

# Prompt Engineering

**John Berryman**

# Hi! I'm John Berryman

career  
1

- Aerospace Engineer (just long enough to get the merit badge)

career  
2

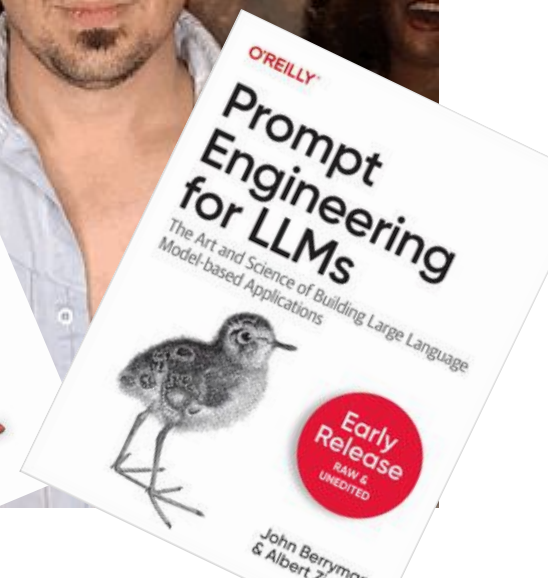
- Search Technology Consultant
- Eventbrite Search Engineer
- Wrote a book. (Swore never to do so again.)
- GitHub Code Search

career  
3

- GitHub Data Science
- GitHub Copilot Prompt Engineer
- Writing a book. (*But why!?*)

career  
4

- LLM Application Consulting – Arcturus?



# What is a Language Model?

JO

**You**

What is a language model?



**ChatGPT**

A language model is a type of artificial intelligence system that is trained to understand and generate human-like text. It learns the structure, grammar, and semantics of a language by processing vast amounts of textual data. The primary goal of a language model is to predict the probability of the next word or sequence of words in a given context.

How has *this* taking the world by storm?



# What is a Large Language Model?

It's the same thing, just a *lot* more accurate.

- c. 2014 the top language models were Recurrent Neural Networks
- Sept 2014 Attention mechanism *Jointly Learning to Align and Translate by* previous context.
- Jun 2017 got rid of RNNs *Jointly Learning to Align and Translate by* produced
- Jun 2018 chopped the Transformer architecture *Jointly Learning to Align and Translate by*
- Jun 2018 chopped the *Generative Pre-Training* only use the decoder *Understanding by*!
- Feb 2019 GPT-2 was trained on 10x the data in *Language Models are Unsupervised Multitask Learners* ... and things started getting weird.

**Our model, called GPT-2 (a successor to GPT), was trained simply to predict the next word in 40GB of Internet text. Due to our concerns about malicious applications of the technology, we are not releasing the trained model. (ref)**

# What is a Large Language Model

- GPT-2 was beating models *trained for specific tasks*
  - missing word prediction
  - pronoun understanding
  - part of speech tagging
  - text compression
  - summarization
  - sentiment analysis
  - entity extraction
  - question answering
  - translation
  - content generation
- But with great power comes great responsibility. Models can:
  - Generate misleading news articles
  - Impersonate others online
  - Automate the production of abusive or faked content to post on social media
  - Automate the production of spam/phishing content



(These are all from the [Feb 2019 GPT-2 release article](#).)

# What is a Large Language Model

- GPT-2 was beating
  - missing word prediction
  - pronoun understanding
  - part of speech tagging
  - text compression

- But with great power comes great responsibility
  - Generate misleading text
  - Impersonate other people
  - Automate the production of fake news
  - Automate the production of hate speech

Our model, called GPT-2 (a successor to GPT), was trained simply to predict the next word in 40GB of Internet text.  
**...And we figured out that now you can just ask it to do stuff and it will!**

**IT'S AMAZING**

***(But also it will help you make bombs, and drugs, and overthrow governments. So...)***

**Due to our concerns about malicious applications of the technology, we are not releasing the trained model.**



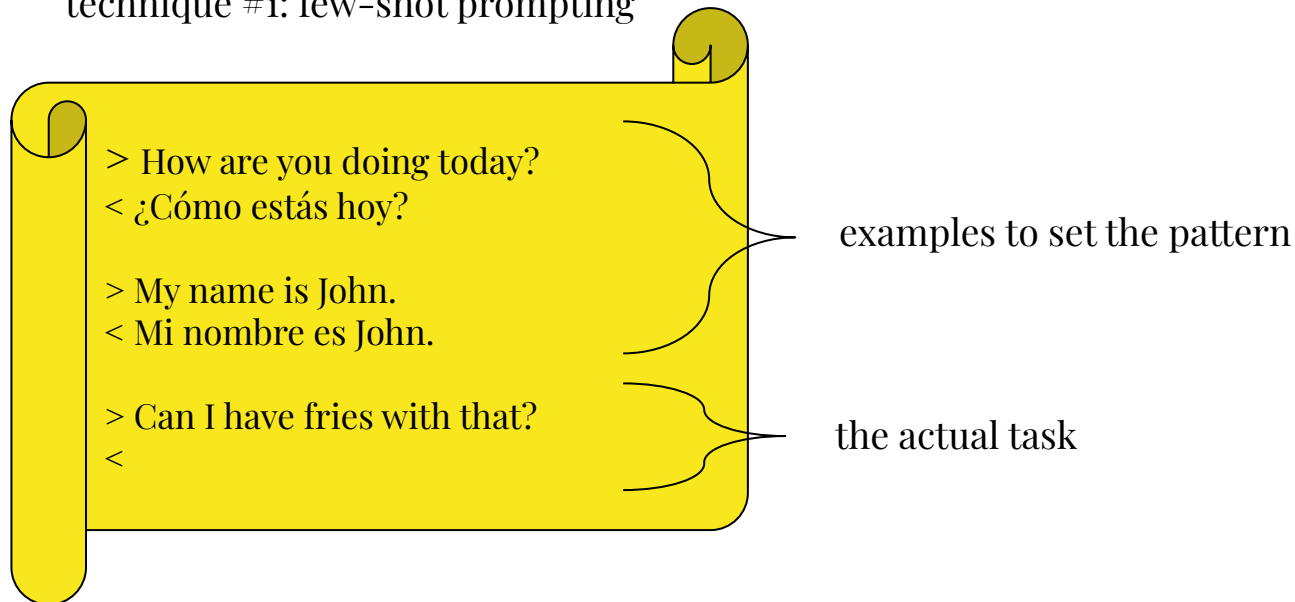
media

(These are all from

[GPT-2 release article.](#))

# Prompt Crafting

technique #1: few-shot prompting



# Prompt Crafting

technique #2: chain-of-thought  
reasoning

Q: It takes one baker an hour to  
make a cake. How long does it  
take 3 bakers to make 3 cakes?

A:

*"Chain-of-Thought Prompting Elicits  
Reasoning in Large Language Models"*

Jan 2022



# Prompt Crafting

technique #2: chain-of-thought reasoning

Q: Jim is twice as old as Steve. Jim is 12 years how old is Steve.

A: In equation form:  $12=2*a$   
where  $a$  is Steve's age. Dividing both sides by 2 we see that  $a=6$ .  
Steve is 6 years old.

Q: It takes one baker an hour to make a cake. How long does it take 3 bakers to make 3 cakes?

A:

*"Chain-of-Thought Prompting Elicits Reasoning in Large Language Models"*

Jan 2022

# Prompt Crafting

technique #2: chain-of-thought  
reasoning

Q: It takes one baker an hour to make a cake. How long does it take 3 bakers to make 3 cakes?  
A: Let's think step-by-step.

*"Large Language Models are Zero-Shot Reasoners" May 2022*

# Prompt Crafting

technique #3: document mimicry

*What if you found this scrap of  
paper on the ground?*

My cable is out! And I'm going to  
miss the Superbowl!

*What do you think the rest of the  
paper would say?*

# Prompt 0

Document  
type is  
transcript

Technique #3: document mimicry

It tells a story  
to condition a  
particular  
response.

It uses  
markdown to  
establish  
structure

```
# IT Support Assistant  
The following is a transcript  
between an award winning IT  
support rep and a customer.
```

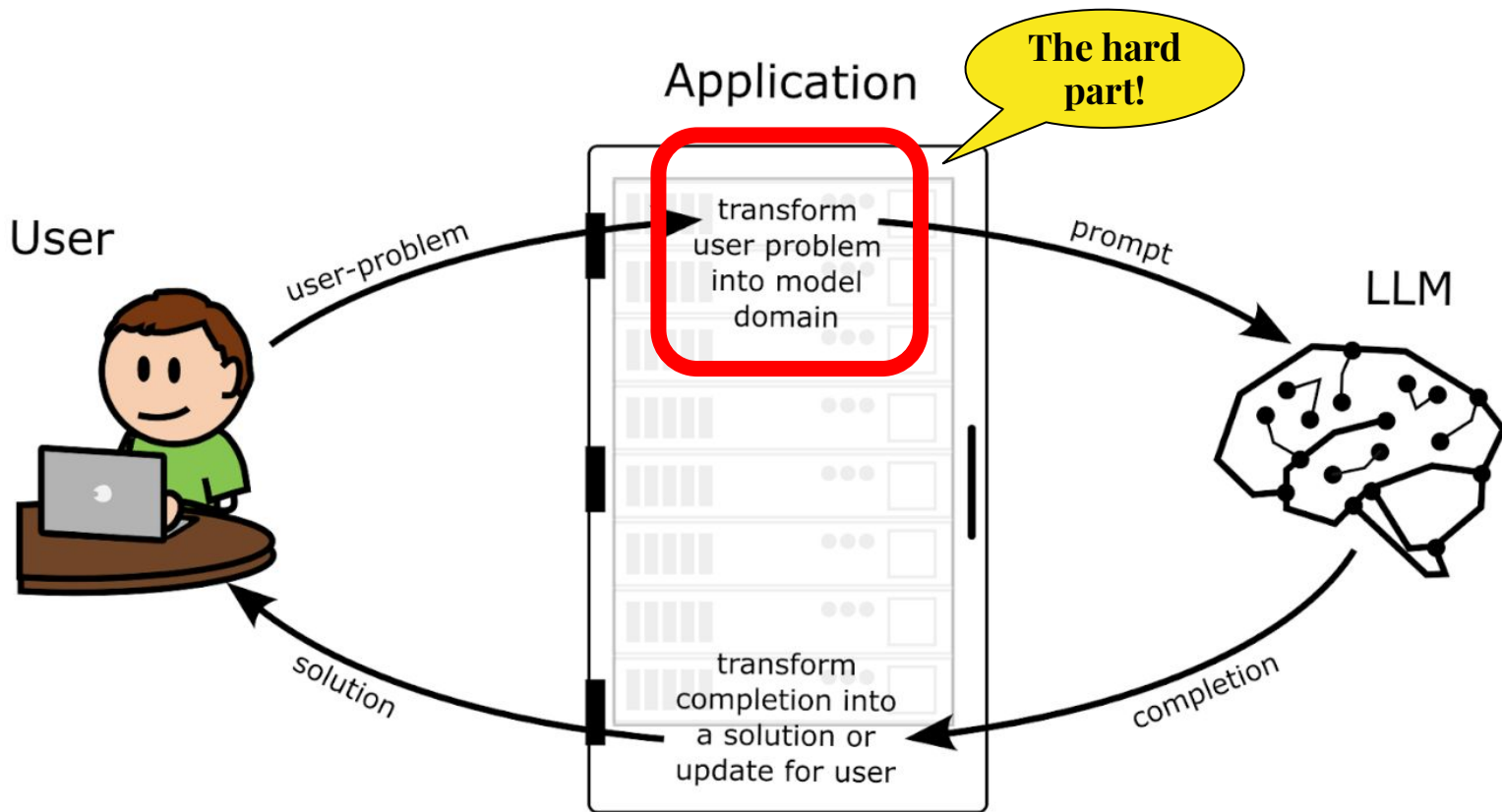
```
## Customer:  
My cable is out! And I'm going to  
miss the Superbowl!
```

```
## Support Assistant:
```

## Prompt Crafting Intuition:

- LLMs understand better when you use familiar language and constructs.
- LLMs get distracted. Don't fill the prompt with lots of "just in case" information.
- LLMs aren't psychic. If information is neither in training or in the prompt, then they don't know it.
- If you look at the prompt and *you* can't make sense of it, a LLMs is hopeless.

# Building LLM Applications





# Creating the Prompt


- Collect context
- Rank context
- Trim context
- Assembling Prompt


# Creating the Prompt: Copilot Code Completion

- Collect context – current document, open tabs, symbols, file path
- Rank context – file path  $\gg$  current document  $\gg$  open tabs  $\gg$  symbols
- Trim context – drop open tab snippets; truncate current document
- Assembling Prompt

file path 

snippet from  
open tab 

current  
document 

cursor 

```
// pkg/skills/search.go

// <consider this snippet from ../skill.go>
// type Skill interface {
//     Execute(data []byte) (refs, error)
// }
// </end snippet>

package searchskill

import (
    "context"
    "encoding/json"
    "fmt"
    "strings"
    "time"
)

type Skill struct {
    }
```



# The Introduction of Chat

API

```
messages =
[
  {
    "role": "system"
    "content": "You are
an award winning
support staff
representative that
helps customers."
  },

  {"role": "user",
   "content": "My cable
is out! And I'm going
to miss the
Superbowl!"
  }
]
```

document

```
<|im_start|> system
You are an award winning IT
support rep. Help the user with
their request.<|im_stop|>

<|im_start|> user
My cable is out! And I'm going to
miss the Superbowl!<|im_stop|>

<|im_start|> assistant
Let's figure out how to diagnose
your problem...
```

benefits

- Really easy for users to build assistants.
  - System messages make controlling behavior easy.
  - The assistant always responds with an complete thought and then stops.
- Safety is baked in:
  - Assistant will (almost) never respond with insults or instructions to make bombs
  - Assistant will (almost) never hallucinate false information.
  - Prompt injection is (almost) impossible.

# The Introduction of Tools

```
{  
  "type": "function",  
  "function": {  
    "name": "get_weather",  
    "description": "Get the weather",  
    "parameters": {  
      "type": "object",  
      "properties": {  
        "location": {  
          "type": "string",  
          "description": "The city and state",  
        },  
        "unit": {  
          "type": "string",  
          "description": "degrees Fahrenheit or Celsius",  
          "enum": ["celsius", "fahrenheit"]  
        },  
        "required": ["location"],  
      },  
    },  
  },  
}
```

Input:

```
{  
  "role": "user",  
  "content": "What's the weather  
             like in Miami?"  
}
```

Function Call:

```
{  
  "role": "assistant",  
  "function": {  
    "name": "get_weather",  
    "arguments": {  
      "location": "Miami, FL"  
    }  
  }  
}
```

Real API request:

```
curl  
http://weathernow.com/miami/FL?deg=f  
{  
  "temp": 78  
}
```

Function Response:

```
{  
  "role": "tool",  
  "name": "get_weather",  
  "content": "78°F"  
}
```

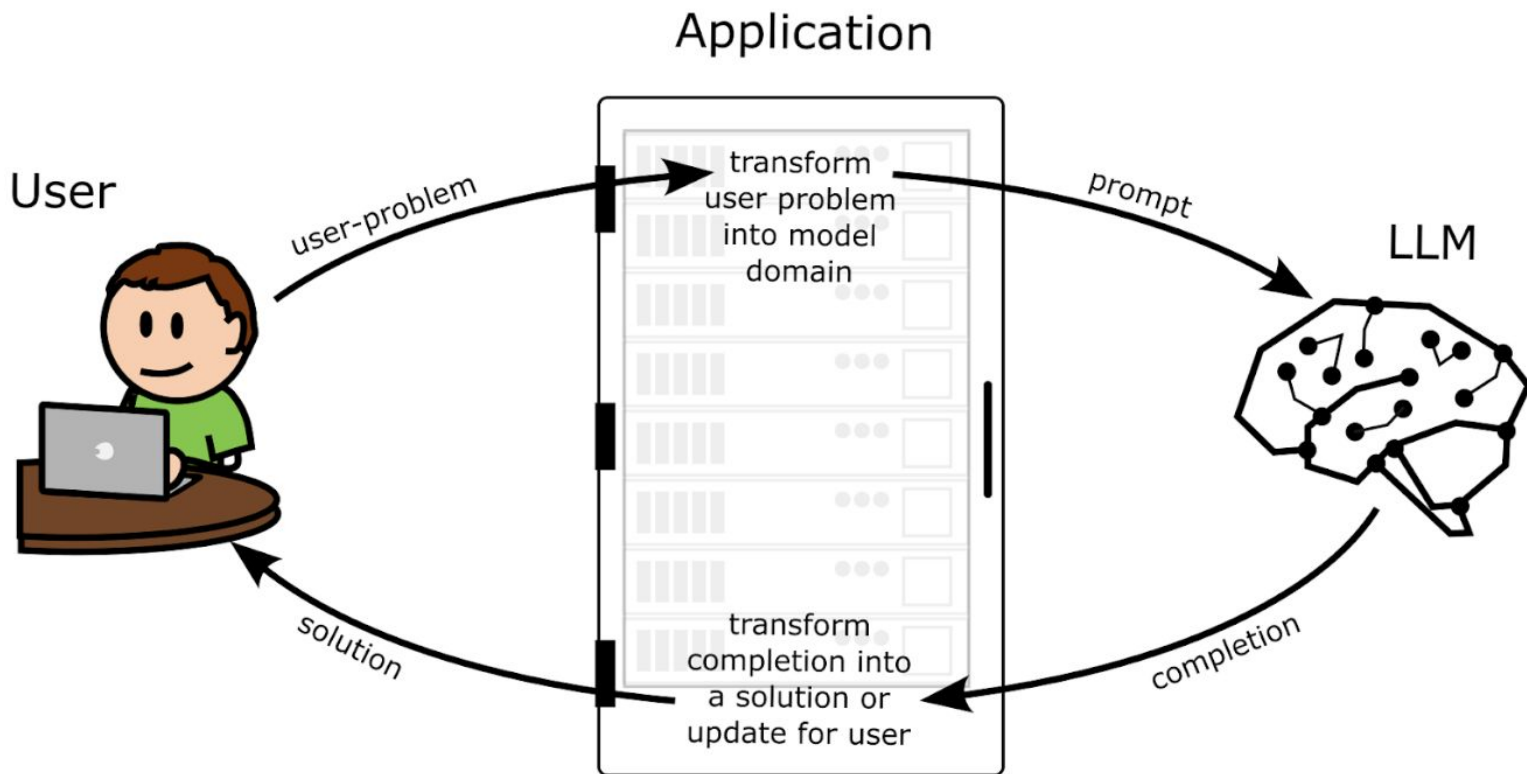
Assistant Response:

```
{  
  "role": "assistant",  
  "content": "It's a balmy 78°F"  
}
```

- Agents can reach out into the real world
  - Read information
  - Write information
- Model chooses to answer in text or run a tool
- Tools can be called in series or in parallel
- Tools can be interleaved with user and assistant text

[function calling Jun 13, 2022](#)

# Building LLM Applications



# Building LLM Applications: Bag of Tools Agent

functions:

- `getTemp ()`
- `setTemp (degreesF)`

user: make it 2 degrees warmer in here

assistant: `getTemp ()`

function: 70°F

assistant: `setTemp (72)`

function: success

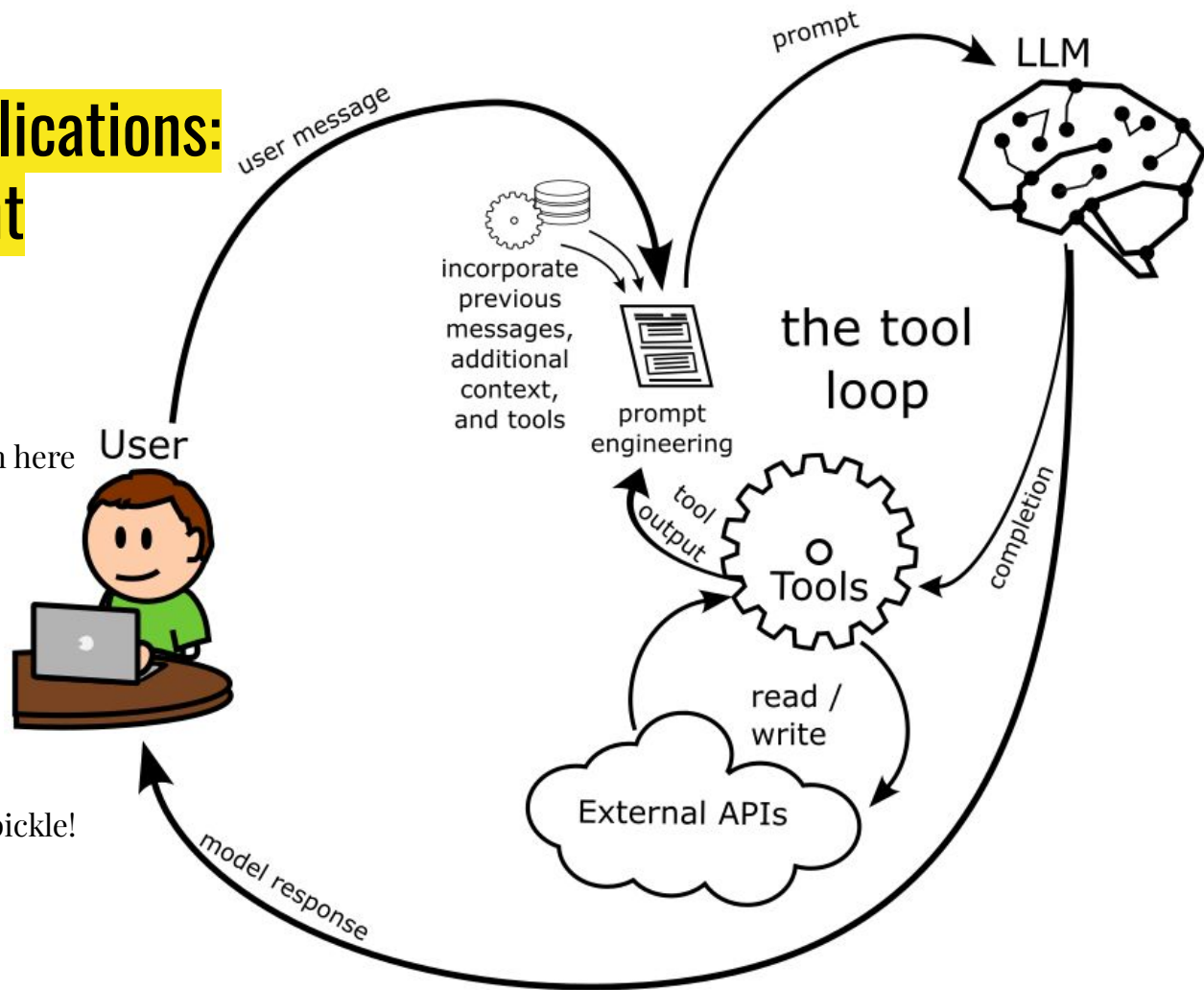
assistant: Done!

user: actually... put it back

assistant: `setTemp (70)`

function: success

assistant: Done again, you fickle pickle!

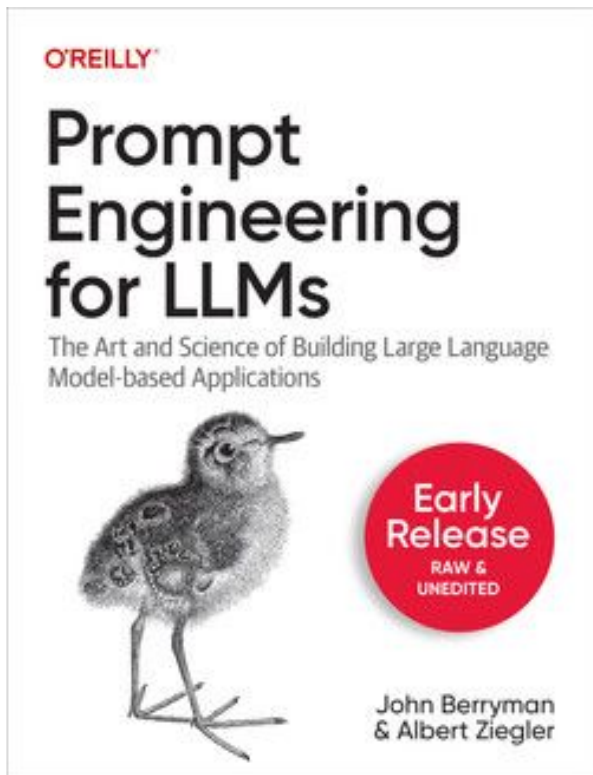


# Creating the Prompt: Copilot Chat

- Collect context:
  - *References* – files, snippets, issues, that users attach or tools produce
  - Prior messages
- Rank, Trim and Assemble:
  - must fit:
    - system message
    - function definitions (if we plan to use them)
    - user's most recent message
  - fit if possible:
    - all the function calls and evals that follow
    - the references that belong to each message
    - historic messages (most recent being most important)
  - fallback to no-function usage if we can't fit with (causes Assistant to respond and turn to complete)

# Tips for Defining Tools

- Don't have "too many" tools - look for evidence of collisions
- Name tools simply and clearly (*and in TypeScript format?*)
- Don't copy/paste your API - keep arguments simple and few
- Keep function and arg descriptions short and consider what the model knows
  - It probably understands public documentation.
  - It doesn't know about internal company acronyms.
- More on arguments
  - Nest arguments don't retain descriptions
  - You can use `enum` and `default`, but not `minimum`, `maximum`...
- Skill output - don't include extra "just-in-case" content
- Skill errors - when reasonable, send errors to model (validation errors)



# Questions?

***P.S.*** I'm also available for LLM application consulting at [\*jfberryman@gmail.com\*](mailto:jfberryman@gmail.com)

**Questions?**