Presenting...
Prompt Engineering in Emacs

Shane Mulligan

<2021-03-01 Mon>

Following along

Repositories for following along

github1s.com/mullikine/presentation-prompt-engineering-in-emacs
github1s.com/semiosis/examplary
github1s.com/semiosis/pen.el
github1s.com/semiosis/prompts
github1s.com/semiosis/prompt-engineering-patterns
github1s.com/minimaxir/gpt-3-client

Demo

```
ssh -oBatchMode=no shane@124.197.60.232 -p 9922
```

Text Generator

Background knowledge

- `GPT-3` is a `seq2seq` model (a text generator)
    - It's stochastic but can be configured to be deterministic.

Key concepts

- prompt,
- completion, and
- tokens

Limitations

Combined, the text prompt and generated completion must be below 2048 tokens (roughly ~1500 words).

context-stuffing With only 2048 tokens, you need to make use of your real estate by providing instructions and making implicit information explicit.

Prompt Engineering

Characteristics

- declarative, like `html`
- stochastic, like `problog`
- Unlocks new types of applications
- Speeds up development

Prompts as functions

pen-generate-prompt-functions

Generate prompt functions for the files in the prompts directory
Function names are prefixed with pen-pf- for easy searching.
http://github.com/semiosis/prompts

exemplary: examples as functions

An example-oriented DSL that can be used to construct and compose NLP tasks.
Why is a DSL needed for this? Just to make the code a little more terse.

Regex

https://github.com/pemistahl/grex

```
(def regex
  "example 1\nexample2" "^example [12]$"
  "example 2\nexample3" "^example [23]$"
  "pi4\npi5" "^pi[45]$")
```

Analogy

```
(def analogy
  ;; Each line is a training example.
  "NNs" "NNs are like genetic algorithms in
  that both are systems that learn from
```

Prompt YAML format Part 1

meeting-bullets-to-summary.prompt

```
title: "meeting bullet points to summary"
prompt: |+
    Convert my short hand into a first-hand
    account of the meeting:

    <1>

    Summary:
engine: "davinci-instruct-beta"
temperature: 0.7
max-tokens: 60
```

Prompt YAML format Part 2

meeting-bullets-to-summary.prompt

```
top-p: 1
frequency-penalty: 0.0
presence-penalty: 0.0
best-of: 1
stop-sequences:
- "\n\n"
conversation-mode: no
stitch-max: 0
```

stitch-max Keep stitching together until reaching this limit. This allows a full response for answers which may need n*max-tokens to reach the stop-sequence.

Prompt YAML format: Part 3

meeting-bullets-to-summary.prompt

```
vars:
- "notes"
examples:
- |+
    Tom: Profits up 50%
    Jane: New servers are online
    Kjel: Need more time to fix software
    Jane: Happy to help
    Parkman: Beta testing almost done
```

Some prompts I've made

generate-vim-command.prompt

```
Vim

Insert "Q: " at the start of the line
:%s/^/Q: /g.
###
Remove whitespace from the start of each line
:%s/^\s*/\1/g
###
Join each line with the next line
:1,$j
###
Make all occurrences of Steve lowercase
:%s/Steve/steve/g
###
<1>
```

Create a prompt

Ask the audience

- What type of text to generate
    - Could be code, prose, etc.

Tutorials

Ruby

`https://www.twilio.com/blog/generating-cooking-recipes-openai-gpt3-ruby`