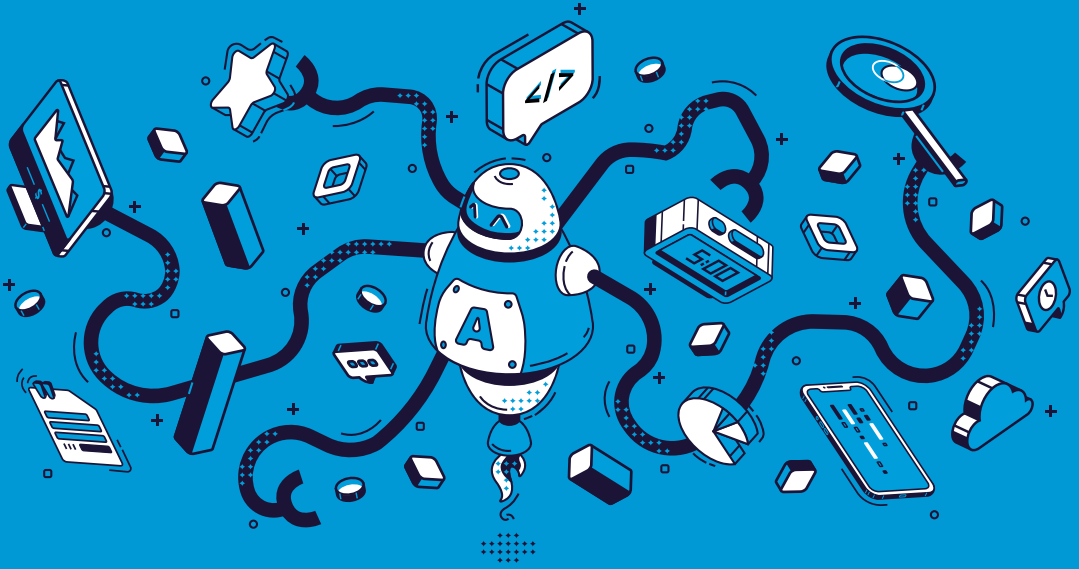


Foreword by **Patricia Thaine**, Co-Founder & CEO, Private AI



ChatGPT for Java

A Hands-on Developer's Guide
to ChatGPT and Open AI APIs

—
Bruce Hopkins

Apress®

ChatGPT for Java

A Hands-on Developer's Guide
to ChatGPT and Open AI APIs

Bruce Hopkins

Foreword by Patricia Thaine,
Co-Founder & CEO, Private AI

Apress®

ChatGPT for Java: A Hands-on Developer's Guide to ChatGPT and Open AI APIs

Bruce Hopkins
Beaverton, OR, USA

ISBN-13 (pbk): 979-8-8688-0115-0
<https://doi.org/10.1007/979-8-8688-0116-7>

ISBN-13 (electronic): 979-8-8688-0116-7

Copyright © 2024 by Bruce Hopkins

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Melissa Duffy
Development Editor: Laura Berendson
Editorial Assistant: Gryffin Winkler

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, Suite 4600, New York, NY 10004-1562, USA. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at <http://www.apress.com/bulk-sales>.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub. For more detailed information, please visit <https://www.apress.com/gp/services/source-code>.

Paper in this product is recyclable

Table of Contents

About the Author	ix
About the Technical Reviewer	xi
Foreword	xiii
Chapter 1: Introducing ChatGPT for Java Developers.....	1
Who Is This Book For?	1
Chapter Overview.....	2
Download the Code Now!.....	2
So, What Exactly Is ChatGPT and Why Do I Need to Use the OpenAI APIs?	2
Regex vs. ChatGPT: Fight!	5
Analysis Question #1: Who Didn't Get Any Ice Cream and Why?	7
Analysis Question #2: Which Kid Was Probably Left Sad?	9
Let's Unlearn Some Words in Order to Learn More About the ChatGPT API	10
Models. Models? Models!!!	10
When We Talk About Tokens, Think About the StringTokenizer and Not Access Tokens.....	15
Temperature Is All About Creativity.....	16
Getting Started with the OpenAI Playground	17
1. System.....	19
2. User	20
3. Assistant (Optional).....	20
4. Add Message (Optional).....	21
5. View Code (Optional).....	21

TABLE OF CONTENTS

6. Model (Optional)21

7. Temperature (Optional)22

8. Maximum Length (Optional)22

Try It Now! Experimenting with the “System” Role.....22

Conclusion23

Chapter 2: Using ChatGPT As Your Java Pair-Programmer25

 Creating Your First Java ChatGPT App: ListModels.java26

 List Models Endpoint.....26

 Creating the Request.....26

 Handling the JSON Response27

 Model (JSON).....27

 Chat Endpoint.....32

 Creating the Request.....32

 Chat (JSON)33

 Handling the Response.....42

 Chat Completion (JSON)43

 Wait, How Many Tokens Are in My Prompt?.....46

 ChatGPT Token Counter46

 Creating the Next Java App: ChatGPTClient.java.....48

 Conclusion56

**Chapter 3: Using AI in the Enterprise! Creating a Text Summarizer
 for Slack Messages57**

 So, What Is Prompt Engineering?.....58

 Updating ChatGPTClient.java (and Related Classes) with the Builder Pattern58

 ChatGPT Is Here to Take Away Everyone’s Jobs (Not Really)64

 Examining a Real World Problem: Customer Support for a Software Company64

Prompt Engineering 101: Text Summarization	68
Prompt #1: “tl;dr”	69
Prompt #2: “Explain This in 3 Sentences or Less”	71
Prompt #3: “I’m a Manager. Explain to Me What Happened”	72
Prompt #4: “Give Me Suggestions on Next Steps”	74
Let’s Talk About Real Prompt Engineering	77
Registering a Slack Bot App.....	77
Specifying What Your Bot Can (and Can’t) Do By Setting the Scope	80
Confirming Your Settings	82
Viewing the OAuth and Permissions Page.....	82
Installing Your Slack Bot App to Your Workspace	83
Getting Your Slack Bot (Access) Token	85
Inviting Your Bot to Your Channel.....	86
Finding the Channel ID of Your Channel.....	87
Using Your Slack Bot App to Automatically Grab Messages from a Channel	87
Setting Up Your Dependencies	87
Programmatically Reading Messages from Slack with ChannelReaderSlackBot.java.....	91
Exercises Left for the Reader.....	96
Conclusion	97
Chapter 4: Multimodal AI: Creating a Podcast Visualizer with Whisper and DALL-E 3	99
Introducing the Whisper Model by OpenAI.....	102
Features and Limitations of the Whisper Model.....	105
Transcriptions Endpoint	108
Creating the Request.....	108
Request Body (Multipart Form Data)	109

TABLE OF CONTENTS

Creating a Utility App to Split Audio Files: AudioSplitter.java 111

Creating the Audio Transcriber: WhisperClient.java 116

Having a Little Fun and Trying Things Out with a Podcast 122

Going Meta: Prompt Engineering GPT-4 to Write a Prompt for DALL-E..... 126

Create Image Endpoint..... 128

 Creating the Request..... 128

 Create Image (JSON) 129

 Handling the Response..... 131

 Image (JSON)..... 132

Creating the Image Generator: DALLEClient.java 132

DALL-E Prompt Engineering and Best Practices 136

 DALL-E Golden Rule #1: Get Familiar with the Types of Images that
 DALL-E Can Generate..... 137

 DALL-E Golden Rule #2: Be Descriptive with What You Want in the
 Foreground and Background..... 138

Conclusion 139

Exercises Left for the Reader 140

**Chapter 5: Creating an Automated Community Manager Bot with
Discord and Java..... 141**

 Choosing Discord as Your Community Platform..... 142

 Creating a More Advanced Bot Than Our Slack Bot 143

 Creating a More Advanced Bot Than Any Typical Discord Bot..... 143

 Understanding the Roles for the Bots..... 144

 Our Example Bank: Crook’s Bank..... 144

 First Things First: Create Your Own Discord Server 145

 Create the Q&A Channel..... 147

 Registering a New Discord Bot App with Discord 148

 Specifying General Info for the Bot 150

Specifying OAuth2 Parameters for the Bot	151
Invite Your Bot to Your Server.....	153
Getting the Discord ID Token for Your Bot and Setting the Gateway Intents	155
Creating a Q&A Bot App in Java to Answer Questions from a Channel	158
Setting Up Your Dependencies	159
Creating The First Discord Bot: TechSupportBotDumb.java.....	160
Loving the Lambda Expression to Simplify Code.....	164
Handling Messages Sent to the Discord Server	165
Success! Running Your First Discord Bot: TechSupportBotDumb.java.....	166
Streamlining the Process of Registering Our Next Discord Bot App with Discord.....	167
Registering a New Discord Bot App with Discord.....	167
Specifying General Info for the Bot.....	168
Specifying OAuth2 Parameters for the Bot.....	168
Invite Your Bot to Your Server	169
Getting the Discord ID Token for Your Bot and Setting the Gateway Intents.....	169
Creating the Next Discord Bot: ContentModeratorBotDumb.java	169
Handling Messages Sent to the Discord Server	173
Success Again! Running Your Second Discord Bot: ContentModeratorBotDumb.java.....	174
Conclusion	174
Exercises Left for the Reader.....	175
Chapter 6: Adding Intelligence to Our Discord Bots, Part 1: Using the Chat Endpoint for Q&A.....	177
Making TechSupportBot.java More Intelligent	178
Important Changes to Note from the Previous Version of the Tech Support Bot	185
Updates to the onMessageReceived() Method	186

TABLE OF CONTENTS

Analyzing ChatGPTClientForQAandModeration.java..... 186

 Using JSONPath in Order to Extract Content Quickly in JSON Files 192

 Running Our Intelligent Q&A Bot: TechSupportBot.java 192

We Have a Monumental Achievement... With One Slight Flaw 195

Update the System Message to ChatGPT and Let's Try Again..... 196

Conclusion 198

**Chapter 7: Adding Intelligence to Our Discord Bots, Part 2:
Using the Chat and Moderation Endpoints for
Moderation.....199**

Moderations Endpoint..... 201

 Creating the Request..... 202

 Create Moderation (JSON) 203

 Handling the JSON Response 203

 Moderation (JSON) 204

Creating Our Client for the Moderations Endpoint: ModerationClient.java..... 208

Making ContentModeratorBot.java More Intelligent 213

Important Changes to Note from the Previous Version of the Content
Moderator Bot 219

 Updates to the onMessageReceived() Method 220

Running Our Intelligent Content Moderator Bot: ContentModeratorBot.java..... 221

Conclusion 223

Exercises Left for the Reader 223

Appendix 1: List of OpenAI Models225

Index.....229

About the Author



Bruce Hopkins is a technical writer and world-renowned expert. He is both an Oracle Java Champion as well as an Intel Software Innovator. Bruce is also the author of the Apress book *Bluetooth for Java*.

About the Technical Reviewer



Van VanArsdale is a technology leader with over 30 years of experience in the software industry. He holds a B.S. in Computer Information Systems from the University of Massachusetts Lowell and an M.S. in CIS from Missouri State University. He has worked as a software engineer, architect, manager, and teacher. Van currently leads a team at a top financial services company and is an adjunct instructor at Missouri State University.

Foreword

Having had the pleasure of knowing Bruce for a few years now, I've witnessed his ability to take cutting-edge concepts and problems and make them accessible to programmers from a myriad of different backgrounds and abilities. Bruce's experience is particularly suitable for the task of breaking down many of the latest AI technologies, like ChatGPT, into core components that developers can use in their day-to-day problem-solving. He is the co-author of *Bluetooth for Java*, which came out just four years after Bluetooth made its way into the market, and he is the author of numerous technical guides written for major technology companies who needed their newest inventions to be adopted by developers.

As the co-founder and CEO of a Microsoft-backed AI company, called Private AI, I have the privilege of speaking with developers, managers, and C-level executives from organizations around the world not only about the implementation of responsible use of data but about the questions they have around generative AI, what kinds of problems to tackle, and where to start. Technology like ChatGPT is so new that the underlying model architecture allowing it to perform as well as it does – the Transformer architecture, that is – only came out in 2017 in the research paper titled “Attention Is All You Need.” As a result, a lot of the key questions that managers are struggling with are also questions developers are avidly diving into, including “What can I do with this technology?”, “How and where should I implement it?”, “Where should I start?”, “What are the pitfalls?”, and the ever-gnawing question when a lot of attention is placed on new inventions: “Is this all just hype?”

FOREWORD

Happily, many of these questions are addressed in concrete ways within this book, which is a crucial step toward cutting through to the core of what a new technology is actually useful for. In the same way that learning a programming language teaches you how to think in a completely different way than when using natural language, learning how to make the best use of AI makes you think in a completely different way than programming. In natural languages, the key is learning vocabulary, grammar, syntax, and semantics. In programming languages, it is logic, mathematics, syntax, scale, and significantly more of a first principles understanding of the world. In Artificial Intelligence, it's all about the data and how they relate to the task at hand. You can have the most powerful models in the world for generating text, for navigating roads, for generating art, but if you apply them to a task they were not meant for, prepare to be disappointed. However, if you understand fundamentally what they were built to do and use them accordingly, prepare to be amazed.

The concise, practical examples in this book will not only allow for you to quickly start building projects with ChatGPT, but will also start honing your intuition around how to think about this technology. Indeed, while the focus is working in Java, developers who use any programming language will benefit from diving in.

Patricia Thaine
Co-Founder and CEO, Private AI
www.private-ai.com

CHAPTER 1

Introducing ChatGPT for Java Developers

Who Is This Book For?

First of all, this book is for Java developers who don't have any training or experience in artificial intelligence, natural language processing, machine learning, or deep learning. You may have heard of the term “language model,” but I'm going to assume that it's NOT a term that you use every day.

Secondly, you might be familiar with (or have tried) ChatGPT, but you don't *quite* understand how everything works “under the hood” and you're not sure how to get started in order to use Java and ChatGPT programmatically together to “AI enable” your own applications and services.

Note Although ChatGPT is a household name, OpenAI, the company behind it, lacks broad recognition and isn't as widely recognized. So, although this book is about how to use ChatGPT programmatically within your Java apps, the APIs that we will be using are officially the OpenAI REST APIs. Therefore, we'll be using the term “ChatGPT API” and “OpenAI API” synonymously.

Chapter Overview

In this chapter, we're going to dive right in and explain a few terms that you might not be familiar with, and then go straight to the ChatGPT Playground. The Playground is essentially a web interface for developers that's built on top of ChatGPT REST API. So, once you're familiar with the Playground, the calls and responses from the ChatGPT API will feel like second nature to you.

Download the Code Now!



This book is filled with tons of examples and example code, so why bother typing everything yourself? Below is the URL to the GitHub repo to access the source code.

<https://github.com/Apress/ChatGPT-for-Java>

So, What Exactly Is ChatGPT and Why Do I Need to Use the OpenAI APIs?

Simply stated, ChatGPT is a group of **language models** (which are the result of training deep learning algorithms) developed by OpenAI that is capable of generating human-like text responses based on given **prompts**. So what's a prompt? A prompt is simply the input text that you give to ChatGPT. So let's kick things off by giving ChatGPT a prompt to ask how well it's familiar with Java design patterns (Listing 1-1).

Listing 1-1. PROMPT. A Simple Prompt with ChatGPT About Java Design Patterns

Describe 3 commonly used design patterns used by Java developers and why those patterns are important

If you've had the opportunity to work at a company using Java professionally, then you've probably received a question very similar to the prompt above in an interview. Listing 1-2 is ChatGPT's response.

Listing 1-2. RESPONSE. ChatGPT's Response to the Prompt in Listing 1-1

Certainly! Design patterns are reusable solutions to common problems that arise in software design. They are best practices that experienced developers have discovered and refined over time. Java developers frequently utilize design patterns due to the language's object-oriented nature. Here are three commonly used design patterns in Java:

1. Singleton Pattern:

- Purpose: This pattern ensures that a class has only one instance and provides a global point of access to that instance. It's particularly useful when you want to control access to shared resources, such as configuration managers or connection pools.

- Importance: The Singleton pattern is important because it provides a controlled mechanism to prevent creating multiple instances of a class, which can save memory and synchronize access to resources. Moreover, it ensures that operations using

the singleton are using the same instance and data, which is critical in scenarios where consistent state or coordinated actions are required.

2. Observer Pattern:

- Purpose: This pattern defines a one-to-many dependency between objects. When one object (known as the subject or observable) changes its state, all of its dependents (observers) are notified and updated automatically. It's the underlying principle behind the Model-View-Controller (MVC) architecture, where the model is the subject and the views are observers.

- Importance: The Observer pattern is vital for building systems that are modular and decoupled. Changes in one part of the system don't directly affect others, allowing for easier maintenance and scalability. This pattern also makes it easier to add new observers without modifying existing code, enhancing the system's extensibility.

3. Factory Pattern:

- Purpose: The Factory pattern provides an interface for creating instances of a class, but it's up to the derived subclasses to implement the method and determine which class to instantiate. It promotes the loose coupling by eliminating the need to bind application-specific classes into the code. Instead, the code interacts only with the resulting interface or abstract class, so the system can evolve without altering the calling code.

- Importance: Using the Factory pattern, a system can be more flexible and scalable. If there's a need to introduce a new product or modify an existing one, it can be done without disturbing the existing code. It encapsulates the object creation process and separates it from the main logic of

the program, promoting the Single Responsibility Principle. This ensures that a class is only responsible for one reason to change.

All these patterns, and many more, provide a vocabulary for developers, making it easier to communicate complex software designs and concepts. They embody solutions to recurring problems and can significantly speed up the development process, reduce errors, and improve code quality.

Not bad, huh? ChatGPT was not only able to provide a list of common design patterns used by Java developers, but also explain valid reasons on why they are important – all done in Natural Language.

So, at the end of the day, what does this mean for Java developers? Well, it's actually a big deal. That's because developers are now able to create applications that can have some type of "understanding" of text in Natural Language. Before the advent of AI tools like ChatGPT, developers typically had to create regular expressions (also called "regex") in order to perform rudimentary character and String pattern matching within a body of text. However, pattern matching and Natural Language Understanding are two totally different things.

Regex vs. ChatGPT: Fight!

Note If you already know that a regular expression can't perform Natural Language Understanding or sentiment analysis, then feel free to skip this section.

I'm thoroughly convinced that every programmer somewhere in their lifetime has met *some guy* who happens to be an expert in writing regular expressions. Regular expressions are great because they serve the purpose of being able to parse large amounts of text in order to find patterns within the text programmatically.

However, one of the biggest downsides to regular expressions is that once they have been written, they are extremely difficult to read (in my opinion, even by the developer who originally wrote it).

So let's see how regex holds up against ChatGPT, which has Natural Language Processing (NLP) and Natural Language Understanding (NLU) capabilities.

Listing 1-3 is a story of an impractically sad situation. However, it drives home the point that although regular expressions can be used to find words and phrases within a body of text, it can't be used to provide any type of NLU.

Listing 1-3. Sadstory.txt - A Sad Story About a Kid Who Didn't Eat Ice Cream

In the city of Buttersville, USA on Milkmaid street, there's a group of three friends: Marion Yogurt, Janelle de Queso, and Steve Cheeseworth III. On a hot summer's day, they heard the music from an ice cream truck, and decided to buy something to eat.

Marion likes strawberries, Janelle prefers chocolate, and Steve is lactose intolerant. That day, only two kids ate ice cream, and one of them bought a bottle of room-temperature water. The ice cream truck was fully stocked with the typical flavors of ice cream.

Analysis Question #1: Who Didn't Get Any Ice Cream and Why?

Now let's analyze this for a bit and ask some questions among ourselves. First of all, who didn't get any ice cream and why? The obvious answer is that Steve did not get any ice cream because of his lactose intolerance. However, since the story did not directly say that Steve did not buy ice cream, there's no way for a regular expression to match a text pattern in the story.

The regular expression could look for keywords such as "didn't have," "no ice cream," or the names of the kids. However, it would only be able to provide a response based on the presence of these patterns. For example, if the regular expression matches the pattern "didn't have" or "no ice cream" with Steve's name, it could show you the result of the text pattern. However, it certainly wouldn't be able to explain **why** Steve was the one who didn't have ice cream or provide any context-specific reasoning.

Now, let's provide the same story to ChatGPT and ask the question, "Who didn't get any ice cream?" Listing 1-4 takes our question and the preceding story and puts both together as a prompt.

Listing 1-4. PROMPT. The Sad Story Put into a Prompt

```
Using the information in the following story, who didn't get any ice cream and why?
```

```
###
```

```
In the city of Buttersville,USA on Milkmaid street, there's a group of three friends: Marion Yogurt, Janelle de Queso, and Steve Cheeseworth III. On a hot summer's day, they heard the music from an ice cream truck, and decided to buy something to eat.
```

Marion likes strawberries, Janelle prefers chocolate, and Steve is lactose intolerant. That day, only two kids ate ice cream, and one of them bought a bottle of room-temperature water. The ice cream truck was fully stocked with the typical flavors of ice cream.

Notice how it is a best practice to provide some type of separation, in this case, “###”, when creating a prompt (like the one above) where you have both instructions and data together. Later on, when we start using either the Playground or Java to programmatically invoke the ChatGPT API, you’ll see that there’s a much better way to provide that separation.

So, after sending the prompt, ChatGPT will provide you an answer as shown in Listing 1-5.

Listing 1-5. RESPONSE. ChatGPT’s Answer to Analysis Question #1



Based on the given information, Steve is lactose intolerant and therefore cannot eat ice cream. Therefore, Steve is the one who didn't get any ice cream.

As you can see, ChatGPT can utilize NLP and NLU, and therefore it can *understand* (artificially intelligently, that is) the context of the scenario. It can interpret the relationships between the kids, their preferences, and Steve’s lactose intolerance. It is able to understand that the last names of the children, the street name, and the city name are the names of dairy products, but are obviously irrelevant to the question at hand.

Analysis Question #2: Which Kid Was Probably Left Sad?

Now, to further prove the point that a regular expression is unable to provide any type of NLP or NLU, now let's use a new term called **sentiment analysis**. Therefore, after the ice cream truck drove away, which kid was left sad?

Since the story had no mention of any of the children's feelings or emotions, there is no text pattern that would allow any regular expression to return a match.

However, if you posed the same question to ChatGPT, it will return a response as shown in Listing 1-6.

Listing 1-6. RESPONSE. ChatGPT's Answer to Analysis Question #2



Since Steve is lactose intolerant and cannot eat ice cream, he would be the kid left sad because he couldn't enjoy the ice cream like Marion and Janelle.

Therefore, ChatGPT is able to comprehend the scenario, reason through the information, and provide a correct answer along with an *explanation* for that answer.

Let's Unlearn Some Words in Order to Learn More About the ChatGPT API

First of all, before you get started working with the ChatGPT and OpenAI APIs, there are few words and terms that you should be familiar with first; otherwise, things won't exactly make sense. So let's make sure that we're all clear on the definition of Models, Prompts, Tokens, and Temperature when using ChatGPT programmatically.

Models. Models? Models!!!

As a Java developer, when you hear the term “model,” you may immediately think of object-oriented programming and the representation of real-world entities in your Java class, right? For example, think of the term, “object model.” Additionally, if you're ever worked with any type of database before, then the term “model” may ALSO conjure into your mind the idea of the representation of data and their relationships in your database. For example, think of the term, “data model.”

However, when working with the ChatGPT APIs (and artificial intelligence in general, for that matter), you need to forget both of those definitions, because they don't apply. In the realm of artificial intelligence, a “model” is a pre-trained **neural network**.

Remember, as I mentioned earlier, you won't need a PhD in Machine Learning in order to read this book. So what's a neural network? Simply stated, a neural network is a fundamental component of artificial intelligence systems, because they are designed to simulate the way the human brain works by using interconnected layers of artificial neurons to process and analyze data. These networks can be trained on vast amounts of data to learn patterns, relationships, and to make predictions.

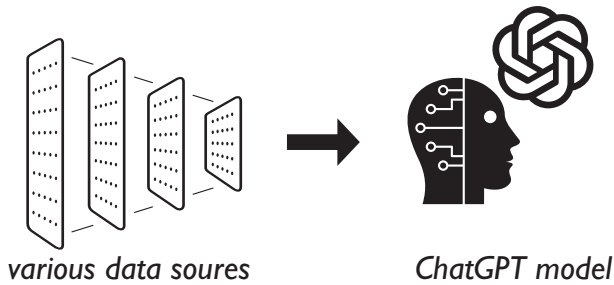


Figure 1-1. An AI Model Is Trained on Vast Amounts of Data

In the context of AI, a “pre-trained model” refers to a neural network that has been trained on a specific task or dataset before being made available for use by developers. This training process involves exposing the model to large amounts of labeled and categorized (also called, “annotated”) data and adjusting its internal parameters to optimize its performance on the given task.

Let’s look at some of the models provided by OpenAI for developers to use to AI-enable their applications.

GPT-4 GPT-4 is the latest generation of OpenAI’s GPT set of models. GPT stands for Generative Pre-trained Transformer, and these models have been trained to understand natural language as well as multiple programming languages. The GPT-4 models take text and images as inputs as the prompt and provide text as the output.

Some of the GPT-4 models available are

- gpt-4
- gpt-4-32k
- gpt-4-vision

(continued)