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

```
1  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

Some prompts I've made

generate-vim-command.prompt

```
 1  Vim
 2
 3  Insert "Q: " at the start of the line
 4  :%s/^/Q: /g.
 5  ###
 6  Remove whitespace from the start of each line
 7  :%s/^\s*/\1/g
 8  ###
 9  Join each line with the next line
10  :1,$j
11  ###
12  Make all occurrences of Steve lowercase
13  :%s/Steve/steve/g
14  ###
15  <1>
```

Tasks suitable for GPT-3

Classification

- Tweet Sentiment
- Company categorization
- Labeling parts of speech

- `http://github.com/semiosis/prompts/blob/master/prompts/tweet-sentiment-classifier.prompt`
- `http://github.com/semiosis/prompts/blob/master/prompts/keyword-extraction.prompt`

Generation

- Tweet Sentiment
- Company categorization
- Labeling parts of speech

generate-vim-command.prompt

```
 1  Vim
 2
 3  Insert "Q: " at the start of the line
 4  :%s/^/Q: /g.
 5  ###
 6  Remove whitespace from the start of each line
 7  :%s/^\s*/\1/g
 8  ###
 9  Join each line with the next line
10  :1,$j
11  ###
12  Make all occurrences of Steve lowercase
13  :%s/Steve/steve/g
14  ###
15  <1>
```

Prompt YAML format Part 1

meeting-bullets-to-summary.prompt

```
 1  title: "meeting bullet points to summary"
 2  prompt: |+
 3      Convert my short hand into a first-hand
 4      account of the meeting:
 5
 6      <1>
 7
 8      Summary:
 9  engine: "davinci-instruct-beta"
10  temperature: 0.7
11  max-tokens: 60
```

Prompt YAML format Part 2

meeting-bullets-to-summary.prompt

```
1  top-p: 1
2  frequency-penalty: 0.0
3  presence-penalty: 0.0
4  best-of: 1
5  stop-sequences:
6  - "\n\n"
7  conversation-mode: no
8  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
```

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`

examplary: 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

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

examplary: examples as functions

Analogy

```
 1  (def analogy
 2    ;; Each line is a training example.
 3    "NNs" "NNs are like genetic algorithms in
 4    that both are systems that learn from
 5    experience"
 6    "Social media" "Social media is like a
 7    market in that both are systems that
 8    coordinate the actions of many
 9    individuals.")
10
11  (def field
12    "chemistry" "study of chemicals"
13    "biology" "study of living things")
```

Something funny

```
› source › git › mullikine › prompt-engineer-mode › prompts › obfuscate-language.prompt
prompt: |+
  The human asks the AI to make a simple description more complicated. The AI responds by
writing a verbose response full of jargon to explain a simple concept.

  Human: explain how to use a drinking fountain.

  AI: To activate the mechanized dihydrogen monoxide dispensary service station, use the flanges
to provide a downward force to the simple circular activation console and continue to apply
force until the pneumatic system begins operating as intended, which can be surmised at a time
when the dihydrogen monoxide travels in a parabolic arc which at is greatest exceeds three
inches from the rounded excess liquid receptacle. Once hydration is completed, cease applying
force to the console and regard whether or not the dispensary has ceased expelling contents as
expected.

  Human: explain how to play with a dog

  AI: Being a typical domesticated animal, having been genetically selected for controllability,
trainability, and general anthropomorphic features and functions, the typical dog possesses
highly specialized and developed feeding behavior. The typical feeding behavior involves routine
sustenance intake which can be observed by monitoring changes in animal size. Routine sustenance
intake involves active and passive feeding habits, and one can gauge animal health by observing
the relative levels of hydration. During training, reward conditioned reinforcement is applied.
This involves the practice of rewarding targeted behaviors through the application of
reinforcers, with the provision that the targeted behavior is observed. Relevant to the
discussion at hand is the typical active and passive behavior exhibited by a typical dog.

  Human: explain how to <1>

  AI:
engine: "davinci"
temperature: 0.8
max-tokens: 100
top-p: 1
frequency-penalty: 0.2
presence-penalty: 0.4
best-of: 1
stop-sequences:
  - "\n"
inject-start-text: yes
inject-restart-text: yes
show-probabilities: off
varnames:
```

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