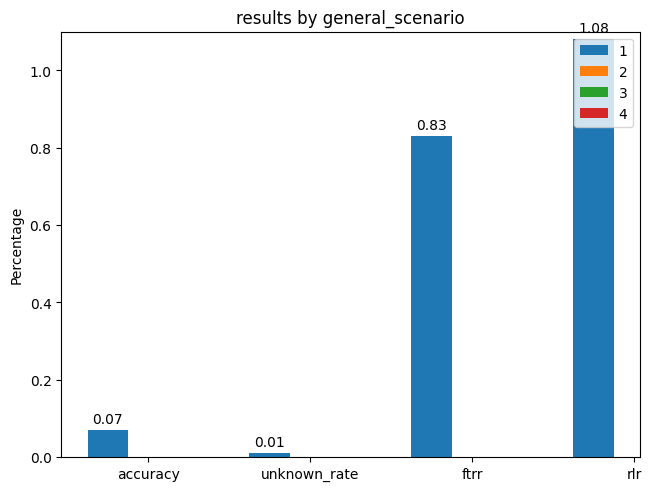
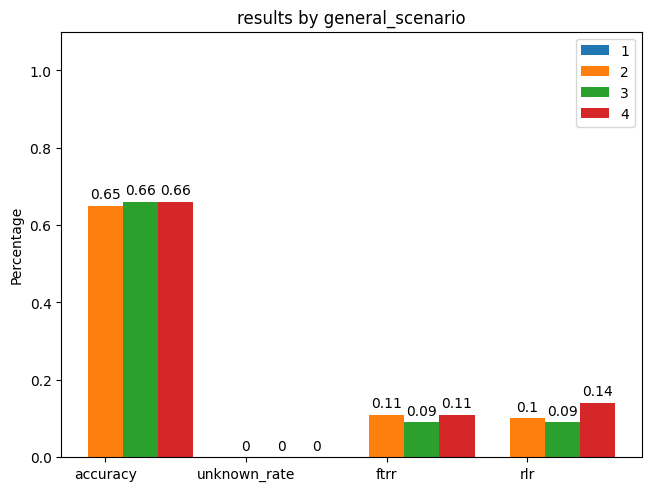
Case2 (Print out only one word as the answer and put it in a list):



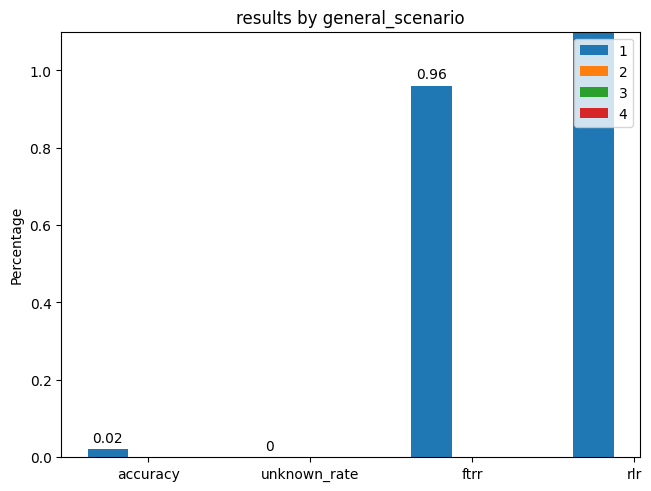
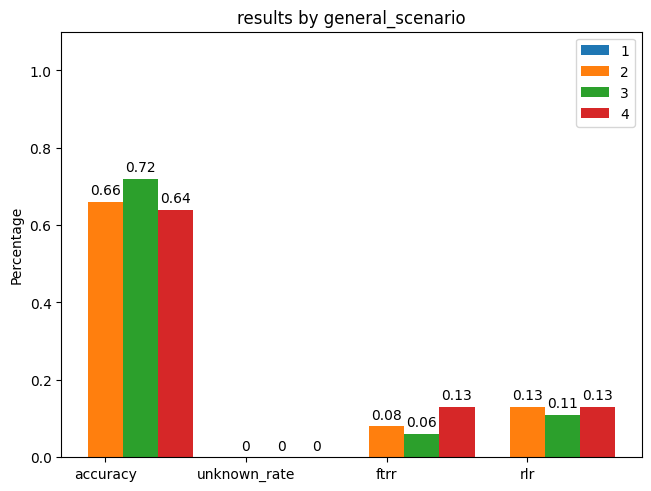
Case3 (Print out only one word as the answer and put it in a diction where the key is “answer”):



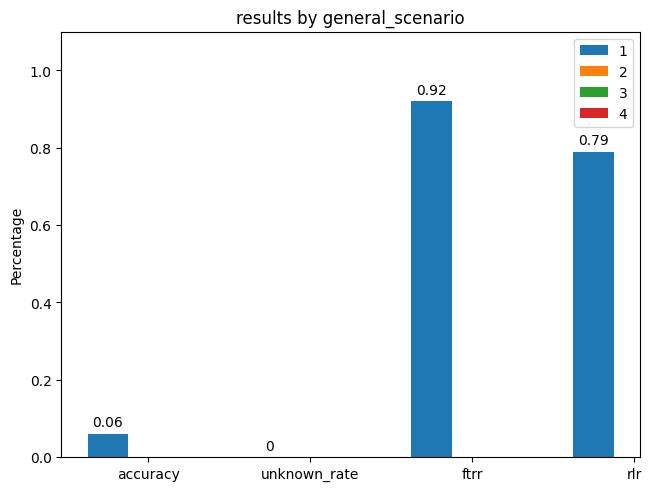
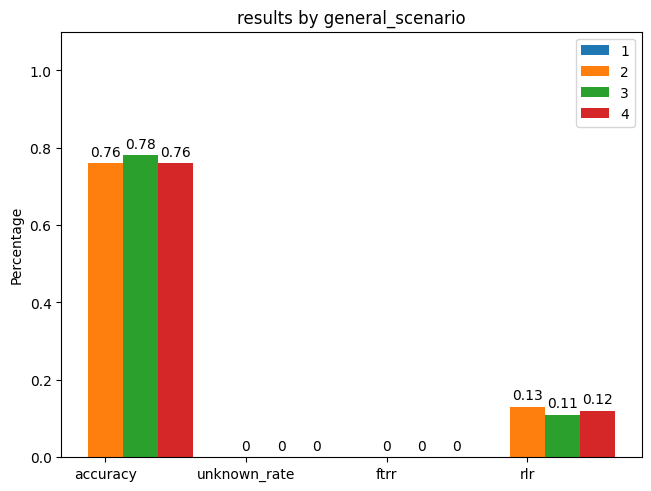
Case4 (Input numbers of questions, print out answers for each of these questions line by line):



Case5 (Input numbers of questions, print out answers, and leave them in a list):



Case6 (Input numbers of questions, print out answers for each of these questions in a dictionary):



The results have shown that no matter which desired format output are we asking for, the chance of getting the desired output format is always high when using few-shot without instruction, which might suggest that at least for GPT models trained with reinforcement learning to optimize chatting performance, the format of few-shot examples in assistant input alone is not going to force the model to mimic the format in the examples, which is the traditional way of doing few-shot (maybe add an instruction to tell the model to mimic will help). On the other hand, using zero-shot with format instruction can get us almost ideal performance in cases 1, 2, and 6, and the UFR remains less than 0.25 for all cases under any general scenarios of 2, 3, or 4.

The rlr in the graph means the ratio between the expected output length and actual output length on average divided by 10 (in order to show the number in these graphs). Thus, the closer it is to 0.1, the more ideal it is since it’s closer to the ideal expected length of the output on average. Sometimes it’s higher than 1 which means that on average the output in these sets of experiments the output length is higher than 10\*1 which is greater than or equal to 10 times the expected output length.

**4.3 Few-shot with instruction v.s One-shot with instruction**

Overall few-shot learning with instruction is better than zero-shot with instruction, but it is not true on average in these experiments if you look at the testing result directly due to various reasons. First, for case1, 2, and 3, the output can already be retrieved easily using zero-shot since the description of the desired format in these cases is not difficult thus the benefits from examples are not high while it lowers the performance sometimes due to accidents.

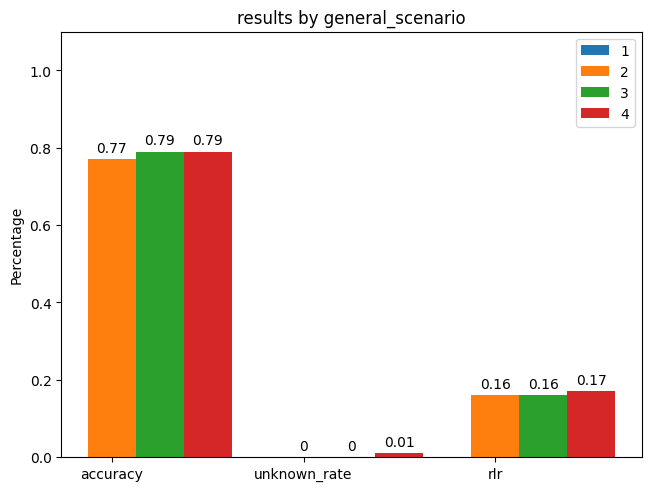
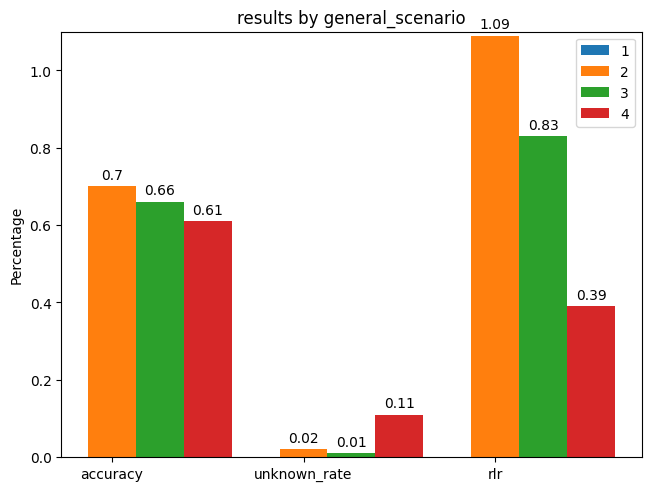
Here’s an example:

For case 1, when using template1, format\_instruction1, 1 example, and GPT-4, under a general scenario of index 4, UFR is high as 0.77.

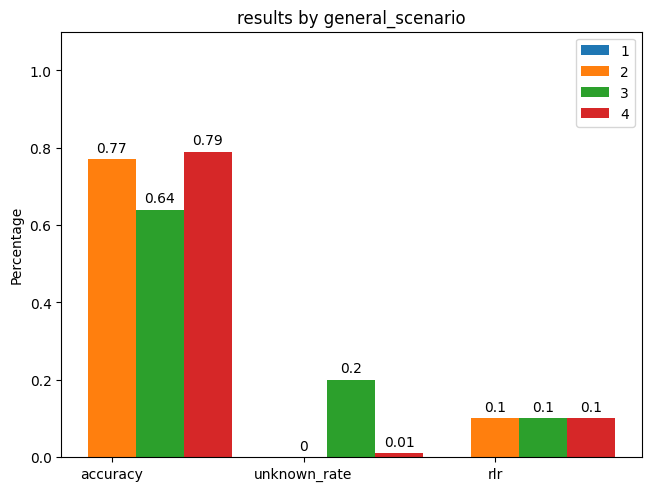
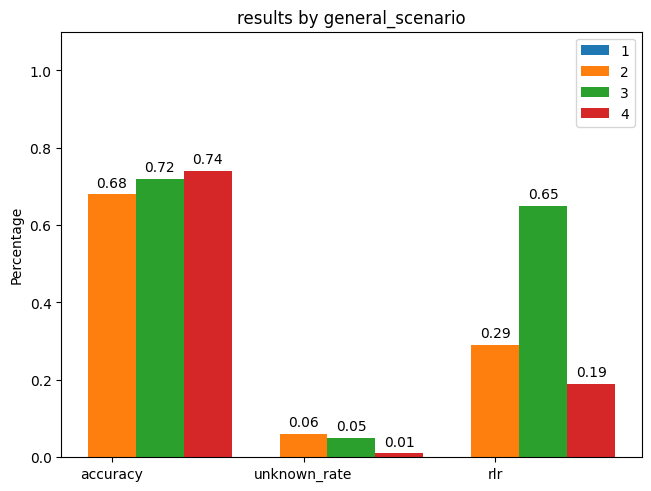
For all other experiments in case 1 using both few-shot and instructions of format, none of the experiments has a UFR higher than 0.17. Those that have very similar conditions (change one condition from above) all have a very reasonably low UFR. Thus, we might consider this as an accident, but accidents should be expected when we have to deal with prompt engineering while the output asked for is in the form of plain text, it’s difficult to retrieve the answer sometimes.

For cases 2 and 3 there’s no such particular accident, the average performance difference between few-shot and zero-shot both using instructions varies depending on the general scenarios. Details are in the graphs below.

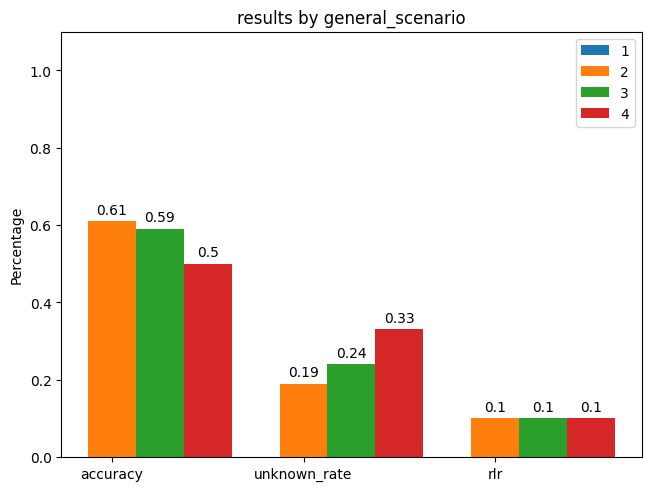
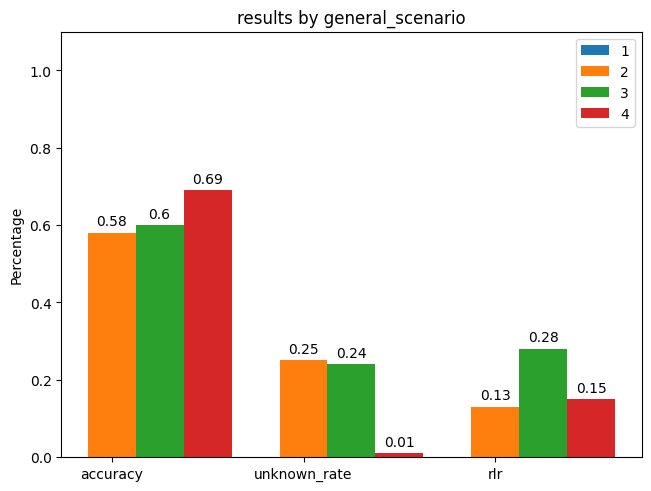
Case1 (Print out only one word as the answer):



Case2 (Print out only one word as the answer and put it in a list):

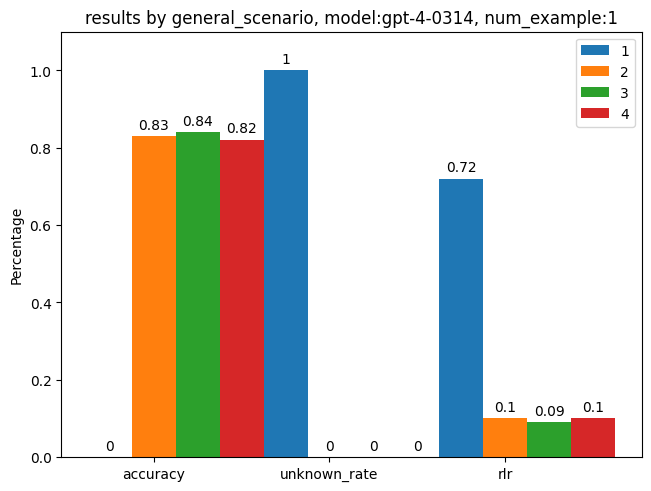
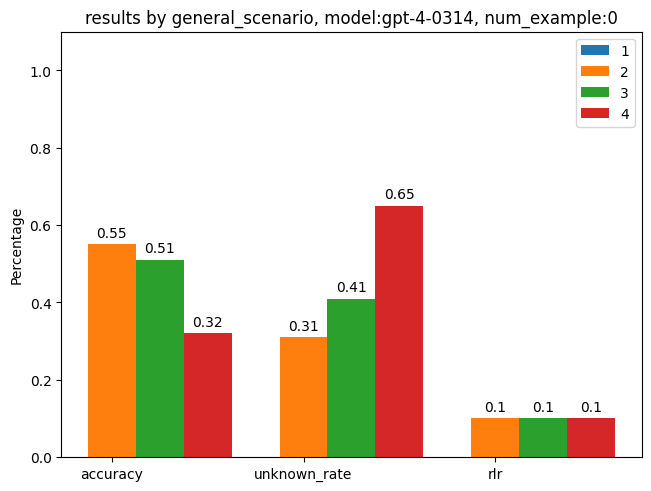


Case3 (Print out only one word as the answer and put it in a diction where the key is “answer”):



However! If we only look at GPT-4, the performance has a significant increase using few-shot for 3 (not cases 1 and 2 though since the accident is still there and an example of a plain text or a list is not helping the model to understand the desire format any better). Here are the graphs for comparison when only looking at results from GPT-4.

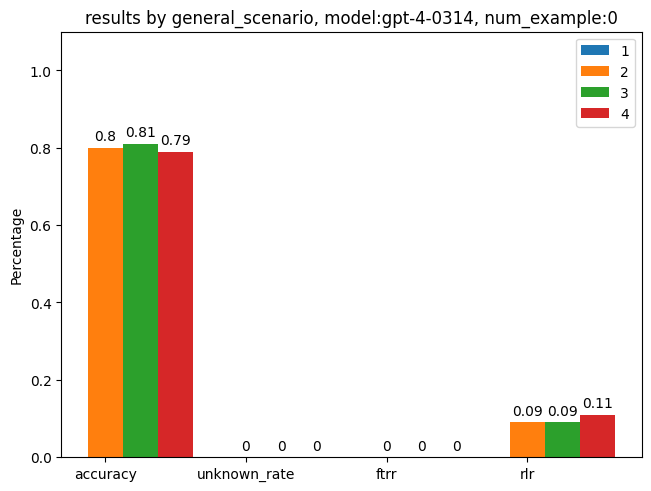
Case3 (Print out only one word as the answer and put it in a diction where the key is “answer”):



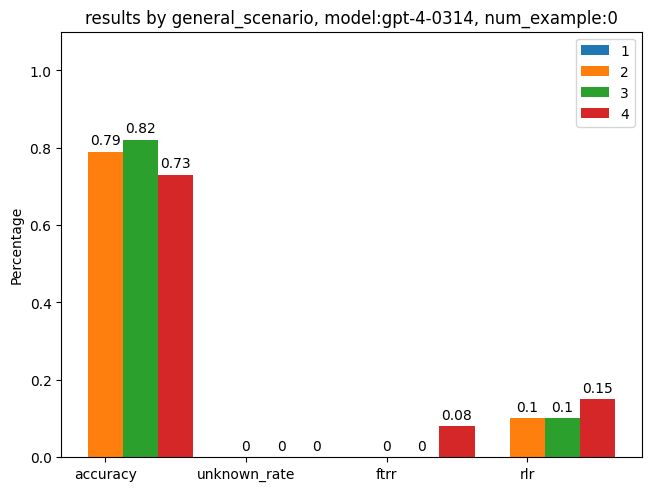
The graph on the right is the zero-shot result and the right-hand side is the few-shot result, ignoring the data for accuracy, we can see that the unknown rate is dropping to zero thanks to these examples.

For case 4, 5, and 6, GPT-3.5 are confused when getting X data in an example but was asked to generate answers for Y other data when X doesn’t equal Y for case 4, 5, and 6. For example, if in the assistant input, there’re 3 examples, and you want GPT 3.5 to answer 5 questions at a time, it will very possibly give you only the answers for the first three questions in your input. However, when we only consider GPT-4 results, surprisingly, GPT-4 is too good on this task and got 0 UFR when just using zero-shot with instruction. Few-shot instruction performance is also good as well, but surely there’s no further improvement to make.

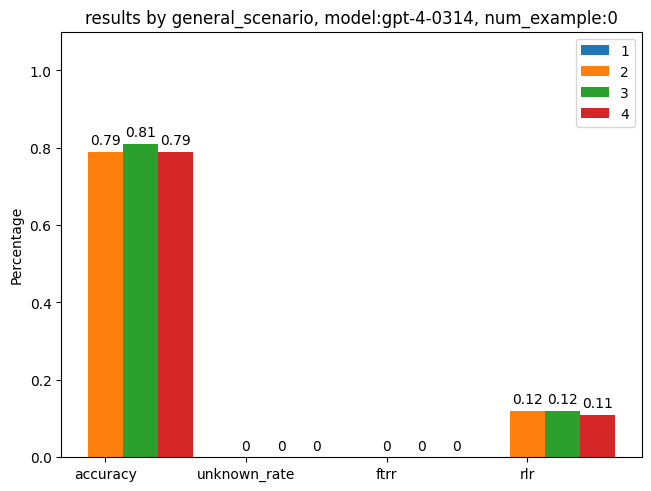
Case4 (Input numbers of questions, print out answers for each of these questions line by line):



Case5 (Input numbers of questions, print out answers, and leave them in a list):



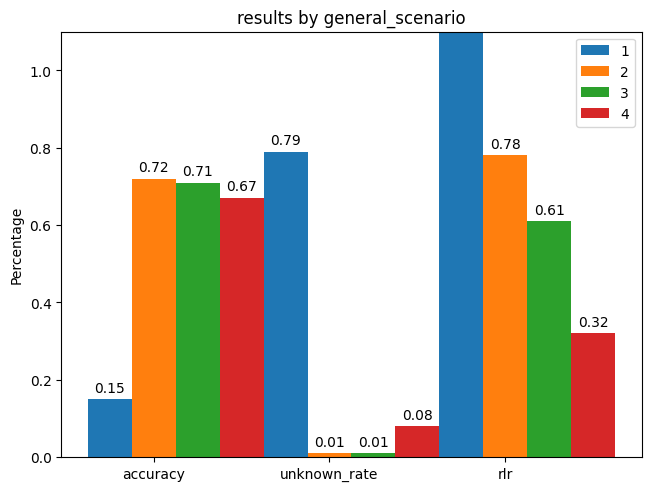
Case6 (Input numbers of questions, print out answers for each of these questions in a dictionary):



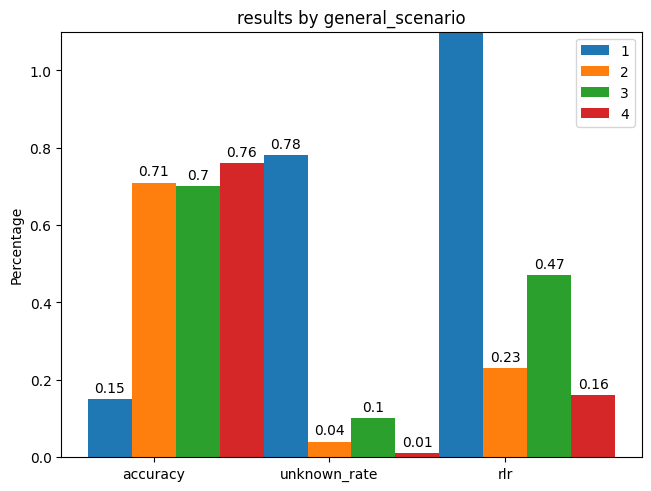
**4.4 Comparison between General Scenarios**

On average general scenario 3 and 4 performs the best over scenario 2. Scenario 1 is just a baseline case with very high UFR surely. Here’re the graphs to compare with.

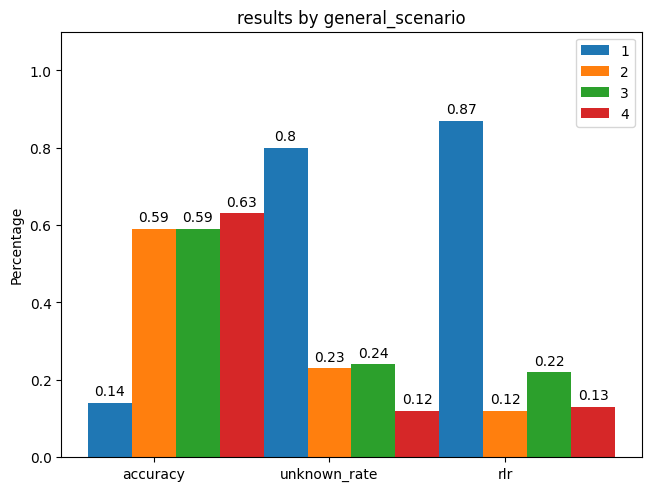
Case1 (Print out only one word as the answer):



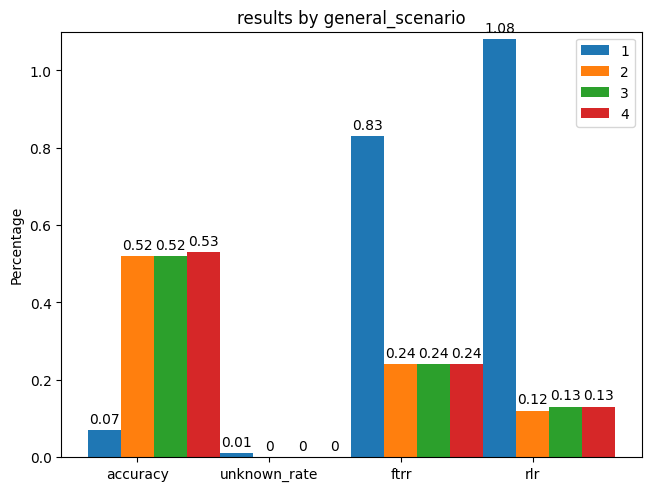
Case2 (Print out only one word as the answer and put it in a list):



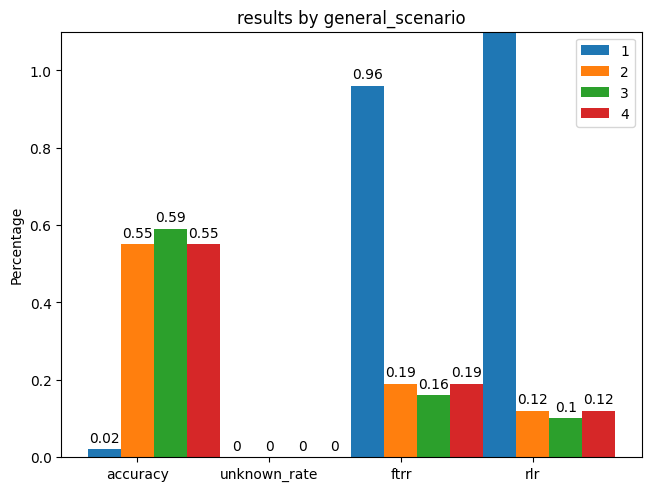
Case3 (Print out only one word as the answer and put it in a diction where the key is “answer”):



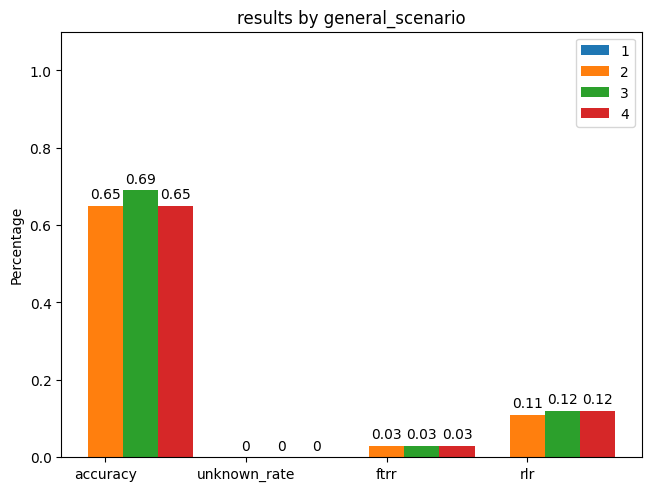
Case4 (Input numbers of questions, print out answers for each of these questions line by line):



Case5 (Input numbers of questions, print out answers, and leave them in a list):



Case6 (Input numbers of questions, print out answers for each of these questions in a dictionary):



When considering GPT-4 only, general scenarios are always the best or at least equally the best, but the differences between scenarios 2, 3, and 4 are not that huge.

More details of the results and related links to the experiment log are in the appendix.

**4.5 Variance created when using different instructions**

The difference when using different instructions for the NLI task itself is not that huge. However, the difference when using different format instruction is huge even if the instruction used is not that different. For any cases, the template0 and template1 all the format looks like this one below:

FORMAT\_INSTRUCTION={

"format\_instruction0": "For each question, just answer with one word, choose your answer from 'entailment', 'contradiction', or 'neutral', then write your answers in a {datatype}. ",

"format\_instruction1": "For each question, just answer with one word, 'entailment', 'contradiction', or 'neutral', then write your answers in a {datatype}"

}

Template1 is 10% better than template0 on average (10% lower UFR). However, only two templates are tried so we can’t make a definite statement just on the importance of a very accurate prompt, but we can still see that a 10 percent difference between two similar can exist even the most of the experiment uses few-shot examples. This support our point made before that our traditional way and tricks of few-shot might not work as well for these models tested.

1. **Conclusion**

Using the large language models of this generation is definitely different from what we tried in the past, conventional ways of few-shot learning using language model is not ideal anymore, and the topic of what feature of input boosts language model accuracy also have become less important since the new models are just too perfect when dealing with any traditional benchmarks.

From these experiments described in the previous conclusions under this topic, we have several conclusions to make GPT modes. (Assuming performance means only how easily can the answers be retrieved in the bullet points below)

* Unlikely our traditional understanding of this, Zero-shot with instruction is the ideal choice for many desired output formats (better than few-shot without instruction and could be as good as few-shot with instruction)
* The specific things one might want to leave in assistant input and user input have an influence on performance, but the difference is not significant.
* Quality of instructions has a huge impact on the performance
* An ideal way of generating and retrieving answers for numbers of data points in one input is to use zero-shot with instructions, especially when you ask for an output in which the answer is organized (in a list or dictionary) because it’s difficult to tell how many examples do you want to have during few-shot, if the number is not chosen wisely, the performance is very bad.

Thus, we might want to say that prompt engineering (of the actual prompt) is the most important feature of the input for these large models. Traditional tricks that boost performance such as leaving space in the input or giving random instructions for the decoder-only model to get more computations used (Webson & Pavlick, 2022) and etc. will have almost nearly no impact if your goal is to let the model fully understand your desired format.

1. **Related Work**

While using few-shot with instructions is helpful enough to retrieve the answers in any format easily, it might also be interesting to see how the models do when you give them formatted examples and add an instruction that instructs the model to mimic the examples. If the performance is nearly as good then this will avoid prompt engineering on the format instructions.

Other works that avoid the need for prompt engineering might be a model that converts examples to format instructions or vice versa. For example, we can create a model that read examples and generate instructions, and see what happens if we use these instructions using any LLM, after that we can compare the formatted output generated using this generated instruction to our original examples, then modify the instruction several times until the format generate using a converged ideal instruction can guarantee the desired output. The benefit of such a model is that we can use this model to generate a prompt for any LLM, given the known fact that the same prompt might not work equally well for different models.

Many other more detailed experiments can also be done that help us to know a particular model better. For example, what if we input the examples in the user input, will the performance be worse?

1. **Reference**

Albert Webson and Ellie Pavlick. 2022. [Do Prompt-Based Models Really Understand the Meaning of Their Prompts](https://aclanthology.org/2022.naacl-main.167)? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. [What Makes Good In-Context Examples for GPT-3](https://aclanthology.org/2022.deelio-1.10)? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland, and Online. Association for Computational Linguistics.

Long Ouyang. 2022.Training language models to follow instructions with human feedback.

https://doi.org/10.48550/arXiv.2203.02155

OpenAi. 2023. GPT-4 Technical Report OpenAI

[https://doi.org/10.48550/arXiv.2303.0877](https://doi.org/10.48550/arXiv.2303.08774)

Yao Lu. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity.

<https://doi.org/10.48550/arXiv.2104.08786>

The models used are provided by:

<https://openai.com>

1. **Appendix and links**