

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](https://github.com/mullikine/presentation-prompt-engineering-in-emacs)  
[github1s.com/semiosis/exemplary](https://github.com/semiosis/exemplary)  
[github1s.com/semiosis/pen.el](https://github.com/semiosis/pen.el)  
[github1s.com/semiosis/prompts](https://github.com/semiosis/prompts)  
[github1s.com/semiosis/prompt-engineering-patterns](https://github.com/semiosis/prompt-engineering-patterns)  
[github1s.com/minimaxir/gpt-3-client](https://github.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.

## Characteristics

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

# Prompts as functions

# 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>
```

# Create a prompt

Ask the audience

# Tutorials

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