Kaggle Seasons #11

QUEEN MARY MACHINE
LEARNING SOCIETY

Drawing With LLMs

- Prize Money: 50.000$ distributed across the Top-5
- More importantly: Awards points & medals (recognition)
- Company: Kaggle
- Reason: Introduction of a new competitions mode

Source: https://www.kaggle.com/competitions/drawing-with-llms

kaggle

Competition Exploit

#	Team	Members		Score	Entries	Last	Join
1	To Infinity and Beyond			1.131	22	9h	
2	new team name			0.901	32	13h	
3	butsureminder			0.757	9	1d	
4	Nikita Babych			0.757	19	3d	
5	kei hasegawa			0.704	12	5h	

S_SHOHEI · 75TH IN THIS COMPETITION · POSTED 13 HOURS AGO

Sharing my masterpiece: LB 0.635

Since the metric will be updated, I will share my solution.

It's so beautiful, isn't it?

LB 0.635

aesthetic_score = 0.8385

BILZARD · 425TH IN THIS COMPETITION · POSTED 3 DAYS AGO

"Text not present" Attack [VQA score~0.88]

I found VQA model is really honest 😂


```
p(fidelity) = 0.900
p(text) = 0.019
p(aesthetic) = 0.284
```

First Submission

1. Go to https://www.kaggle.com/competitions/drawing-with-llms/overview
2. Click on *“Official Starter Notebook”*
3. Click on *“Copy & Edit”*
    i. https://www.kaggle.com/code/dster/drawing-with-llms-starter-notebook/notebook
4. Click on “Save Version”
5. Enter a “Version name”
6. Click “Save”
7. Go back to https://www.kaggle.com/competitions/drawing-with-llms/overview
8. Click “Submit Prediction”, select “Notebook Version” and confirm “Submit”

Approach 1 - Model Switching

- Try out different models to see which models excel in what areas
- Model sizes (look for balance!):
  - More parameters - higher quality models
  - Less parameters - faster model
- Example model families: Gemma 2, Qwen2.5, Llama 3.1, Deepseek R1, Gemma 3 (launched yesterday)
- Example model sizes: Gemma 2 2B, Gemma 2 9B, Gemma 2 27B
- Models can come in variants. E.g. Gemma 2 9B, Gemma 9B-it
- Open LLM Leaderboard: https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard

kaggle

Approach 2 - Prompt Engineering

- Few-shot learning and zero-shot learning
  - Mixture of different types of examples (e.g. complex/simple SVGs)
- Try out different phrasings
- Experiment with chain-of-thought (giving the model opportunity to plan out what it will do next)
- Provide more details about what's required

```
<example>
<description>"a visual representation of a blue square enclosed by a red circle"</description>
<plan>
Draw a red circle as the background shape. Draw a blue square and position it to be centered within the circle, ensuring it is smaller than the circle so it appears enclosed.
</plan>
```svg
<svg viewBox="0 0 256 256" width="256" height="256">
  <circle cx="50" cy="50" r="40" fill="red"/>
  <rect x="30" y="30" width="40" height="40" fill="blue"/>
</svg>
```
</example>
```

kaggle

Further Improvements - Optimising for speed

- The competition requires submissions to meet the following criteria:
  - Generate all 500 SVGs in the test data in under 9 hours - Average 64.8s/SVG
  - No SVG can take longer than 5 minutes to generate

Approaches:

- Use a smaller model. E.g. Gemma 2 27B -> Gemma 2 9B
- Implement a stop token sequence
- Reduce max new tokens

kaggle

Further Improvements - Stop token sequence

- The model will stop generating new tokens after it has generated the stop sequence.
- In our case, we will use “</svg>” as a stop sequence, as this tag marks the end of the SVG file
- This ensures we don't waste compute generating unnecessary tokens after the SVG is complete.

```python
class SVGTagStoppingCriteria(StoppingCriteria):
    def __init__(self, tokenizer, max_length=None):
        super().__init__()
        self.tokenizer = tokenizer
        self.svg_tag_token_ids = self.tokenizer.encode("</svg>", add_special_tokens=False)
        self.max_length = max_length
        self.current_length = 0

    def __call__(self, input_ids: torch.LongTensor, scores: torch.FloatTensor, **kwargs) -> bool:
        self.current_length += 1
        if self.max_length is not None and self.current_length > self.max_length:
            logging.info(f"Stopping due to max_length: {self.max_length}")
            return True

        generated_tokens = input_ids[0].tolist()
        decoded_text = self.tokenizer.decode(generated_tokens)

        if re.search(r'</svg>\s*$', decoded_text, re.IGNORECASE):
            logging.info("Stopping due to </svg> tag.")
            return True
        return False
```

```python
stopping_criteria_list = StoppingCriteriaList([
    SVGTagStoppingCriteria(self.tokenizer, max_length=max_new_tokens)
])

with torch.no_grad():
    output = self.model.generate(
        **inputs,
        max_new_tokens=max_new_tokens,
        do_sample=False,
        stopping_criteria=stopping_criteria_list,
    )
```

Further Improvements - Changing max new tokens

- Max new tokens limits how many tokens the model generates (if a stop sequence is not reached)
- Increasing this parameter will allow your model to generate more complex SVGs (with correct prompting), but will take longer
- Decreasing it prevents the model from spending too much time on an overly complex SVG - but the SVG code could be truncated - causing syntax errors - which can be rectified using an algorithm

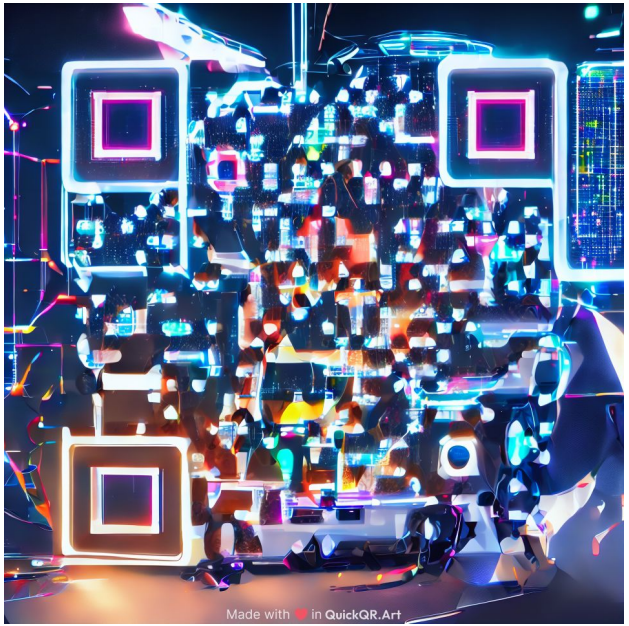
```python
# You could try increasing `max_new_tokens`
def predict(self, description: str, max_new_tokens=450) -> str:
    def generate_svg():
        try:
            prompt = self.prompt_template.format(description)
            inputs = self.tokenizer(text=prompt, return_tensors="pt").to(DEVICE)

            stopping_criteria_list = StoppingCriteriaList([
                SVGTagStoppingCriteria(self.tokenizer, max_length=max_new_tokens)
            ])

            with torch.no_grad():
                output = self.model.generate(
                    **inputs,
                    max_new_tokens=max_new_tokens,
                    do_sample=False,
                    stopping_criteria=stopping_criteria_list,
                )
```

New Communications Platform

- Join our Discord:
  - Link: https://discord.gg/xcfp4UHGWa
  - QR-Code:



kaggle

Thanks for Listening!

QUEEN MARY MACHINE
LEARNING SOCIETY