

Looking Ahead

Human history only makes sense on a logarithmic scale. It took humans countless eons to figure out farming, millennia beyond that to invent writing, centuries more to invent the steam engine, and decades more to invent the automobile, computer, and smartphone. Just a few years after that, around 2012, deep learning appeared on the scene.

OpenAI's GPT-2 was announced in 2019, and then ChatGPT was announced in 2022. This ignited an explosion of development around LLMs. Many companies have jumped into the fray—Anthropic, Google, Microsoft, Meta, xAI, NVIDIA, Mistral, and more—all building new LLMs that have leap-frogged the previous ones in capability, capacity, and speed. In mere months, LLMs have morphed from document completion engines, to chat engines, to agents that can interact with the outside world.

Buckle up, readers. If you think the pace of change is fast now, then just wait, it's only going to get faster. (Maybe that Ray Kurzweil guy was on to something!) In this final chapter, let's look ahead to some of the developments on our horizon and how they will change your work as a prompt engineer.

Multimodality

There is a huge push toward the use of multimodal models. OpenAI kicked off this trend with GPT-4, which was able to process images as part of the prompt. Although OpenAI has not disclosed details about how exactly the model works, most likely, it closely follows the methods published in [academic literature](#).

In one such method, a convolutional network is used to convert image features into embedding vectors of the same dimensions as those used for text tokens. The image vectors are imbued with positional information so that the relationships among the features in the image are retained. The image and text vectors are then concatenated. Finally, the transformer architecture processes this information in much the same way that text-only LLMs process pure text (see [Chapter 2](#)). Multimodality can naively be extended to video input—all you have to do is sample images from the video, as demonstrated in this [OpenAI cookbook](#).

As they mature, multimodal models are going to be extremely useful in domains that can't be captured by text. For example, it's easy to imagine how these models can be used to make the world much more accessible to a person with vision impairment. A vision model could help them read signs, find buildings, and navigate unfamiliar environments.

Another reason that multimodal models are important is because they give the models access to a large volume of rich training data. Over the past few years, there has been increasing concern that we might actually run out of training data! The models are large enough that they can learn increasingly intricate details about the world. However, if we overtrain with a set of data that is too small, then these models can overfit—effectively memorizing text rather than modeling how the world works. Amazingly, literally *the text of the entire public internet* may not be enough for the next generation of large models.

However, when we incorporate images and video into the training, we gain access to vastly more content. Moreover, the image and video content carry a very different type of information that can help models better understand the world around them; with access to images, it should become much simpler for models to understand tasks related to spatial reasoning, social cues, physical common sense, and much more.

As a prompt engineer, when you build future LLM applications, you are likely to include images and videos in the prompt—and even though they constitute a completely different form of information, you can make use of some of the lessons in this book when dealing with them. Remember to include only images that are relevant to the conversation at hand so that the model doesn't get distracted. Frame the images with text that properly introduces their role in the conversation and make use of patterns and motifs that were in the training data. For instance, don't introduce a new type of diagram to convey information when there is a common format that is more readily available on the internet.

User Experience and User Interface

The UI of many consumer applications is currently moving toward conversational interactions. It makes sense, right? Humans have been speaking to each other for 200,000 years but only clicking buttons on screens for the past 40. In this section, we'll focus on a new element of the conversation that has caught our attention—artifacts—or, as we like to call them, *stateful objects of discourse*.

Think about it. In day-to-day collaboration with other humans, we often talk about a *thing*—the *object of discourse*. And as we talk about it, we can talk about how we want to change it, we can actually modify it, and we can talk about how it has changed over time—meaning we can talk about its state. Pair programming is a great example here. The files are the objects of discourse, and during pairing, we can change them and talk about how they are changing.

In most chat applications today, the assistants don't address the object of the conversation in a stateful way. If you ask ChatGPT to write a function and then later modify it, it can't go back and update the contents of the function. Instead, it rewrites the function over and over again, from scratch. Rather than having one object whose state has evolved, ChatGPT writes N objects into the conversation.

What's more, it is difficult to specify which object you're talking about, especially if you have multiple objects in play. Which function were you talking about? Which version? These problems make it difficult to work with the assistant *on* something rather than just having a conversation in which ideas are expected to fly by and go out of scope.

As we were wrapping up this book, Anthropic introduced *Artifacts*, which represents a step in the direction of stateful objects of discourse. In a conversation with Anthropic's Claude, the Artifact *is* the stateful object of discourse. It can be an SVG image, an HTML file, a mermaid diagram, code, or any other type of text fragment. During a conversation, the user works with the assistant to modify the Artifact until it reaches the user's expectations. And while the back-and-forth conversation is captured in a transcript on the left side of the screen, the Artifact they are discussing remains—statefully—on the right side of the screen (see [Figure 11-1](#)). If the user asks for the Artifact to be modified, then the state of the Artifact is updated in place rather than regurgitated over and over again in the transcript.

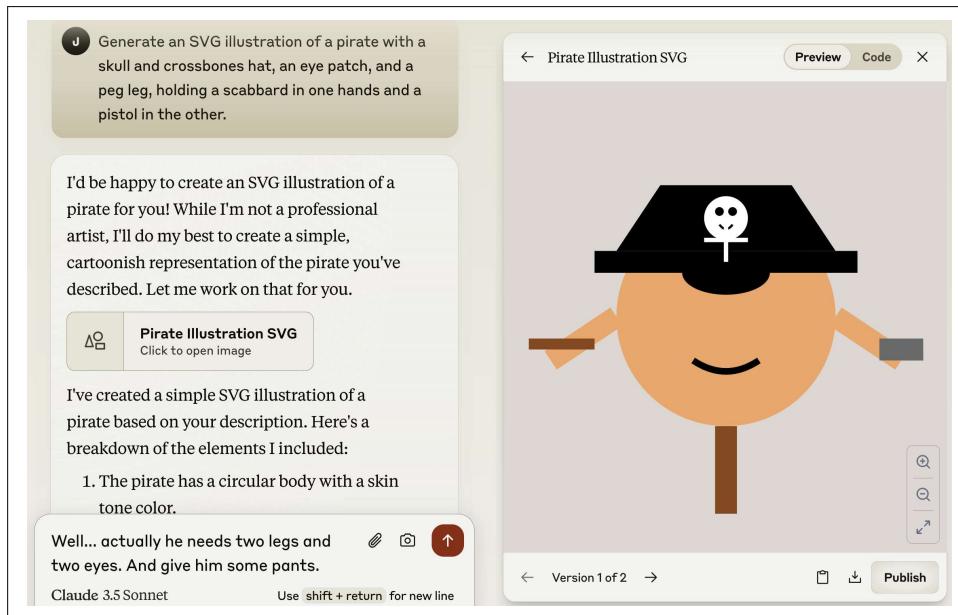


Figure 11-1. Working with Claude to draw a peg-legged pirate with a patch over his eye while statefully discussing the fact that the image appears to be missing actual legs and eyes

Claude's Artifacts paradigm is very close to what we have in mind, but there's still room for improvement. For instance, much of the change is just in the UI, rather than in the prompt engineering. When you ask for a change, Claude is still rewriting the entire Artifact from scratch; it just knows to put the Artifact in the right pane. This form of editing might not scale well to longer documents.

Additionally, it's not easy to interact with multiple Artifacts at once. Claude's interface assumes one Artifact at a time. If you start talking about a different Artifact, then the UI treats it as if it were just a different version of the previous interface. Another problem with multiple Artifacts is that it's hard to refer to them. It would be nice if both the UI and the prompt included shorthand names for the items.

Finally, Claude's interface and the prompt engineering (for that matter) don't allow the user to edit the Artifact. If you see a small problem that you could easily fix, the only way to address it is to tell the assistant to fix the problem for you (by retyping the whole file). It would be a better experience if the user could update the Artifact and then have this update reflected in the next prompt so that the model is aware of the change.

When building LLM interfaces in your own LLM applications, it can be a good idea to lean into a conversational interface since conversation is so intuitive for humans. But if so, you need to invest time to really get it right—it's easy to whip up something bare-bones with LLMs, but it will be a gimmicky distraction rather than something that provides true benefits, unless it's properly thought through and fleshed out.

Model designers are aware of this need, and they innovate to support it. Tools were a great improvement—they gave the assistants the ability to take action in the world. Artifacts are similarly useful—they allow conversations to be about *things* (i.e., stateful objects of discourse). What's next?

The conversational UI is also a great way to keep users in the loop. As we discussed in [Chapter 8](#), models tend to stray off course if left to their own devices. But in a close conversational interaction, users can identify problems early and put the assistant back on course.

Intelligence

Has anyone noticed that LLMs are getting a lot smarter? Yeah...and they're going to keep getting smarter too. Let's look at some upcoming developments.

For one thing, we're getting smarter with our benchmarks. Benchmarks are problem sets with known answers that allow us to measure how well models perform relative to humans and relative to one another. At this point, several of the most useful benchmarks have saturated (see [Figure 11-2](#)), meaning that the leading models tend to ace them. This makes the benchmarks useless for evaluating model improvements. There are two reasons that models saturate the benchmarks: (1) models really are getting smarter—a *good thing*, and (2) models are cheating by training on the benchmarks—a *very bad thing*. The “cheating” isn’t intentional; it’s just that after a couple of years, information from benchmarks gets duplicated (verbatim or through descriptions) all over the internet and accidentally pulled into training.

To fix this, we in the AI community are being diligent in upgrading our benchmarks (for instance, on the [Open LLM Leaderboard 2](#)). We have also started using nonmemorable benchmarks such as [ARC-AGI](#), which is effectively a set of psychometric intelligence tests composed of patterns of shapes. They test how well the individual—or LLM—can understand and reproduce novel patterns, and it’s impossible to memorize all of the test questions because they belong to a very large space of possible tests, they are algorithmically generated, and you can always generate more of them.

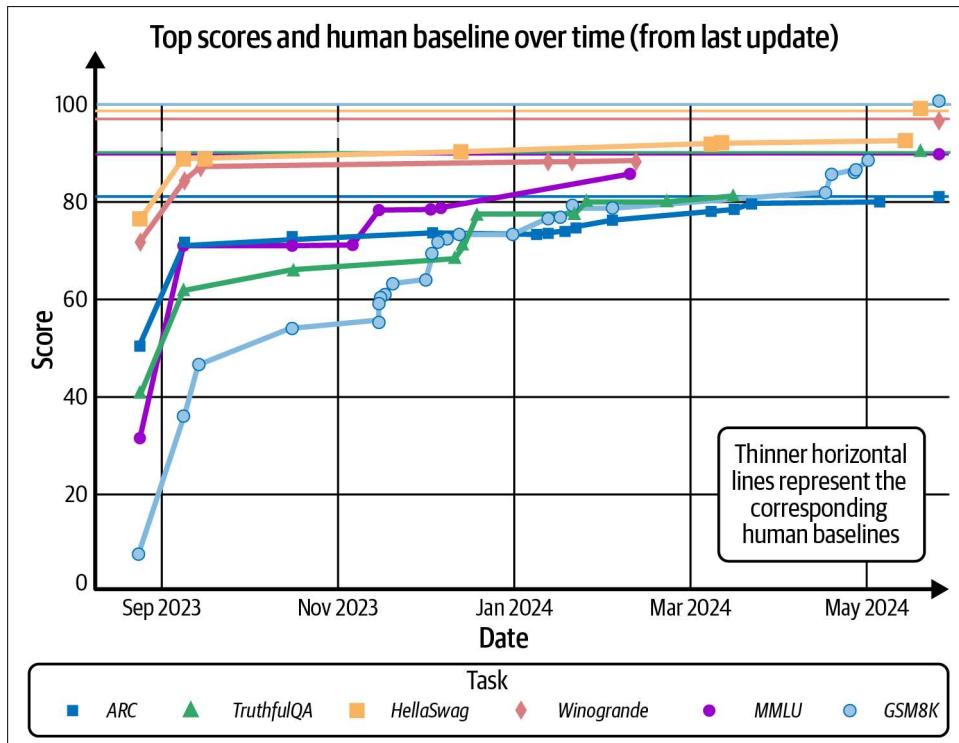


Figure 11-2. Popular benchmarks saturate over time, making them useless as benchmarks going forward

We're also getting smarter with model training. You can actually watch this happen when you use ChatGPT or its competitors because, thanks to better RLHF training (refer back to [Chapter 3](#)), the models are doing a much better job of expressing their chain-of-thought reasoning, which inevitably leads to more useful responses.

Next, we're getting more creative with training approaches. For example, large models generally don't utilize their full capacity, so, if you can figure out a way to convey knowledge to a small model, then you can effectively compress the information of the large model into the small model. A training approach known as *knowledge distillation* uses a large model as a "teacher" of a small model. Rather than training the small model to predict the next token, knowledge distillation trains the small model to mimic the large model by predicting the full set of probabilities for the next token. This richer set of training data allows smaller models to be trained quickly, with only a slight decrease in accuracy compared to their teacher models. In exchange for the hit in accuracy, these small models are significantly cheaper and faster than the large models they were trained from.

Besides training, model improvements will come from architectural innovation. Here, we name just a few. For one, models are getting smaller and faster through *quantization* approaches, in which, instead of representing parameters as 32-bit floating point numbers, you can approximate them with 8-bit parameters, reducing the model size considerably and correspondingly decreasing the cost and increasing the speed.

In your prompt-engineering work, you should be able to expect these trends to continue. If something is too expensive today, it will be cheaper tomorrow. If something is too slow today, it will be faster tomorrow. If something doesn't fit in the context today, it will fit tomorrow. And if the model isn't smart enough today, it will be tomorrow. However, always remember that even though models will get smarter, they will never be psychic. If the prompt doesn't contain the information that *you* would need to solve the problem, then it's probably insufficient for the model as well.

Conclusion

If we had to sum up the main lessons from this book, there would be two:

1. LLMs are nothing more than text completion engines that mimic the text they see during training.
2. You should empathize with the LLM and understand how it thinks.

Regarding the first lesson, when we started writing this book, the only models we had access to were completion models—give them a portion of a document (a.k.a. *the prompt*), and they would generate plausible text to complete the document. But then, the chat APIs came to dominate, tools came along next, and perhaps, Artifacts will be the next big thing. But even still, at their core, LLMs are just completing documents so that they resemble other documents the model has been taught to “like.” It’s just that now, the documents look like a chat transcript.

The prompt-engineering lesson here is to follow the well-trodden course (the Little Red Riding Hood principle from [Chapter 4](#))—make your prompts follow the patterns and motifs seen in training data and you will be much more likely to get completions that are well behaved and easy to anticipate. For instance, you can format complex text as markdown, and if there is a standard document format for information you are communicating to the LLM, then you’ll have better luck using that format than coming up with a new one the model has never seen.

Regarding the second lesson, on empathy, think of an LLM as your big, dumb mechanical friend who happens to know much of the content of the internet. Here are some things that will help you understand:

LLMs are easily distracted

Don't fill up the prompt with useless information that—*cross your fingers*—might just help. Make sure every piece of information matters.

LLMs should be able to decipher the prompt

If, as a human, you can't understand the fully rendered prompt, then there is a very high chance that the LLM will be equally confused.

LLMs need to be led

Provide explicit instructions for what is to be accomplished and, when appropriate, provide examples demonstrating how the task should proceed.

LLMs aren't psychic

As the prompt engineer, it's your job to make sure the prompt contains the information that the model needs to address the problem. Alternatively, give the model the tools and the instructions to retrieve it.

LLMs don't have internal monologues

If the LLM is allowed to think about the problem out loud (in a chain of thought), then it will be much easier for it to come to a useful solution.

Hopefully, this book has given you all that you need to jump headlong into prompt engineering and LLM application development. Be assured, the accelerating change that we currently experience will continue. Since software will be easier to create, you will find more examples of highly individualized apps or even disposable apps. Applications will take on the nondeterministic nature of the LLMs, leading to more flexible and open-ended experiences. Development will change. You will work in tandem with an AI assistant to get your work done—if you don't already.

Whatever shape the world finds itself in, it will be a shape of your making. As a prompt engineer, you have the tools at hand and the know-how to build a future of your choosing. Embrace the acceleration. Keep experimenting. Stay flexible. In the words of the late Sir Terry Pratchett:

The whole world is tap-dancing on quicksand. In this case, the prize goes to the best dancer.¹

¹ Terry Pratchett, *The Fifth Elephant* (New York: Doubleday, 1999)

Index

Symbols

" ("), 135
(hash sign), 69
& (&), 135
&apos ('), 135
> (>), 135
< (<), 135
" ("), 135
' (&apos), 135
* (asterisk), 69
< (<), 135
> (>), 135
\nclass, 153
\ndef, 153
\nif, 153
\n\tdef, 153
` (backticks), 69, 131, 151
... (ellipsis), 140

A

A/B testing, 239
acceptance rate metric, 83
achieved impact, 240, 242
actions, 48
additive greedy approach, 145
advanced workflows
 allowing LLM agents to drive workflows, 218
 versus basic workflows, 217
 roles and delegation, 219
 stateful task agents, 219
advice conversations, 127-130, 137, 234
agency, 169 (see also conversational agency)
agents

allowing to drive workflows, 218
assembling crews of, 220
autonomous, 187
defining and delegating to, 219
in reward models, 48
stateful task agents, 219
AGI (artificial general intelligence), 199
Airflow, 212
ALFWorld, 185
Algolia, 116
alignment tax, 51
Amazon book reviews, 93
analytic reports, 130-132, 137
anchoring, 95-97
Anthropic
 Artifacts prompt, 133-135, 247
 choosing a provider, 161
 introduction of HHH alignment, 47
application design
 consistency in, 90, 93
 converting user's problem to model domain, 68-73
 determining model size, 73
 evaluating application quality, 81-83
 evaluating completion quality, 154
 exploring loop complexity, 77-81
 feedforward pass, 75-81, 100
 fine-tuning, 74
 latency, 73
 “loop” of interaction, 65-67, 75-81, 224
 role of LLM applications, 65
 transforming back to user domain, 74
 user's problem domain, 67
 using LLMs to complete prompts, 73

- application state, 78
application urgency, 100
arbitrary tasks, generating on the fly, 218
ARC-AGI, 249
arguments
 argument hallucination, 180
 best practices for, 179
 naming, 179
Artifacts
 abridged Artifacts prompt, 133-135
 description of, 133
 examples of, 133
 stateful object of discourse, 247
artifacts, 187, 189
Artificial Analysis website, 161
artificial general intelligence (AGI), 199
asides, 137
asterisk (*), 69
attention game, 15, 95
attention mechanism, 7
authorization, 181, 196, 211
auto-regressive models, 29-32
AutoGen, 210, 219
AutoGPT, 12
autonomous agents, 187
available_functions definition, 172
- B**
- background context, 137
backticks (` `), 69, 131, 151
backward-and-downward vision, 42
base models, 45, 47
basic workflows
 versus advanced workflows, 217
 assembly of, 211-217
 optimizing, 216
 Shopify plug-in marketing example, 214-217
 steps required to build, 204
 tasks, 205
batch workflows, 214
beginning and end (preamble elements), 151, 155
benchmarks, 249
biases
 accepting moderate amount of, 97
 anchoring, 95-97
 discriminatory bias, 47
 during testing, 230
- in few-shot prompting, 100
 LLM self-assessment and, 235
 logit bias, 58, 157
 SOMA assessment and, 235
 truth bias, 21
- binary decisions, 231
BM25 (best matching 25) algorithm, 108
book reviews, 93, 96
branch-solve-merge approach, 186
brevity, 138
- C**
- calibration, 157
camel case naming conventions, 179
canned conversations, 227
capitalization, 27
chain-of-thought prompting
 basis for, 42
 definition of term, 79
 length of preambles for, 148
 making models more thoughtful, 181-183
 models using by default, 210
 versus ReAct, 184
 scratchpad approach, 131
- chat models
 ChatML, 54
 versus completion models, 72
 drawbacks of, 169
- ChatGPT
 asked to describe LLMs, 4
 evolution of, 10
 generality of, 199
 models involved in creating, 47
 popularity of, 1
Chekhov's gun fallacy, 105
clarification, 90
classification, 155-157, 177, 231
cognitive biases, 95-97
Cohere, 161
comments and questions, xii
comparability, 101
completion models
 advice conversations, 128
 versus chat models, 72
 prompt templates for, 207
 usefulness of, 64
- completions (see also document types)
 anatomy of ideal completions, 147-153
 definition of term, 16

determining quality of, 154
enhancing with logprobs, 153-158
misleading or incorrect, 170 (see also hallucinations)
model selection, 159-165
patterns in, 72
quality of, 48
complexity, dimensions of, 67, 77
composite strings, 138
compression, 119
confidence, 154
consensual agents, 220
consistency, 90, 93
content (see also prompt content)
 extracting structured, 208
 focusing app on higher density, 158
 generating for email marketing, 206, 215
 up-to-the-moment information, 170
content policing, 128
context
 dimensions of, 103-105
 direct context, 76
 external context, 78
 finding dynamic content, 102
 including variable amounts of, 140
 indirect context, 76
 irrelevant context, 105
 large amounts of, 94
 removing static prompt context, 165
 retrieving, 76
 snippetizing context, 76
 for task-based interactions, 187-191
context window, 28, 94, 117, 123
continued pre-training, 163
contrastive A/B testing, 241
contrastive pre-training, 111
conversational agency
 building conversational agents, 191-196
 context for task-based interactions, 187-191
 definition of agency, 169
 full context of, 188
 limiting factors, 197
 reasoning, 181-187
 tool usage, 169-181
 versus workflow agency, 201-204
convolutional neural networks (CNNs), 246
copyrighted material, 50
core takeaways, 251
cosine similarity, 109
cost-effective models, 51, 159, 162
counterfactual situations, 22
course correction, 195
CrewAI, 220
cross entropy loss, 157
current exchanges, 187
cyclic graphs, 213

D

DAGs (directed acyclic graphs), 212
dangerous tools, 181, 196, 211
davinci-002 model, 47
decoders and encoders, 6
deep learning, 245
dependencies (of prompt elements), 142
deterministic tokenizers, 24
diffs, 164
dimensions of complexity, 67, 77
direct context, 76
direct feedback, 240
directed acyclic graphs (DAGs), 212
directed exploration, 226
discriminatory bias, 47
document completion, 18, 64
document types (see also completions)
 advice conversation, 127-130, 137
 analytic report, 130-132, 137
 structured documents, 133-136
downward (dumbward) LLM vision, 42
DSPy, 99, 217
dynamic content
 comparability and, 101
 definition of term, 89
 example of, 89
 finding, 102-105
 latency and, 100
 preparability and, 101
 purpose of, 100
 retrieval-augmented generation, 105-116
 strings from variable sources, 89
 summarization, 116-119

E

ease of use, 159
echo parameter, 157
edge cases, 97
editorial hints, 135
elastic prompt elements, 140
elastic snippets, 139

- Elasticsearch, 116
ellipsis (...), 140
email generation task, 206, 215
embedding models, 109-116, 246
empathy, 251
encoders and decoders, 6
end-of-text tokens, 29
environments, 48
errors (see also hallucinations)
 in LLM-based tasks, 210
 “straightforward first, errors later” pattern,
 98
 tool errors, 180
 typographical errors, 157
escape sequences, 135
euclidean distance, 109
evaluation (see also quality)
 of application quality, 82-83, 223
 challenges of and tips for, 81-83
 of completions, 154
 items assessed during, 224
 of workflows, 217
 (see also LLM workflows)
offline evaluation, 82, 225-238
online evaluation, 82, 239-243
selecting systems for, 243
of tasks in isolation, 211
test selection, 225
types of, 223
using suites of tests, 224
example suites
 advantages of, 226
 canned conversations, 227
 components of, 225
 definition of example, 226
 directed exploration and, 226
 finding examples, 228-230
 mocking conversations, 227
 versus test suites, 225
exp function, 153
explicit clarification, 90, 94
external context, 78
- F**
- FAISS library, 111
feedback, 217, 240 (see also reinforcement learning from human feedback)
feedforward pass, 75-81, 100, 224
few-shot prompting
- best uses for, 95, 100
definition of term, 91
drawbacks of, 94-100
formatting snippets, 139
learning implicit rules with, 93
shaping subtle expectations with, 93
versus zero-shot prompting, 92
fine-tuning, 16, 162-165
finish tool, 81, 184
floating-point inaccuracies, 158
fluff preambles, 149
format and style, 92
foundation models, 16
frameworks, 200, 220
freeform text format, 129
full fine-tuning, 163
function calls, structured output in, 209
function-calling models, 74
functional correctness, 240
functional testing, 233
functionality, 159
- G**
- general summaries, 119
generality, 199
generative pre-trained transformer models (see GPT models)
gold standard solutions, 231
Google, 161
GPT (generative pre-trained transformer) models
 base models, 45
 introduction of, 9-10
 transformer architecture, 37-43
GPT tokenizer, 24, 139
group chat managers, 220
- H**
- hallucinations
 argument hallucination, 180
 definition of term, 21
 goals for perfect assistants, 46
 inducing, 21
 keeping LLMs honest, 50
 preventing by providing background, 21
 preventing with model alignment, 47
 preventing with RAG, 78, 105
“happy path first, then unhappy path” order, 98
harness tests, 217, 224

hash sign (#), 69
HHH (helpful, honest, and harmless) alignment, 47
hidden knowledge, 169
hierarchical agents, 220
hierarchical summarization, 117-119
high urgency, 100
history, 4-10
HotpotQA dataset, 183
HTML-style comments, 135
Hugging Face, 28, 161
human thought, 19, 22-28, 211
hypothetical situations, 22

I
idiosyncratic behavior, 51
image processing, 245
impact, measurements of, 242
implicit clarification, 90, 93
importance (of prompt elements), 141
in-context learning, 125
inception approach, 128, 130
incidental metrics, 240, 242
incompatibilities (of prompt elements), 142
indirect context, 76
inertness, 138
instruct models, 52
instructions
 explicit clarification, 90, 94
 rules of thumb for, 91
integration, real-world, 128
intelligence
 artificial general intelligence, 199
 model selection and, 159
 upcoming developments in LLMs, 249-251
interactions
 multi-round, 128
 natural, 128
internal monologue, 182
introduction (prompt element), 124
intuition, 27
irrelevant context, 105
iterative reasoning and action, 183-185

J
Jaccard similarity, 107
jailbreaking, 46, 69
JSON
 as good choice for OpenAI, 136

defining and using tools, 171-175
property modifiers, 180
recognizable start and end, 151

K
Kaggle dataset, 93
knapsack problems, 144
knowledge distillation, 250

L
large language models (LLMs) (see also application design)
ability to recognize patterns, 91, 97-100
authors' discovery of, 2-4
auto-regressive models, 29-32
basic functioning of, 2, 5, 16-18, 63
book overview, x
core takeaways concerning, 251
determining realistic capabilities, 43
document completion, 18, 64
drawbacks of, 170, 199
history of, 4-10
human thought versus LLM processing, 19, 22-28
impact on software development, ix
impact on workflow, 1 (see also LLM workflows)
multimodal models, 245-251
potential drawbacks of, 45
potential uses for, 1
prerequisites to learning about, x
rapid adoption and expansion of, 245
sampling process, 32-37
trained for tool usage, 170-178
transformer architecture, 37-43
underlying principle of, x, 251
understanding LLM behavior, 251

latency
 acceptable, 148
 amount of context and, 190
 application evaluation, 242
 batch versus streaming workflows, 214
 dynamic content and, 100
 mitigating, 104
 model selection and, 162
 model size and, 73
 recording during testing, 224
 reducing network, 111

layers, 37

lexical retrieval, 107-108, 115
linear programming, 144
LiteLLM, 159
Little Red Riding Hood principle, 68, 71, 127, 135, 165, 176, 179, 251
LLaMA, 161
LLM frameworks, 200
LLM workflows
 advanced workflows, 217-221
 basic workflows, 204-217
 conversational versus workflow agency, 201-204
 definition of term, 212
 overview of, 200
LLM-application loops, 65-67, 75-81, 224
LLM-as-a-service companies, 161
LLM-as-judge, 210
logistic regression, 157
logit bias, 58, 157
logprobs (logarithm of the probabilities)
 classification and, 155-157
 converting to standard probability, 153
 definition of term, 33, 153
 determining completion quality with, 154
 investigating prompts with, 157
 purpose of, 58
 retrieving, 153
lookup tool, 81, 183
“loop” of interaction, 65-67, 75-81, 224
loss masking, 162
lost middle phenomenon, 125
low urgency, 100
low-rank adaptation (LoRA), 164
lowercase, avoiding in names, 179
Luigi, 212

M

machine learning (ML), 92, 155, 165, 199, 211, 221
make-believe prompts, 22
malicious applications, 9
Markdown
 benefits of for reports, 131
 new section marker (\n#), 152
 start and end structure, 151
markerless format, 129
Markov model, 5
masking, 40
math problems, 170, 182, 199

max_tokens parameter, 58
measurements of impact, 242
medium urgency, 100
metrics
 contrastive A/B testing, 241
 direct feedback, 241
 functional correctness, 242
 incidental metrics, 242
 measurements of impact, 242
 types of, 240
 user acceptance, 242
mind maps, 102
minibrains, 37-43, 95
miniprompts, 107
Mistral, 161
misunderstandings, 90
model alignment, 47
model domain, 68-73
model selection
 allowing for flexibility, 159
 balancing requirements, 160
 choosing a provider, 161
 fine-tuning models, 162-165
 for task-based interactions, 211
 guiding principles for, 159
 model size, 73, 162
model-as-a-service offerings, 28
modular development, 205
modularity, 138
multi-round interactions, 128
multilabel classification, 231
multimodality
 benefits of, 246
 image processing with, 246
 push toward multimodal models, 245
 upcoming developments in intelligence, 249-251
 user experience and user interface, 247-249
multistep problem solving, 183

N

n parameter, 58, 155
names and naming, 179
natural language
 assessing responses with LLMs, 234
 Markov model of, 5
 seq2seq architecture, 6
naturalness, 138
neural retrieval, 109-116

new section marker (\n#), 152
newline character, 124, 139, 180

0

objects of discourse, 247
offline evaluation
 advantages of, 239
 challenges of and tips for, 82
 definition of term, 223
 evaluating solutions, 231-235
 example suites, 225-227
 finding examples, 228-230
 SOMA assessment, 235-238
 tech tree of, 227
online evaluation
 A/B testing, 239
 definition of term, 223
 implicit indicators of quality, 82
 metrics, 240-243

Open LLM Leaderboard 2, 249
open-source models, 161
OpenAI Codex, 16
OpenAI GPT APIs
 best practices for arguments, 180
 chat completion API, 56-59
 chat models, 54-56
 comparing chat with completion, 59
 details of training, 9, 47, 69
 exponential increase in metrics, 10
 GPT-3, 47
 GPT-4, 245
 moving toward tools, 61
 tool definitions, 178

order
 “happy path first, then unhappy path” order, 98
 impact on prompt engineering, 43, 97
overfitting, 18

P

PaLM 540B model, 182
parallelism, 41
patterns and repetitions, 31, 97-100, 249
pause tokens, 182
persistence, 78
“The Pile” data set, 17
Pinecone.io, 111
pipelines, 212
placeholder values, 180

plan-and-solve prompting, 186
playwriting, 61
position (of prompt elements), 141
postscript (completion element), 152
PPO (proximal policy optimization), 49
pre-trained transformer architecture, 9 (see also ChatGPT)
pre-training process, 45, 69
preamble (completion element), 148-151
preamble (context source), 187
preparability, 101
prior conversations, 187, 190
prioritizing snippets, 77
private information, 170
probabilities, 32-37
probability distribution, 97
problem solving, 183
process_messages function, 172-175, 191
prompt assembly

 anatomy of ideal prompts, 123-127
 creating the final prompt, 143-146
 elastic snippets, 139
 formatting snippets, 137-139
 goals for successful, 77
 key to, 123
 relationship among prompt elements, 141-143
 selecting document type, 127-136
prompt content
 anatomy of ideal prompts, 123-127
 book recommendation example, 87, 96
 dynamic content, 100-119
 relationship among elements, 141-143
 role of prompts, 120
 rules for creating instructions, 91
 sources of content, 88
 static content, 89-100, 120

prompt engineering
 birth of, 10
 core lesson of, 251
 definition of term, 2, 11
 factors involved in successful, 189
 goals for successful, 11, 68-73
 impact of order on, 43, 97-100
 investigating prompts with logprobs, 157
 levels of sophistication, 11
 make-believe prompts, 22
 mastering, xi
 plan-and-solve prompting, 186

preventing hallucinations, 21
prompt length, 143
removing static prompt context, 165
role of prompt engineers, x
soft prompting, 165
templated prompts, 207
theatrical play metaphor, 61-63, 129
trends in LLMs, 251
prompt injection, 55
prompt internals, 179
prompt optimization, 99
prompt-crafting engines, 144
prompts, definition of, 2, 11, 16
prototyping, 162
providers, choosing, 161
proximal policy optimization (PPO), 49
proximity, 103
psychometric intelligence tests, 249
pull request (PR) summarization, 226

Q

quality (see also evaluation)
of applications, 81-83, 223
of completions, 48, 154
implicit indicators of, 83
quantization approaches, 251
questions and comments, xii

R

RAG (see retrieval-augmented generation)
ReAct, 183-185
real-world integration, 128
reasoning capabilities
branch-solve-merge approach, 186
chain-of-thought prompting, 181-183
iterative reasoning and action, 183-185
making models more thoughtful, 181
plan-and-solve prompting, 186
Reflexion, 186
reasoning depth, 79
reasoning preambles, 148
recognizable start and end (completion element), 151, 155
recommendation systems, 87
Red Riding Hood principle, 68, 71, 127, 135, 165, 176, 179, 251
Reflexion, 186, 210
refocus (prompt element), 125
regression tests, 224

reinforcement learning (RL), 48
reinforcement learning from human feedback (RLHF)
alignment tax, 51
avoiding idiosyncratic behavior, 51
build process, 47-50
cost-effectiveness of, 51
definition of term, 46
explicit instructions in, 91, 94
instruct models, 52
OpenAI GPT APIs, 56-59
preventing hallucinations, 50
theatrical play metaphor, 61, 129
well-known RLHF-trained chat models, 46
relevance, 106
repetitions and patterns, 31, 97-100, 249
report format, 130-132
requests, intercepting dangerous, 181, 196, 211
requirements, balancing, 160
response, 16
retrieval-augmented generation (RAG)
building RAG applications, 111-115
definition of term, 78, 105
lexical retrieval, 107-108
main focus of, 105
neural retrieval, 109
neural versus lexical retrieval, 115
reward model (RM), 48
RL (reinforcement learning), 48
RLHF model, 49 (see also reinforcement learning from human feedback)
rumor problem, 117
run_conversation function, 191-194

S

sandwich technique, 125
Scattergories, 25
scope section, 130
scoring snippets, 77
search tool, 81, 183
self-correction, 210
seq2seq (sequence to sequence) architecture, 5
sequence to sequence (seq2seq) architecture, 5
sequential agents, 220
set_room_temp function, 174
SFT (supervised fine-tuning) model, 48
Shopify plug-in marketing example, 201-217
side remarks, 137
similarity, 106

snippetizing context, 76
snippetizing documents, 110
snippets
 elastic snippets, 139
 formatting, 137-139
soft prompting, 165
soft search technique, 7
solutions
 evaluating, 231
 functional testing of, 233
 LLM assessment, 234
 matching the gold standard, 231
solvers, 186
SOMA assessment, 235-238
special requirements, 160
specific summaries, 119
speed, 159
spurious patterns, 97-100
stability, 104
start and end (preamble elements), 151, 155
state, 12, 78, 219
stated side remarks, 137
stateful objects of discourse, 247
static content
 clarifying your questions, 90
 definition of term, 89, 120
 example of, 89
 few-shot prompting, 91-100
 hardcoded blocks of text, 89
 removing, 165
 scoring, 102
stemming, 107
step size, 110
step-by-step thinking, 182
stop parameter, 58, 70
stop sequences, 152
stop words, 107
“straightforward first, errors later” pattern, 98
StrategyQA dataset, 182
stream parameter, 58
streaming modes, 152
streaming workflows, 214
strength, 199
stride size, 110
strings
 composite strings, 138
 from variable sources, 89
structural boilerplate preambles, 148
structured documents, 133-136
structured format, 129
structured outputs, in function calls, 209
style and format, 92
stylistic habits, 128
subtractive greedy approach, 145
summarization, 116-119
supervised fine-tuning (SFT) model, 48
Sydney, 90

T

table of contents, 131
target audience, ix
task-based interactions
 context sources, 187-189
 selecting and organizing context, 189-191
tasks
 adding sophistication to, 209
 adding variety to, 211
 definition of term, 205
 evaluating in isolation, 211
 generating arbitrary on the fly, 218
 implementing LLM-base tasks, 206-211
 interconnected nature of, 212
 specifying individual, 205
 stateful task agents, 219
temperature parameter, 34-37, 49, 58, 155
templated prompt task implementation, 207
templates, 207, 208
term frequency-inverse document frequency (TF-IDF), 108
tests (see also evaluation)
 A/B testing, 239
 contrastive A/B testing, 241
 functional testing, 233
 harness tests, 217, 224
 psychometric intelligence tests, 249
 regression tests, 224
 selecting, 225
 unit tests, 158, 186, 210, 224
 using suites of tests, 224
TextGrad, 217
“The Pile” data set, 17
the “loop” of interaction, 65-67, 75-81, 224
think-act-observe pattern, 184
thinking step, 185
thought vectors, 6
thumbs-up/thumbs-down, 241
tiktoken, 28
TL;DR, 79

token IDs, 138
tokenization
 counting tokens, 28
 deterministic tokenizers, 24
 drawbacks of, 24-28
 end-of-text tokens, 29
 GPT tokenizer, 24
 pre- or post-processing, 24
 process of, 23
 transformer architecture, 37-43
 understanding your model's tokenizer, 28
tool loop, 80
tool usage
 accessing hidden knowledge, 169
 dangerous tools, 181, 211
 evaluating right tool called with right syntax, 233
 guidelines for tool definitions, 178-181
 LLMs trained for, 170-178
 tool-based task implementation, 208
tool_choice parameter, 209
top_logprobs parameter, 58
training, 16-18, 69, 162-165, 249
transcript documents, 66
transcript format, 129
transformer, 7, 9, 37 (see also GPT models)
transformer architecture, 7, 37-43, 246
transitioning, 126
triple backticks ('``'), 69, 131, 151
truth bias, 21
TypeScript functions, 176
typographical errors, 157

U

 unidirectional transformers, 42
unit tests, 158, 186, 210, 224
up-to-the-moment information, 170
urgency, 100
user acceptance, 240
user experience, 195-196, 247-249
user proxies, 210, 220
user's problem domain, 67

V

validation errors, 180
Valley of Meh, 125
vector storage, 111
vision model, 246
vocabulary, 23

W

window sizes, 110 (see also context window)
workflows (see LLM workflows)

X

XML format, 135

Y

YAML format, 135, 151

Z

zero-shot prompts, 92

About the Authors

John Berryman is the founder and principal consultant of Arcturus Labs, where he specializes in LLM application development. His expertise helps businesses harness the power of advanced AI technologies. As an early engineer on GitHub Copilot, John contributed to the development of its completions and chat functionalities, working at the forefront of AI-assisted coding tools.

Before his work on Copilot, John built a varied career as a search engineer. His diverse experience includes helping to develop a next-generation search system for the US Patent Office, building search and recommendations for Eventbrite, and contributing to GitHub's code search infrastructure. John is also coauthor of *Relevant Search* (Manning), a book that distills his expertise in the field.

John's unique background, spanning both cutting-edge AI applications and foundational search technologies, positions him at the forefront of innovation in LLM applications and information retrieval.

Albert Ziegler has been designing AI-driven systems long before LLM applications became mainstream. As founding engineer for GitHub Copilot, he designed its prompt engineering system and helped inspire a wave of AI-powered tools and "Copilot" applications, shaping the future of developer assistance and LLM applications.

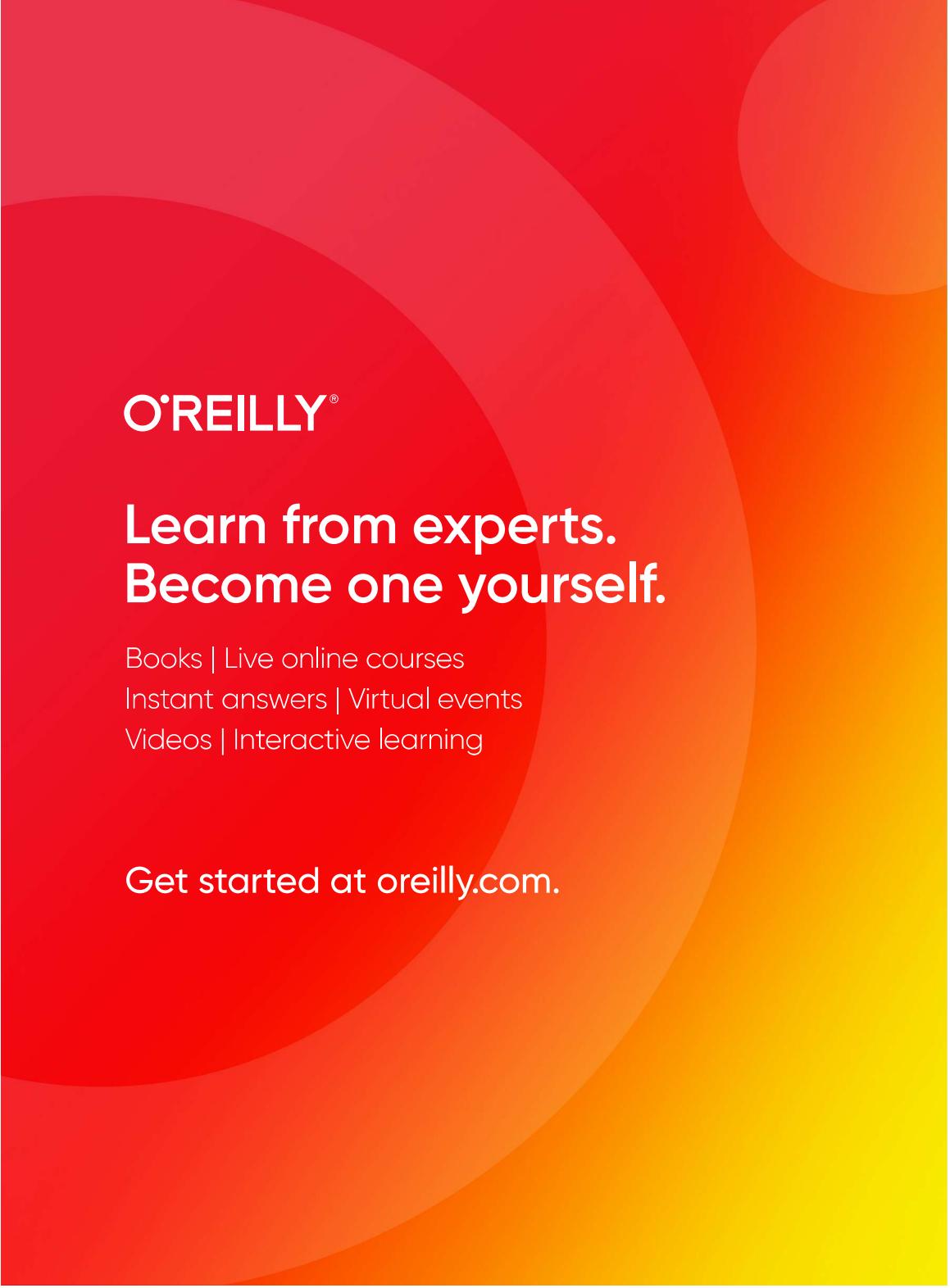
Today, Albert continues to push the boundaries of AI technology as Head of AI at XBOW, an AI cybersecurity company. There, he leads efforts blending large language models with cutting-edge security applications to secure the digital world of tomorrow.

Colophon

The animal on the cover of *Prompt Engineering for LLMs* is a banteng (*Bos javanicus*), a species of wild cattle in Southeast Asia. Banteng live in herds of one bull and many cows. They are brown or black with white stockings and light patches on their backsides. Both sexes have horns, but bulls are typically larger and darker. Adults banteng weigh up to 1,900 pounds and stand at 6 feet (hoof to shoulder).

The wild banteng is classified as Endangered on the IUCN Red List. Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Jose Marzan, based on an antique line engraving from *Meyers Kleines Lexicon*. The series design is by Edie Freedman, Ellie Volckhausen, and Karen Montgomery. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.



O'REILLY®

Learn from experts. Become one yourself.

Books | Live online courses
Instant answers | Virtual events
Videos | Interactive learning

Get started at oreilly.com.