Presenting. . .
Imaginary Programming with Emacs

Shane Mulligan

<2021-03-01 Mon>

Following along

Repositories for following along

`http://github1s.com/semiosis/pen.el`
`http://github1s.com/semiosis/prompts`
`https://mullikine.github.io/posts/imaginary-programming-with-gpt-3/`
imaginary programming glossary
imaginary computing glossary
semiosis protocol glossary
Pen.el glossary
https://arxiv.org/abs/2107.13586 Pre-train, Prompt, and Predict

slides

`http://github1s.com/mullikine/imaginary-programming-transcript-emacsconf-2021`

glossary

`http://github.com/semiosis/glossaries-gh/blob/master/imaginary-programming.txt`

Demo

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

```
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>
```

Tasks suitable for GPT-3

Classification

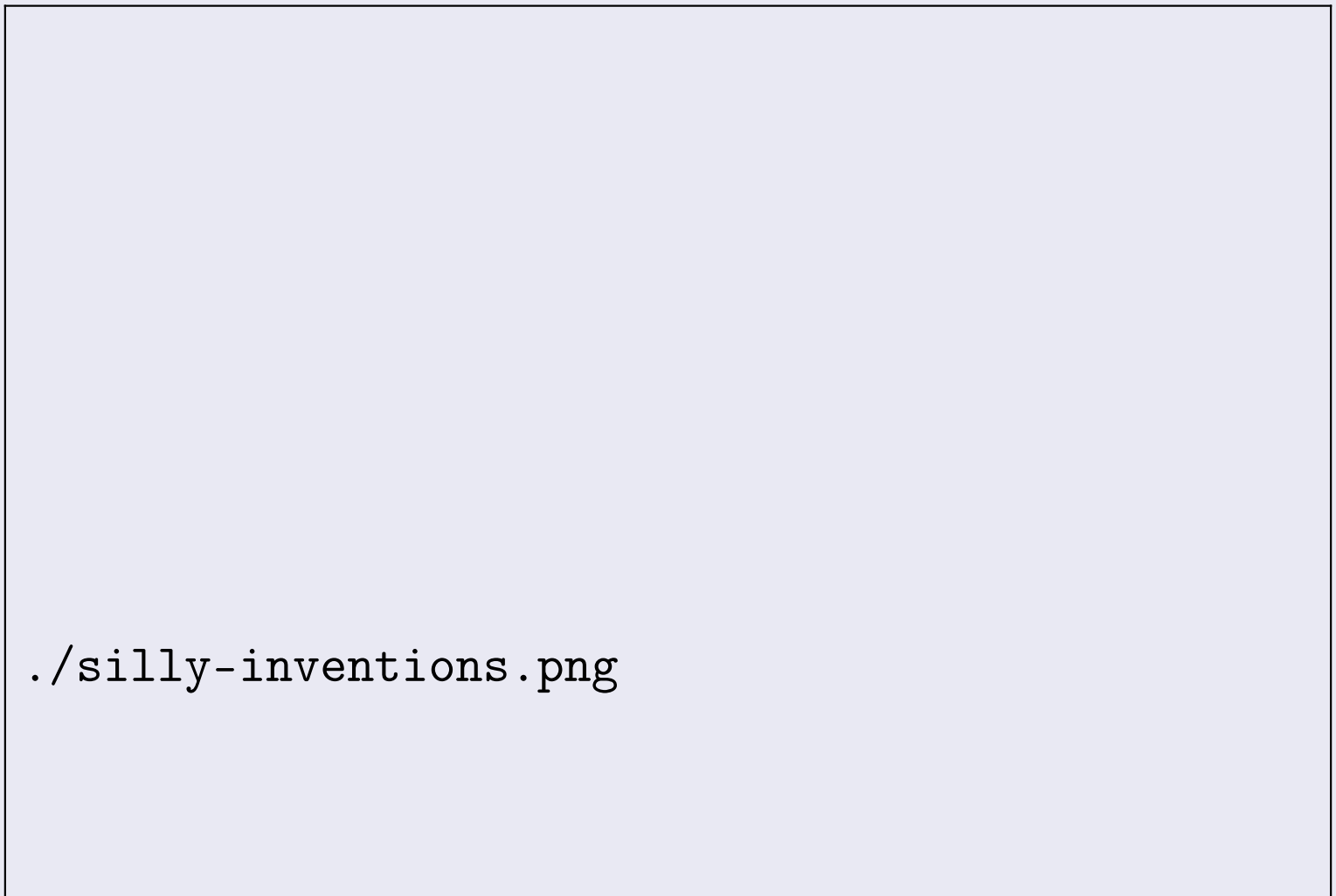
- Tweet Sentiment
- Company categorization
- Labeling parts of speech
- `http://github.com/semiosis/prompts/blob/master/prompts/tweet-sentiment-classifier.prompt`
- `http://github.com/semiosis/prompts/blob/master/prompts/keyword-extraction.prompt`

Tasks suitable for GPT-3

Generation

- Idea Generator

Come up with silly inventions.



Tasks suitable for GPT-3

Conversation

- Q&A agent
- Sarcastic chatbot

`http://github.com/semiosis/prompts/blob/master/prompts/sarcastic-response.prompt`

Design patterns

Taken from Prompt Design 101.
These are manual techniques which should be encoded in a DSL when generating prompts.

1. Reflective description of the task

State what the prompt does at the start.
At the start of the example we state in plain language what the classifier does:

```
1  This is a tweet sentiment classifier.
```

By stating this up front, it helps the API understand much more quickly what the goal of the response is supposed to be and you'll end needing to provide fewer examples.

Design patterns

Taken from Prompt Design 101.
These are manual techniques which should be encoded in a DSL when generating prompts.

2. Use separators between examples

Example: ###.
You can use other characters or line breaks, but ### works pretty consistently and is also an easy to use stop sequence.
Whatever separator you use, make sure that it's clear to the API where an example starts and stops.

Design patterns

Taken from Prompt Design 101.
These are manual techniques which should be encoded in a DSL when generating prompts.

3. Mutiplexer Part 1

Make a prompt more efficient / cheaper.
Design it to generate multiple results from one API call.

```
This is a tweet sentiment classifier
Tweet: "I loved the new Batman movie!"
Sentiment: Positive
###
Tweet: "I hate it when my phone battery dies"
Sentiment: Negative
###
Tweet: "My day has been [U+1F44D]"
Sentiment: Positive
###
Tweet: "This is the link to the article"
Sentiment: Neutral
###
Tweet text
```

Taken from Prompt Design 101.
These are manual techniques which should be encoded in a DSL when generating prompts.

3. Mulitplexer Part 2

```
1. "I loved the new Batman movie!"
2. "I hate it when my phone battery dies"
3. "My day has been [U+1F44D]"
4. "This is the link to the article"
5. "This new music video blew my mind"

Tweet sentiment ratings:
1: Positive
2: Negative
3: Positive
4: Neutral
5: Positive

###
Tweet text
```

Taken from Prompt Design 101.
These are manual techniques which should be encoded in a DSL when generating prompts.

3. Multiplexer Part 3

```
"I can't stand homework"
"This sucks. I'm bored [U+1F620]"
"I can't wait for Halloween!!!"
"My cat is adorable [U+2764][U+FE0F][U+2764][U+FE0F]"
"I hate chocolate"
Tweet sentiment ratings:
1.
```

Query Reformulation

https://www.sciencedirect.com/topics/computer-science/query-reformulation

You can improve the quality of the responses by making a longer more diverse list in your prompt.

One way to do that is to start off with one example, let the API generate more and select the ones that you like best and add them to the list.

A few more high-quality variations can dramatically improve the quality of the responses.

pen.el: Prompt Engineering in emacs

pen.el facilitates the creation, development, discovery and usage of prompts to a Language Model such as GPT-3.

- Create elisp functions based on GPT-3 prompts
- Chain GPT-3 queries together using keyboard macros and functions
- Interactively query, generate and transfrom both prose and code
- Use GPT-3 as a search engine within emacs

pen.el modes Part 1

Prompt-Engineering Minor Mode

`prompt-engineering-mode` is a global minor mode for emacs that provides keybindings for creating and executing prompts generally across emacs.

Prompt Description Major Mode

`prompt-description-mode` is a major mode for editing `.prompt` files.

The `.prompt` file format is based on YAML and an associated schema, which defines the keys which are expected.

pen.el modes Part 2

Pen Messenger Minor Mode

`pen-messenger-mode` is a minor mode for enhancing an emacs-based messenger client with GPT-3 capabilities, such as emoji generation.

Pen Conversation Mode

`prompt-conversation-mode` is a major mode designed to facilitate ongoing conversation with a prompt-based GPT-3 chatbot.

GPT-3 Training Mode

The goal of this mode is to facilitate the workflow of training on OpenAI's API.

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

```
1  vars:
2  - "notes"
3  examples:
4  - |+
5      Tom: Profits up 50%
6      Jane: New servers are online
7      Kjel: Need more time to fix software
8      Jane: Happy to help
9      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

Vexate a simple instruction

./complicate.png

Something funny

How to crack an egg

./crack-an-egg.png

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