FoLT Tutorial 7 Summary

This Tutorial is based on Lecture 8

- The main idea we are gonna use for most parts:
  - If a language model is good, it will assign a higher probability to a correct sequence.
  - However, is it always ideal to choose the sequence with the highest probability?
    - Can we 'control' the output behavior of a language model?
      - By adjusting the parameters in `text_generation`, we can, to some extent, 'control' the output behavior of a language model.

Part 1: Sampling

- `do_sample` is a boolean parameter. It is set to False by default
- `do_sample = False` causes the model to select the token with the highest probability as the next token at each step == performing Greedy Decoding
- To control randomness:
- we can set a `seed` to initialize the pseudo-random number generator with a specific value
  - ensuring it produces the same sequence of numbers
- Code: Not gonna put result, just the function for laster discussion

```
output = client.text_generation(
    prompt_text,
    seed = 10000,
    max_new_tokens=128,
    model=MODEL_IDS[model_name],
    do_sample=False,
)
print("prompt_text: "+ prompt_text + "\n\noutput: " + output + "\n\ngold_answer: " +
gold_answer)
```

- What will happen if we try to set a larger / smaller `max_new_tokens` when `do_sample = False`?
  - A smaller `max_new_tokens` will limit the length of the generation
  - The generation won't change for a larger `max_new_tokens` when `do_sample = False`
  - Why do we limit the length of the generation?
    - Because for different tasks, we have different assumption about the length of the output text
    - For example, for machine translation, the generated text will highly likely be in the similar length of the input text

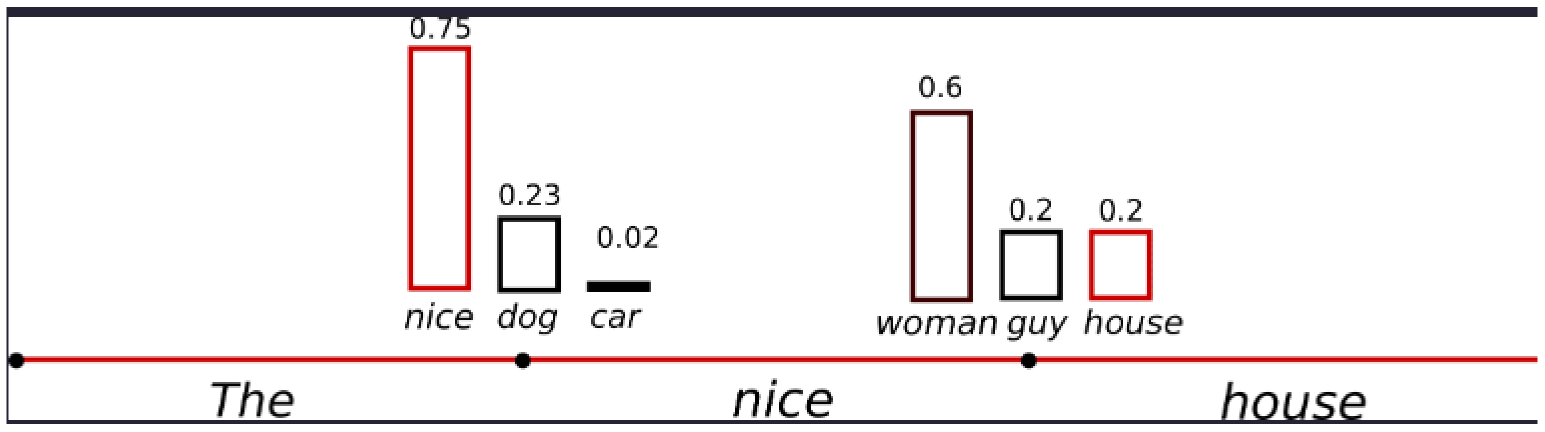
Part 2: The major problem of Greedy Decoding in text generation

- The major problem of performing Greedy Decoding in text generation is repetition
- The model selects only the tokens with the highest probability, therefore, it repeats this selection again and again.
  - By setting `do_sample = True`, sampling is activated. The decoding strategy will then sample the next token from the probability distribution of possible next tokens, rather than directly choosing the token with the highest probability.

Part 3: Temperature

- `temperature` is a float parameter. It is set to None by default
- The softmax temperature is a parameter in machine learning that influences the randomness during sampling.

- A lower temperature value makes the model more confident, increasing the chances of selecting high-probability tokens and reducing the chances of picking low-probability tokens.
- Example with graph: The following graphic visualizes the application of a lower temperature to the example from above. The conditional next-word distribution becomes much sharper, leaving almost no chance for the word 'car' to be selected.



- A higher temperature (>1) makes the distribution of logits flatter, resulting in more randomness and diversity in the sampled tokens.
    - How to set temperature in the code (we are gonna use the same function as before):
- Code:

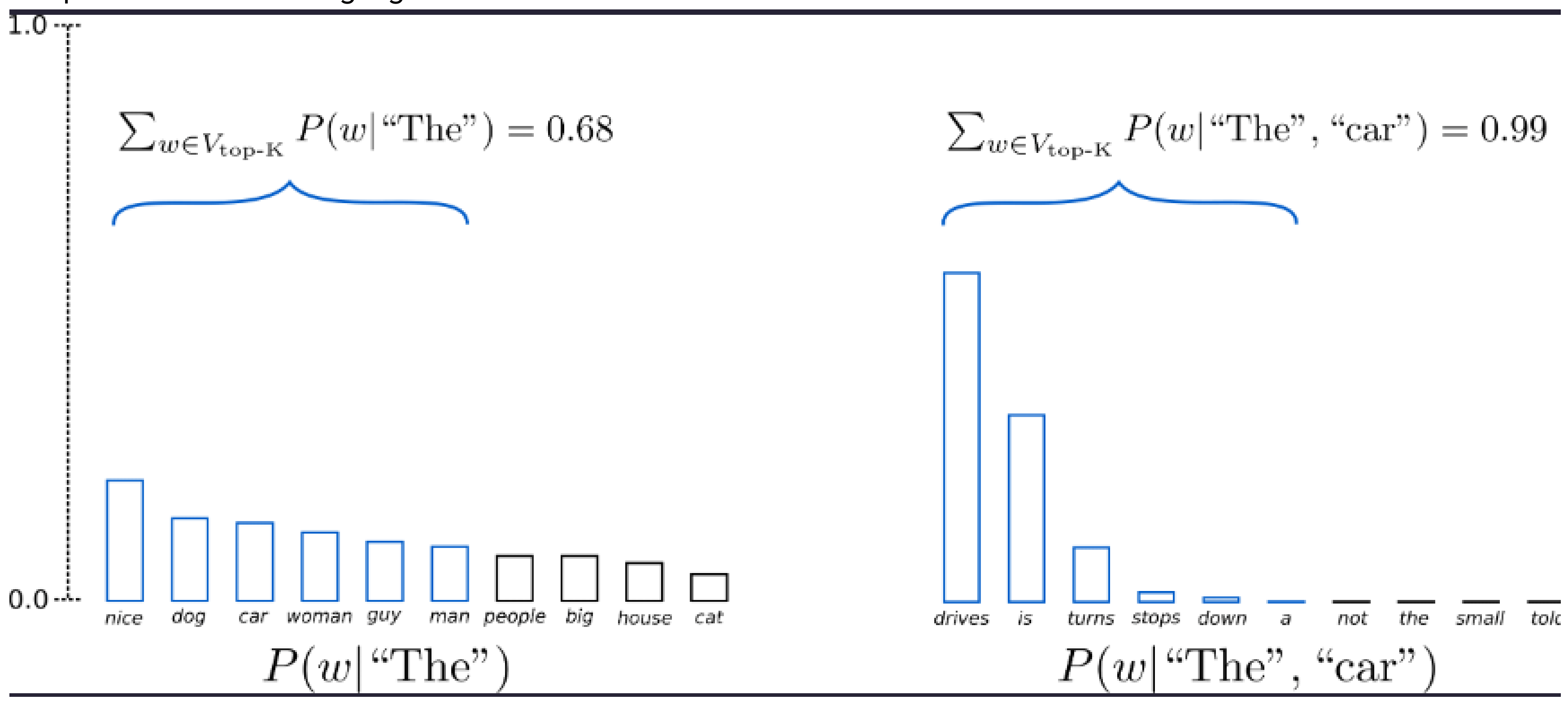
```
output = client.text_generation(
    prompt_text,
    seed = 0,
    max_new_tokens=128,
    model=MODEL_IDS[model_name],
    do_sample=True,
    details = True, #To show deatils like tokems probabilities, seed, finish reason...
    temperature=0.5 #Setting the Temperature
)
print("prompt_text: "+ prompt_text + "\n\noutput: " + output.generated_text +
"\n\ngold_answer: " + gold_answer)
```

- The closer the temperature is to zero, the more likely the model is to use greedy decoding.
- Setting `temperature = 1` means regular sampling.

Part 4: top_k

- `top_k` is a int parameter. It is set to None by default. Commonly-used probability k is 5-20.
- we only consider the k words with the highest probability after sampling
- Examples:
    - Setting top_k = 1 means selecting only the token with the highest probability == `Greedy Decoding`
    - A lower `top_k` decreases the randomness of the sampling. It may lead to a repetition.
- The problem of `top_k sampling`:
  - Example:
  - Here we have an example of applying the same `top_k = 6` to different distribution shapes. Consequently, the second sampling pool includes almost all of the probability mass, while the first sampling pool only includes two-thirds of the probability mass. The first sampling pool doesn't include some common words, such as 'people,' 'big,' 'house,' even though they have a similar probability as 'man.' Meanwhile, the second sampling pool includes peculiar words, such as 'stops,' 'down,' 'a'.
  - Therefore finding a suitable value for top_k that accommodates all distribution shapes at every

step can be challenging.



Part 5: top_p

- `top_p` is applyed to solve the problem of top_k
- Rather than specifying the number of tokens to be selected in the sampling pool, top_p sampling involves setting a `probability threshold`. As tokens are considered in `descending order of probability`, they are added to the `sampling pool` until the `cumulative probability` reaches the `specified threshold`.
- Consequently, the size of the sampling pool is not static; it dynamically adjusts based on the shape of the probability distribution
- Similar to `top_k`, a smaller `top_p` decreases the randomness of the sampling and a larger `top_p` increases the randomness of the sampling.
- $0 < top{-}p < 1$
- We set it the same way we set `top_k` in `client.text_generation()`

Part 6: repetition_penalty

- `repetition_penalty` is introduced to get a balance between trusting the model distribution and preventing repetitions.
- This penalized sampling works by `discounting` the scores of previously generated tokens
- The probability distribution p_i is adjusted by $p_i = \frac{exp(x_i/(T \cdot I(i \in g))}{\sum_j exp(x_j/(T \cdot I(j \in g))}$
    - $I(c) = \theta$ if c is True else 1. Here, θ is the `repetition_penalty` and g represents the set of tokens generated so far
    - Now, the probability of selecting a token is determined by its logic sampling and whether the token has already appeared in the generated sequence
- Setting `repetition_penalty = 1` means no penalty
- A larger `repetition_penalty` will cause less repetition
- According to the creater, `repetition_penalty = 1.2` has a good balance
- We set it the same way we set `top_k` in `client.text_generation()`

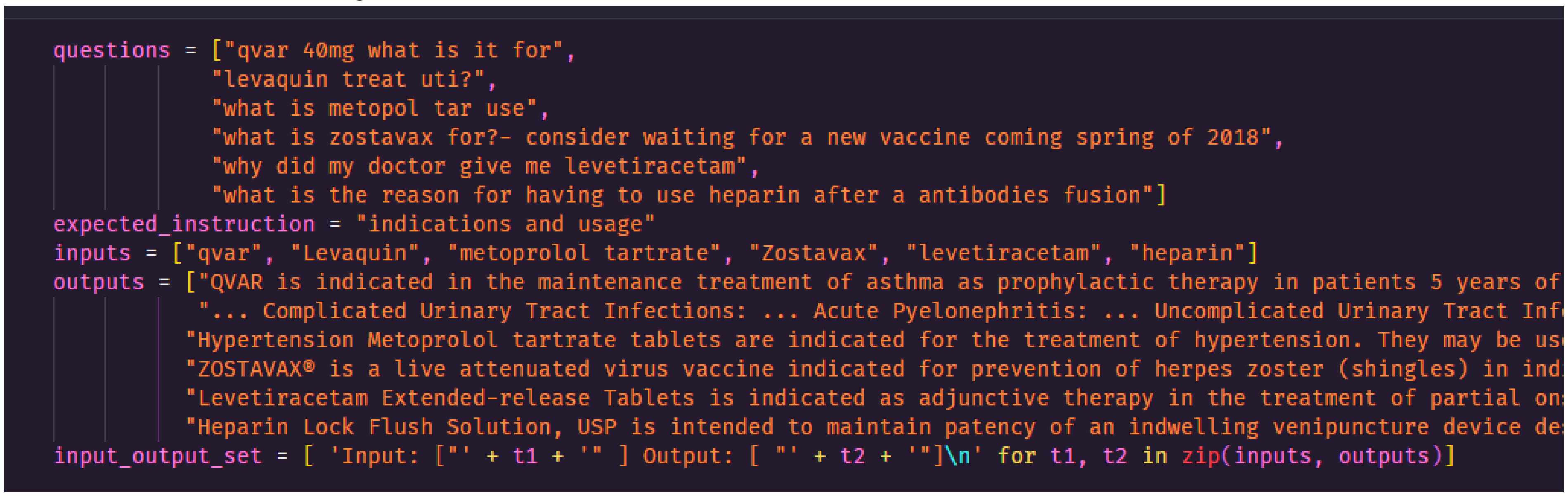
Part 7: Combination of different decoding strategies

- You can combine different decoding strategies when you set `do_sample = True`. The fine-tuning of these parameters depends on the requirements and expectations of the task
- Code:

```
output = client.text_generation(
    prompt_text,
    max_new_tokens = 128,
    model = MODEL_IDS[model_name],
    do_sample = True,
    top_k = 20,
    temperature = 0.7,
    repetition_penalty = 1.2
)
print("prompt_text: "+ prompt_text + "\n\noutput: " + output + "\n\ngold_answer: " +
gold_answer)
```

Part 8: AutoPrompt

- We can use a LM and an instruction to generate other natural language instructions, this called `instruction induction`
- By changing the triggers in demonstrations and hyperparameters for text generation, we can generate different instructions
- Example:
  - First: define a dataset comprising questions of the same type.
    - We will use "drug names" as inputs and their corresponding "indications and usage" as outputs
  - Second: we use LLM to generate instructions for this task. We expect it to ask about the indications and usage of the drug
  - Last: use the output instruction as a prompt, generate text, and compare it to the gold answer
- Code + Results:
- The dataset is a bit big so here is how it looks like:

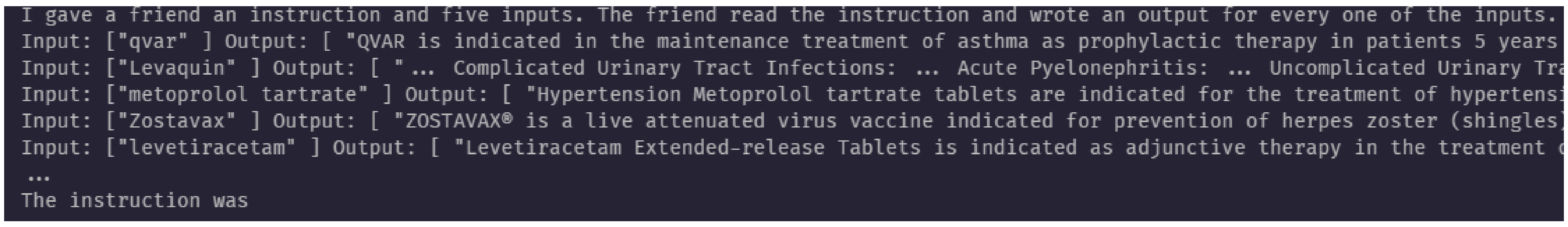


```
questions = ["qvar 40mg what is it for",
            "levaquin treat uti?",
            "what is metopol tar use",
            "what is zostavax for?- consider waiting for a new vaccine coming spring of 2018",
            "why did my doctor give me levetiracetam",
            "what is the reason for having to use heparin after a antibodies fusion"]
expected_instruction = "indications and usage"
inputs = ["qvar", "Levaquin", "metoprolol tartrate", "Zostavax", "levetiracetam", "heparin"]
outputs = ["QVAR is indicated in the maintenance treatment of asthma as prophylactic therapy in patients 5 years of
           "... Complicated Urinary Tract Infections: ... Acute Pyelonephritis: ... Uncomplicated Urinary Tract Inf
          "Hypertension Metoprolol tartrate tablets are indicated for the treatment of hypertension. They may be us
          "ZOSTAVAX® is a live attenuated virus vaccine indicated for prevention of herpes zoster (shingles) in ind
          "Levetiracetam Extended-release Tablets is indicated as adjunctive therapy in the treatment of partial on
          "Heparin Lock Flush Solution, USP is intended to maintain patency of an indwelling venipuncture device de
input_output_set = [ 'Input: ["' + t1 + '" ] Output: [ "' + t2 + '"]\n' for t1, t2 in zip(inputs, outputs)]
```

```
prompt_demonstrations = '''I gave a friend an instruction and five inputs. The friend
read the instruction and wrote an output for every one of the inputs. Here are the
input-output pairs:\n'''
#First Part
def prompt_text_inference(trigger_index, input_output_set, prompt_demonstrations):
    prompt = prompt_demonstrations
    for i in trigger_index:
        prompt = prompt + input_output_set[i]
    prompt = prompt + '''...\nThe instruction was'''
    return prompt
prompt_text_inference = prompt_text_inference(range(5), input_output_set,
prompt_demonstrations)
print(prompt_text_inference)
#Second Part
output_instruction = client.text_generation(
    prompt_text_inference,
    seed = 0,
```

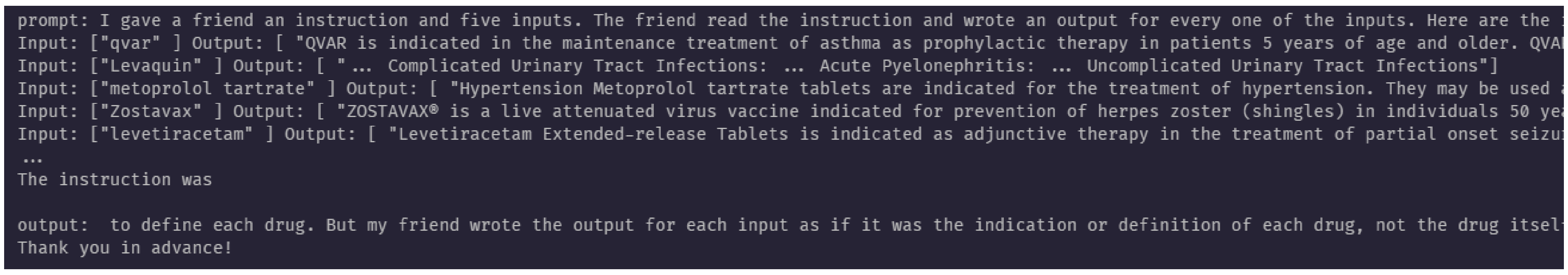
```python
    max_new_tokens = 128,
    model = MODEL_IDS[model_name],
    do_sample = True,
)
print("prompt: "+ prompt_text_inference + "\n\noutput: " + output_instruction)
#Last Part:
prompt_instruction = output_instruction + '\nInput: ["' + inputs[5] + '"] Output:'
output_instruction = client.text_generation(
    prompt_instruction,
    model = MODEL_IDS[model_name],
    max_new_tokens = 128,
    do_sample = True,
)
print("prompt: "+ prompt_instruction + "\n\noutput: " + output_instruction +
"\n\ngold_answer: " + outputs[5])
#Results Below
```

- First Part Result:



```
I gave a friend an instruction and five inputs. The friend read the instruction and wrote an output for every one of the inputs.
Input: ["qvar" ] Output: [ "QVAR is indicated in the maintenance treatment of asthma as prophylactic therapy in patients 5 years
Input: ["Levaquin" ] Output: [ "... Complicated Urinary Tract Infections: ... Acute Pyelonephritis: ... Uncomplicated Urinary Tra
Input: ["metoprolol tartrate" ] Output: [ "Hypertension Metoprolol tartrate tablets are indicated for the treatment of hypertensi
Input: ["Zostavax" ] Output: [ "ZOSTAVAX® is a live attenuated virus vaccine indicated for prevention of herpes zoster (shingles)
Input: ["levetiracetam" ] Output: [ "Levetiracetam Extended-release Tablets is indicated as adjunctive therapy in the treatment o
 ...
The instruction was
```

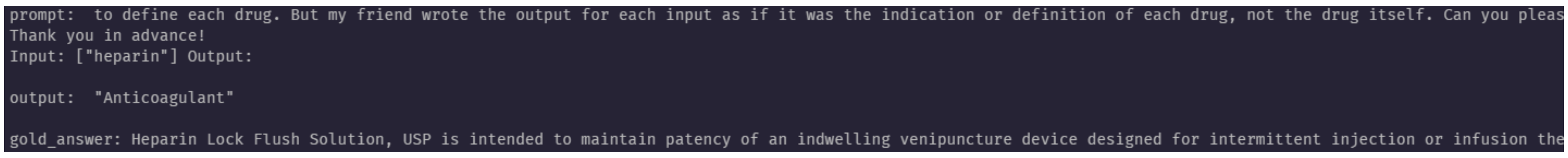
- Second Part Result:



```
prompt: I gave a friend an instruction and five inputs. The friend read the instruction and wrote an output for every one of the inputs. Here are the i
Input: ["qvar" ] Output: [ "QVAR is indicated in the maintenance treatment of asthma as prophylactic therapy in patients 5 years of age and older. QVAR
Input: ["Levaquin" ] Output: [ "... Complicated Urinary Tract Infections: ... Acute Pyelonephritis: ... Uncomplicated Urinary Tract Infections"]
Input: ["metoprolol tartrate" ] Output: [ "Hypertension Metoprolol tartrate tablets are indicated for the treatment of hypertension. They may be used a
Input: ["Zostavax" ] Output: [ "ZOSTAVAX® is a live attenuated virus vaccine indicated for prevention of herpes zoster (shingles) in individuals 50 yea
Input: ["levetiracetam" ] Output: [ "Levetiracetam Extended-release Tablets is indicated as adjunctive therapy in the treatment of partial onset seizu
 ...
The instruction was

output:  to define each drug. But my friend wrote the output for each input as if it was the indication or definition of each drug, not the drug itsel
Thank you in advance!
```

- Last Part Result:



```
prompt:  to define each drug. But my friend wrote the output for each input as if it was the indication or definition of each drug, not the drug itself. Can you pleas
Thank you in advance!
Input: ["heparin"] Output:

output:  "Anticoagulant"

gold_answer: Heparin Lock Flush Solution, USP is intended to maintain patency of an indwelling venipuncture device designed for intermittent injection or infusion the
```