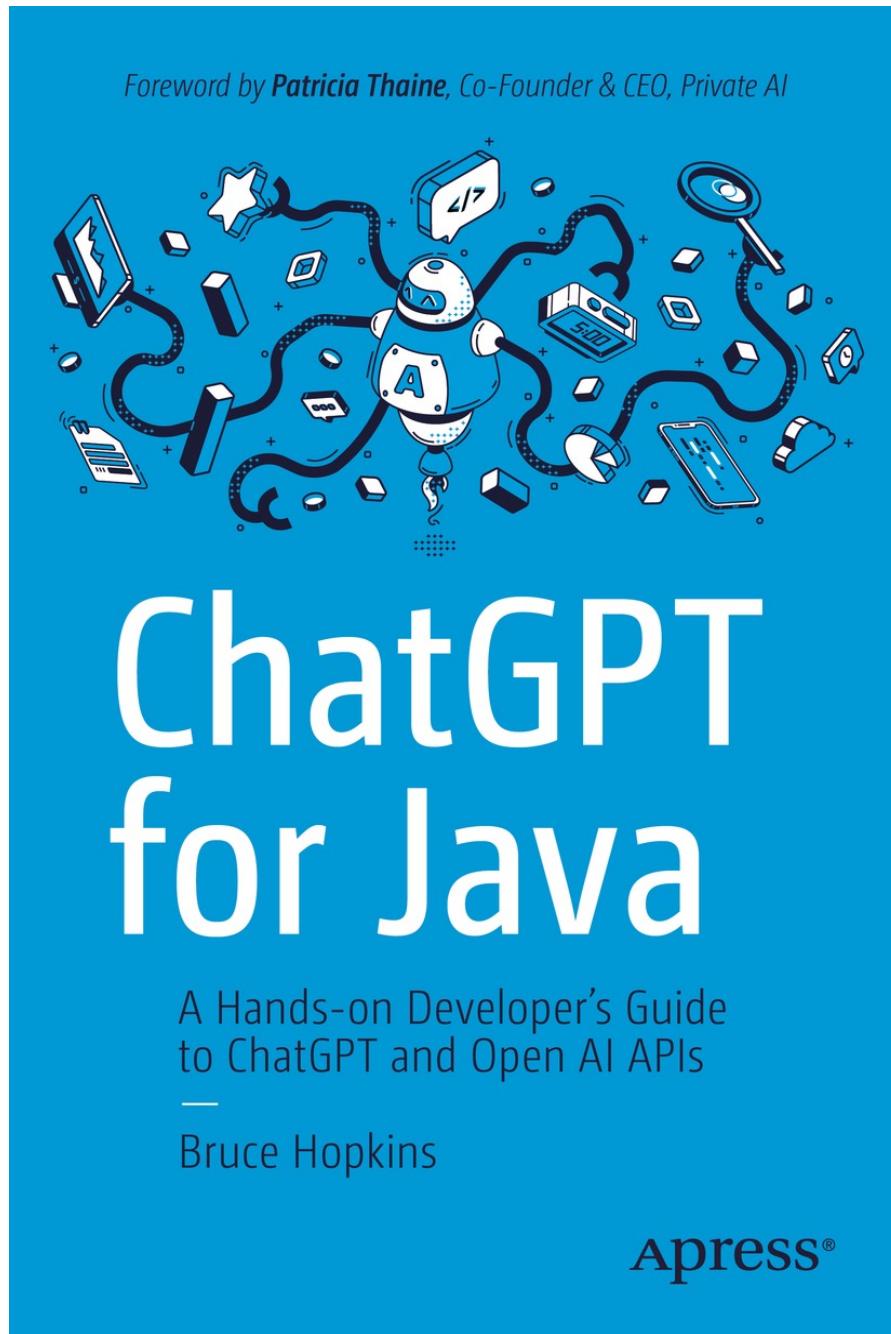


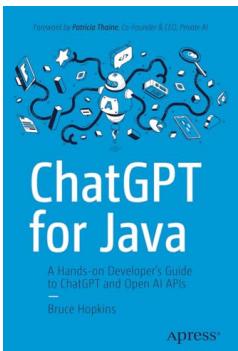
ChatGPT for Java: A Hands-on Developer's Guide to ChatGPT and Open AI APIs

Bruce Hopkins

Visit to download the full and correct content document:
<https://ebookmass.com/product/chatgpt-for-java-a-hands-on-developers-guide-to-chatgpt-and-open-ai-apis-bruce-hopkins/>

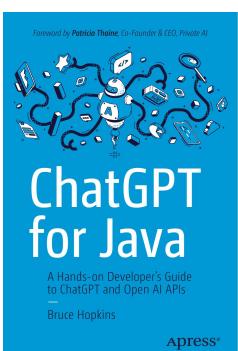


More products digital (pdf, epub, mobi) instant download maybe you interests ...



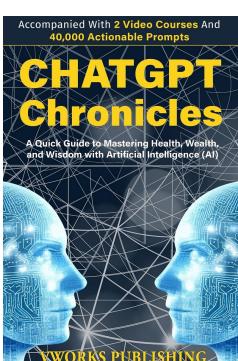
ChatGPT for Java: A Hands-on Developer's Guide to ChatGPT and Open AI APIs 1st Edition Bruce Hopkins

<https://ebookmass.com/product/chatgpt-for-java-a-hands-on-developers-guide-to-chatgpt-and-open-ai-apis-1st-edition-bruce-hopkins/>



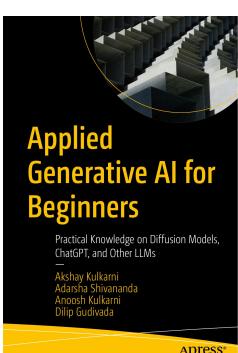
ChatGPT for Java: A Hands-on Developer's Guide to ChatGPT and Open AI APIs 1st Edition Bruce Hopkins

<https://ebookmass.com/product/chatgpt-for-java-a-hands-on-developers-guide-to-chatgpt-and-open-ai-apis-1st-edition-bruce-hopkins-2/>



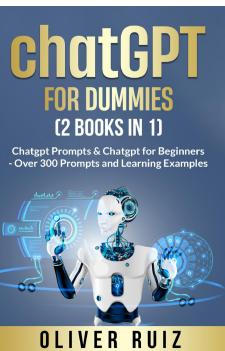
ChatGPT Chronicles: A Quick Guide to Mastering Health, Wealth, and Wisdom with Artificial Intelligence (AI) William Leeson

<https://ebookmass.com/product/chatgpt-chronicles-a-quick-guide-to-mastering-health-wealth-and-wisdom-with-artificial-intelligence-ai-william-leeson/>



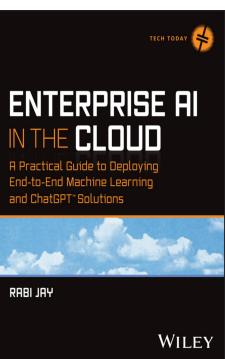
Applied Generative AI for Beginners: Practical Knowledge on Diffusion Models, ChatGPT, and Other LLMs Akshay Kulkarni

<https://ebookmass.com/product/applied-generative-ai-for-beginners-practical-knowledge-on-diffusion-models-chatgpt-and-other-langs-akshay-kulkarni/>



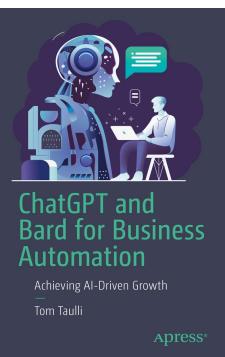
ChatGPT For Dummies (2 Books in 1): Chatgpt Prompts & Chatgpt for Beginners - Over 300 Prompts and Learning Example Oliver Ruiz

<https://ebookmass.com/product/chatgpt-for-dummies-2-books-in-1-chatgpt-prompts-chatgpt-for-beginners-over-300-prompts-and-learning-example-oliver-ruiz/>



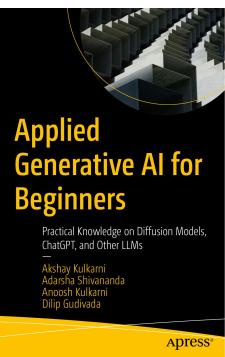
Enterprise AI in the Cloud: A Practical Guide to Deploying End-to-End Machine Learning and ChatGPT™ Solutions Rabi Jay

<https://ebookmass.com/product/enterprise-ai-in-the-cloud-a-practical-guide-to-deploying-end-to-end-machine-learning-and-chatgpt-solutions-rabi-jay/>



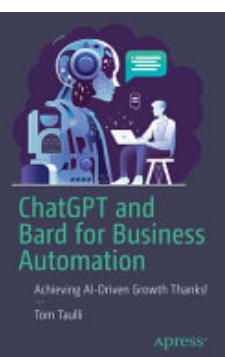
ChatGPT and Bard for Business Automation: Achieving AI-Driven Growth 1st Edition Taulli

<https://ebookmass.com/product/chatgpt-and-bard-for-business-automation-achieving-ai-driven-growth-1st-edition-taulli/>



Applied Generative AI for Beginners: Practical Knowledge on Diffusion Models, ChatGPT, and Other LLMs 1st Edition Akshay Kulkarni

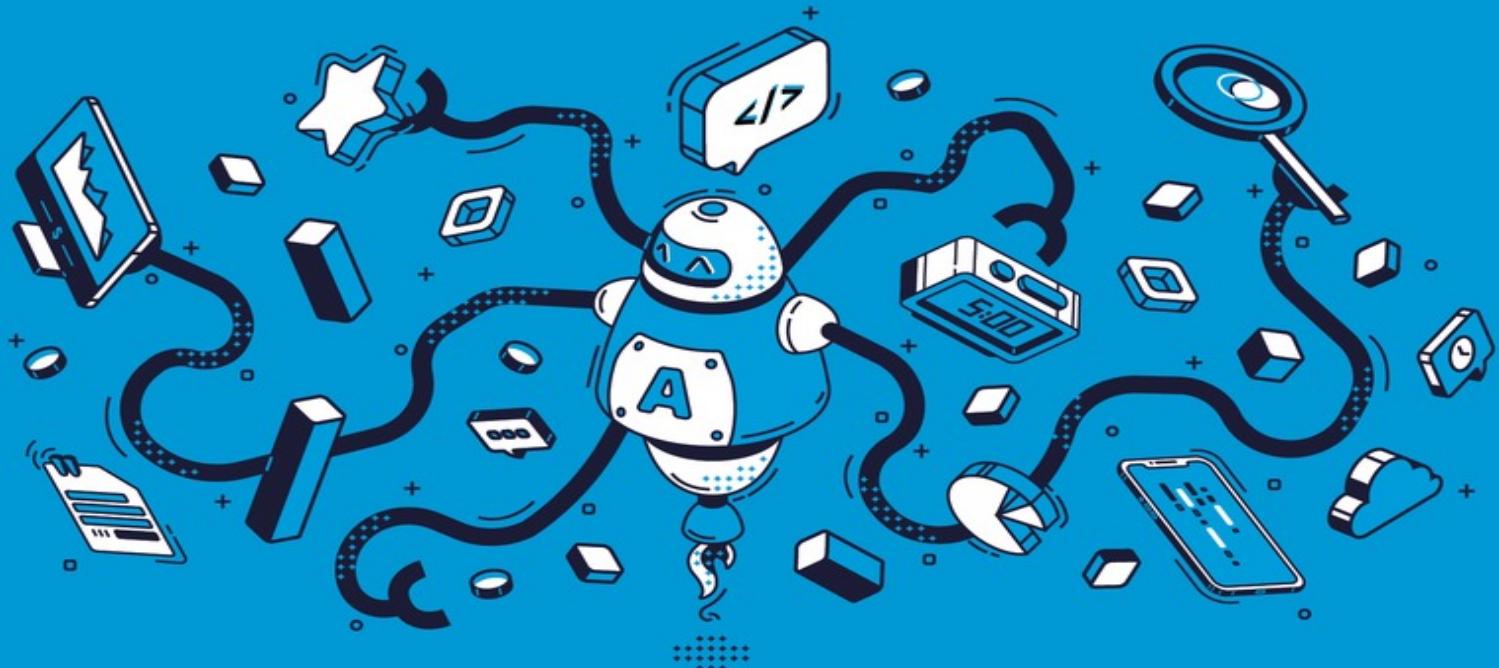
<https://ebookmass.com/product/applied-generative-ai-for-beginners-practical-knowledge-on-diffusion-models-chatgpt-and-other-langs-1st-edition-akshay-kulkarni/>



ChatGPT and Bard for Business Automation: Achieving AI-Driven Growth 1st Edition Tom Taulli

<https://ebookmass.com/product/chatgpt-and-bard-for-business-automation-achieving-ai-driven-growth-1st-edition-tom-taulli/>

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®

Bruce Hopkins

ChatGPT for Java

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

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

Apress®

Bruce Hopkins
Beaverton, OR, USA

ISBN 979-8-8688-0115-0 e-ISBN 979-8-8688-0116-7

<https://doi.org/10.1007/979-8-8688-0116-7>

© Bruce Hopkins 2024

This work is subject to copyright. All rights are solely and exclusively licensed 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.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Apress imprint is published by the registered company APress Media, LLC, part of Springer Nature.

The registered company address is: 1 New York Plaza, New York, NY
10004, U.S.A.

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?”

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

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>.

Table of Contents

Chapter 1: Introducing ChatGPT for Java Developers

[Who Is This Book For?](#)

[Chapter Overview](#)

[Download the Code Now!](#)

[So, What Exactly Is ChatGPT and Why Do I Need to Use the OpenAI APIs?](#)

[Regex vs. ChatGPT: Fight!](#)

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

[Analysis Question #2: Which Kid Was Probably Left Sad?](#)

[Let's Unlearn Some Words in Order to Learn More About the ChatGPT API](#)

[Models. Models? Models!!!](#)

[When We Talk About Tokens, Think About the StringTokenizer and Not Access Tokens](#)

[Temperature Is All About Creativity](#)

[Getting Started with the OpenAI Playground](#)

[1. System](#)

[2. User](#)

[3. Assistant \(Optional\)](#)

[4. Add Message \(Optional\)](#)

[5. View Code \(Optional\)](#)

[6. Model \(Optional\)](#)

[7. Temperature \(Optional\)](#)

[8. Maximum Length \(Optional\)](#)

[Try It Now! Experimenting with the "System" Role](#)

Conclusion

Chapter 2: Using ChatGPT As Your Java Pair-Programmer

Creating Your First Java ChatGPT App: ListModels.java

List Models Endpoint

Creating the Request

Handling the JSON Response

Model (JSON)

Chat Endpoint

Creating the Request

Chat (JSON)

Handling the Response

Chat Completion (JSON)

Wait, How Many Tokens Are in My Prompt?

ChatGPT Token Counter

Creating the Next Java App: ChatGPTClient.java

Conclusion

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

So, What Is Prompt Engineering?

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

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

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

Prompt Engineering 101: Text Summarization

Prompt #1: "tl;dr"

Prompt #2: "Explain This in 3 Sentences or Less"

Prompt #3: “I’m a Manager. Explain to Me What Happened”

Prompt #4: “Give Me Suggestions on Next Steps”

Let’s Talk About Real Prompt Engineering

Registering a Slack Bot App

Specifying What Your Bot Can (and Can’t) Do By Setting the Scope

Confirming Your Settings

Viewing the OAuth and Permissions Page

Installing Your Slack Bot App to Your Workspace

Getting Your Slack Bot (Access) Token

Inviting Your Bot to Your Channel

Finding the Channel ID of Your Channel

Using Your Slack Bot App to Automatically Grab Messages from a Channel

Setting Up Your Dependencies

Programmatically Reading Messages from Slack with ChannelReaderSlackBot.java

Exercises Left for the Reader

Conclusion

Chapter 4: Multimodal AI: Creating a Podcast Visualizer with Whisper and DALL·E 3

Introducing the Whisper Model by OpenAI

Features and Limitations of the Whisper Model

Transcriptions Endpoint

Creating the Request

Request Body (Multipart Form Data)

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

Creating the Audio Transcriber: WhisperClient.java

Having a Little Fun and Trying Things Out with a Podcast

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

Create Image Endpoint

Creating the Request

Create Image (JSON)

Handling the Response

Image (JSON)

Creating the Image Generator: DALLEClient.java

DALL·E Prompt Engineering and Best Practices

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

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

Conclusion

Exercises Left for the Reader

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

Choosing Discord as Your Community Platform

Creating a More Advanced Bot Than Our Slack Bot

Creating a More Advanced Bot Than Any Typical Discord Bot

Understanding the Roles for the Bots

Our Example Bank: Crook's Bank

First Things First: Create Your Own Discord Server

Create the Q&A Channel

Registering a New Discord Bot App with Discord

- [Specifying General Info for the Bot](#)
- [Specifying OAuth2 Parameters for the Bot](#)
- [Invite Your Bot to Your Server](#)
- [Getting the Discord ID Token for Your Bot and Setting the Gateway Intents](#)
- [Creating a Q&A Bot App in Java to Answer Questions from a Channel](#)
 - [Setting Up Your Dependencies](#)
- [Creating The First Discord Bot: TechSupportBotDumb.java](#)
 - [Loving the Lambda Expression to Simplify Code](#)
 - [Handling Messages Sent to the Discord Server](#)
 - [Success! Running Your First Discord Bot: TechSupportBotDumb.java](#)
- [Streamlining the Process of Registering Our Next Discord Bot App with Discord](#)
 - [Registering a New Discord Bot App with Discord](#)
 - [Specifying General Info for the Bot](#)
 - [Specifying OAuth2 Parameters for the Bot](#)
 - [Invite Your Bot to Your Server](#)
 - [Getting the Discord ID Token for Your Bot and Setting the Gateway Intents](#)
- [Creating the Next Discord Bot: ContentModeratorBotDumb.java](#)
 - [Handling Messages Sent to the Discord Server](#)
 - [Success Again! Running Your Second Discord Bot: ContentModeratorBotDumb.java](#)
- [Conclusion](#)
- [Exercises Left for the Reader](#)

Chapter 6: Adding Intelligence to Our Discord Bots, Part 1: Using the Chat Endpoint for Q&A

Making TechSupportBot.java More Intelligent

Important Changes to Note from the Previous Version of the Tech Support Bot

Updates to the onMessageReceived() Method

Analyzing ChatGPTClientForQAandModeration.java

Using JSONPath in Order to Extract Content Quickly in JSON Files

Running Our Intelligent Q&A Bot: TechSupportBot.java

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

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

Conclusion

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

Moderations Endpoint

Creating the Request

Create Moderation (JSON)

Handling the JSON Response

Moderation (JSON)

Creating Our Client for the Moderations Endpoint: ModerationClient.java

Making ContentModeratorBot.java More Intelligent

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

Updates to the onMessageReceived() Method

Running Our Intelligent Content Moderator Bot: ContentModeratorBot.java

Conclusion

Exercises Left for the Reader

Appendix 1: List of OpenAI Models

Index

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.



1. Introducing ChatGPT for Java Developers

Bruce Hopkins¹✉
(1) Beaverton, OR, USA

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).

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

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

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.



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.

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

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.

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.

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

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.

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

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.



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.

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

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.



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.

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

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.

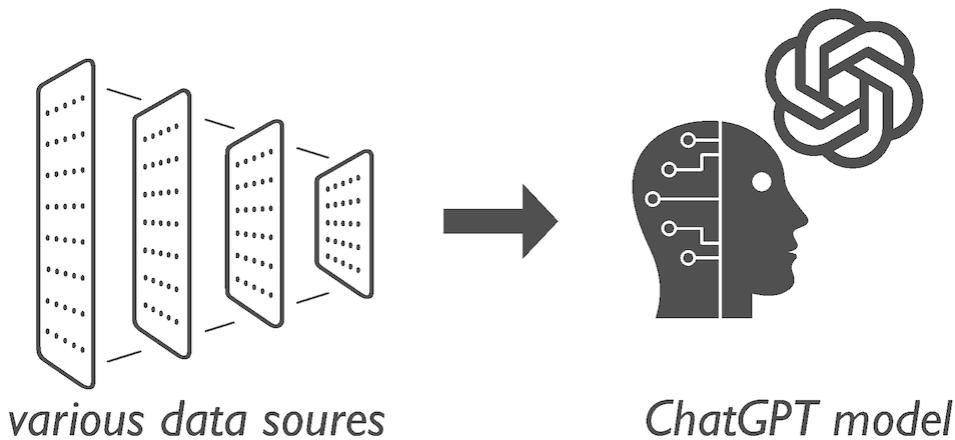


Figure 1-1 An AI Model Is Trained on Vast Amounts of Data

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	<p>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.</p> <p>Some of the GPT-4 models available are</p> <ul style="list-style-type: none">• gpt-4• gpt-4-32k• gpt-4-vision
GPT-3.5	<p>GPT-3.x is the previous generation of OpenAI’s GPT set of models. The original ChatGPT released to the public in November 2022 used GPT 3.</p> <p>Some of the GPT-3 models available are</p> <ul style="list-style-type: none">• gpt-3.5-turbo• gpt-3.5-turbo-16k
DALL·E	<p>The DALL·E model can generate and edit images given a natural language prompt.</p> <p>Later in this book in Chapter 4, we’re going to have some fun with the DALL·E model to visualize the conversation of what is being discussed in your favorite podcast episode.</p> <p>Some of the DALL·E models available are</p> <ul style="list-style-type: none">• dall-e-3• dall-e-2
TTS	<p>The TTS model takes text and converts it to audio with surprisingly good results. In most cases, the audio is almost indistinguishable from a human voice.</p> <p>Some of the TTS models available are</p> <ul style="list-style-type: none">• tts-1• tts-1-hd
Whisper	<p>Simply stated, the Whisper model converts audio into text.</p>

	<p>In this book, we're going to use the Whisper model to search for text in a podcast episode.</p>
Embeddings	<p>The Embeddings model can convert large amounts of text into a numerical representation of how the Strings in the text are related. So how is that useful? Embeddings allows developers to do specific tasks using custom datasets. Yes, this means that you can train the embeddings model on specific data that is relevant to your application. This allows you to do operations such as</p> <ul style="list-style-type: none"> • Searching within your own body of text • Clustering data so that Strings of text are grouped by their similarity • Getting Recommendations (where items with related text strings are recommended) • Detecting Anomalies (where outliers with little relatedness are identified) • Measuring Diversity (where similarity distributions are analyzed) • Classifying Data (where text strings are classified by their most similar label)
Moderation	<p>The moderation models are fine-tuned models that can detect whether text may be sensitive or unsafe. These models can analyze text content and classify it according to the following categories:</p> <ul style="list-style-type: none"> • Hate • Hate/threatening • Harassment • Harassment/threatening • Self-harm • Self-harm/intent • Self-harm/instructions • Sexual • Sexual/minors • Violence • Violence/graphic <p>The moderation models available are</p> <ul style="list-style-type: none"> • Text-moderation-latest • Text-moderation-stable

Legacy and Deprecated	Since the debut of ChatGPT, OpenAI has continued to support their older AI models, but they have been labeled as “legacy” or “deprecated” models. These models continue to exist; however they have released other models that are more accurate, faster, and cheaper to use.
-----------------------	---

Note This is by no means an exhaustive list of models available for developers provided by OpenAI! As newer models are released, the older models will be marked as legacy or deprecated. Therefore, it's important to stay up to date by checking the list of available models on the OpenAI documentation list of models:<https://platform.openai.com/docs/models>

When We Talk About Tokens, Think About the StringTokenizer and Not Access Tokens

When using a third-party API, you might think of a token in the same sense as an access token, which is typically a UUID that allows you to identify yourself and maintain a session with the service. Well, forget that definition for now.

Now, as a Java developer, you've probably had the opportunity to use the class, `java.util.StringTokenizer`, in order to take a String and split it up into an Array of smaller Strings so that you can iterate over it for whatever purpose you need. For example, if you had a paragraph of text, you could let your delimiter be “.” in order to get an Array of sentences in the paragraph.

The good news is that the OpenAI API concept of a **token** is very familiar to the Java concept in the sense that it is a fragment of text. For the OpenAI APIs, a token is a chunk of a text that is approximately 4 characters long. That's it – nothing else special.

So if a token is approximately a 4-character chunk of text, then why do we care about it?

When working with the OpenAI textual models, developers need to be aware of token limitations, because they impact the cost and performance of API calls. For example, the gpt-3.5-turbo model has a token limitation of 4096 tokens, while the gpt-4-vision model has a

limitation of 128,000 tokens (which is approximately the size of a 300-page novel). Token limitations for models are called **context windows**.

As a result, developers need to take into account the length of the prompts as inputs and outputs to the models, ensuring that they fit within the model's token constraints.

Table 1-1 provides a list of some of the most current models with the token limitations and their pricing.

Table 1-1 List of Models with Their Token Limitations and the Cost Per Token

Model	Max Tokens	Cost of Token Input	Cost of Token Output
gpt-4	8,192	\$0.03 / 1K tokens	\$0.06 / 1K tokens
gpt-4-32k	32,768	\$0.06 / 1K tokens	\$0.12 / 1K tokens
gpt-4-vision	128,000	\$0.01 / 1K tokens	\$0.03 / 1K tokens
gpt-3.5-turbo-instruct	4,096	\$0.0015 / 1K tokens	\$0.002 / 1K tokens
gpt-3.5-turbo-16k	16,384	\$0.0010 / 1K tokens	\$0.002 / 1K tokens
text-embedding-ada-002	8192	\$0.0001 / 1K tokens	

Temperature Is All About Creativity

Of course, ChatGPT isn't sentient, so it's incapable of thinking as we humans do. However, by adjusting the **temperature** setting in your prompts to the ChatGPT API, you can enable the responses to be more creative. But understanding what *it* understands is crucial if you want to make best use of its potential.

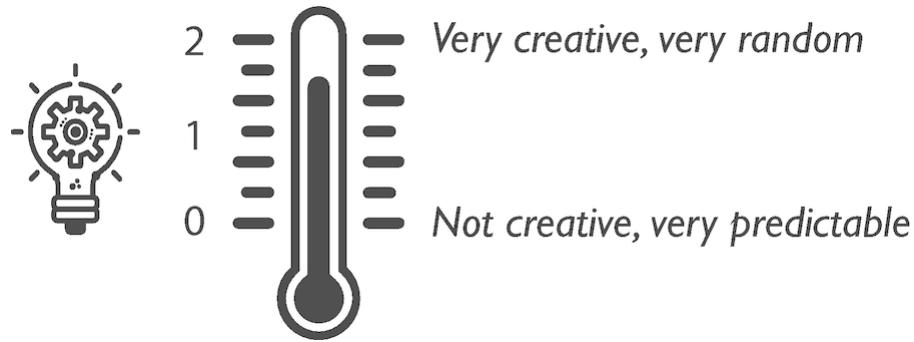


Figure 1-2 Modify the Temperature in Order to Get More (or Less) Creative Responses

Getting Started with the OpenAI Playground

Now it's time to take the concepts that we've learned so far and start to put them to good use! However, we need to do first things first, and therefore, you will need to have a developer account with OpenAI and create an API key.

Head over to the following URL to create your dev account and API key:

<https://platform.openai.com/account/api-keys>

As you can see from the image in Figure 1-3, you can name your API key anything that you want.

The screenshot shows the OpenAI Platform interface. At the top, there are navigation links: Overview, Documentation, API reference, Examples, and Playground. On the right, there are upgrade, help, and personal account options. The main area is titled 'API keys'. It says, 'Your secret API keys are listed below. Please note that we do not display your secret API keys again after you generate them.' A modal window is open, titled 'Create new secret key'. It has a 'Name' field labeled 'Optional' with the value 'developer key for Java apps'. Below the field are 'Cancel' and 'Create secret key' buttons. At the bottom of the page, there is a note: 'If you belong to multiple organizations, this setting controls which organization is used by default when making requests with the API keys above.' A dropdown menu shows 'Personal'. A note at the bottom states: 'Note: You can also specify which organization to use for each API request. See [Authentication](#) to learn more.'

Figure 1-3 Before You Can Access the Playground or Make API Calls, You Need to Have an API Key

You should be aware that as a requirement to create an API key, you will need to provide to OpenAI a credit card so that you can be billed for usage of their models.

Now that you've got your API key, let's go straight to the Chat Playground at the following URL:

<https://platform.openai.com/playground>

Upon entering the Playground, click the combobox at the top and select Chat the option to end the Chat Playground, as shown in Figure 1-4.

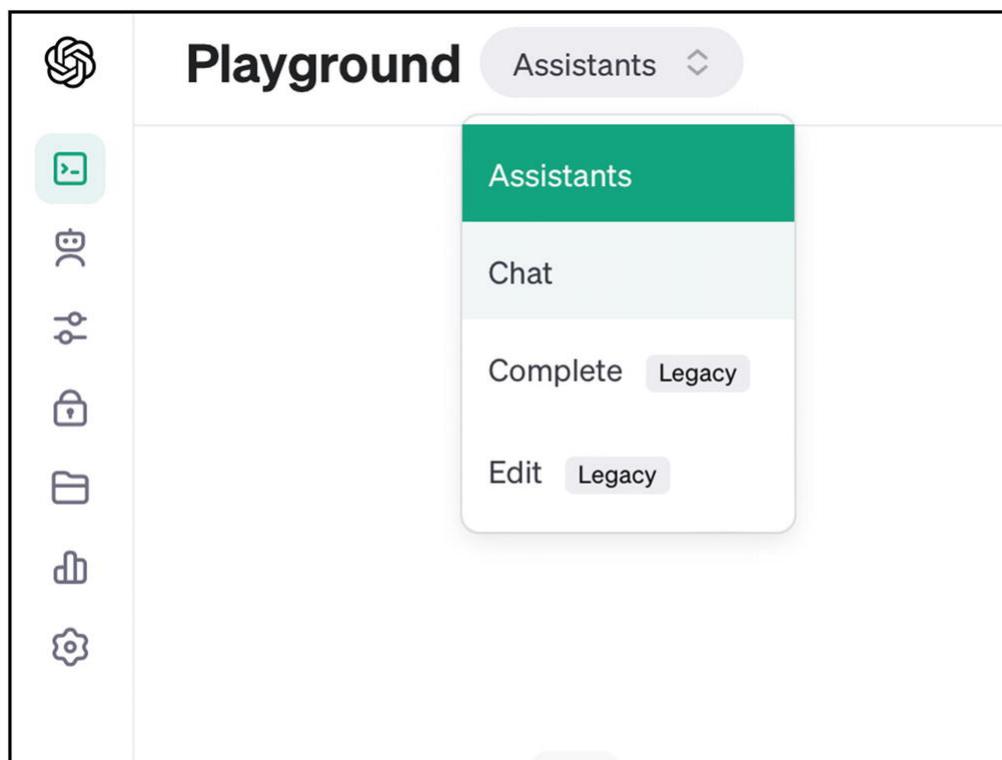


Figure 1-4 After Entering the Playground, Select the Chat Option

Figure 1-5 depicts the Chat Playground, with certain parts numbered so that they can be easily identified.

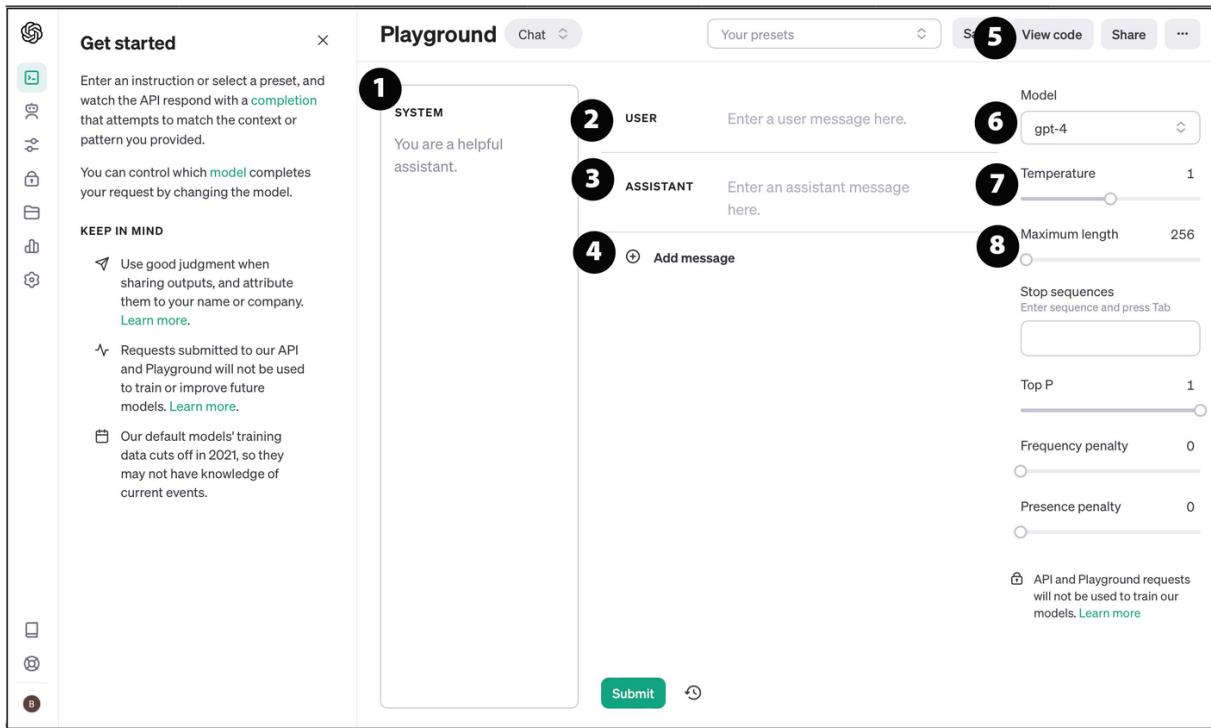


Figure 1-5 The Chat Playground Can Be a Little Daunting at First Glance

1. System

As you can see, the user interface for the Chat Playground is vastly more complex than the ChatGPT website that everyone else uses. So let's talk about the **System** field (see Figure 1-5, item 1).

In my opinion, ChatGPT can be described as “a vastly powerful form of artificial intelligence... with amnesia.” Therefore, when you’re using ChatGPT programmatically, you need to inform the system who it is in the conversation!

Figure 1-6, shown as follows, gives you a glimpse of the thousands of different roles that ChatGPT can play in a conversation.

System: “You are a...”



Figure 1-6 The System Field in the Chat Playground Allows You to Set the Role That ChatGPT Will Play in the Conversation

2. User

The **User** field (Figure 1-5, item 2) in the ChatGPT Playground is where you type your prompt to ChatGPT, which can be anything that you want, for example, “describe how telemedicine will affect the medical industry.”

3. Assistant (Optional)

When you initially load the Chat Playground, the Assistant field (Figure 1-5, item 3) is not visible. In order to make it appear, you need to click the “+” symbol next to “Add Message.” Now, you may be asking yourself, “Why is this field needed at all?” Well, that’s a fine question. If you want ChatGPT to remember something that it has already told you in a previous conversation, then you need to type into the **Assistant** field anything that it has already told you that you think is relevant in order to continue with the conversation. Remember, it is a vastly powerful AI, but it has amnesia!

4. Add Message (Optional)

The **Add Message** “+” symbol (Figure 1-5, item 4) is where you’d click in order to add either an **Assistant** message to the conversation, or another **User** message. Now, you may ask, “What’s the point of adding yet another **User** message to the conversation when I can type what I want in the original **User** field above?” Good question.

If you wanted to separate your command from your data, then you would use a separate **User** message for that.

Do you remember in Listing 1-4 earlier in this chapter, where we had to use the “###” to separate the command to ChatGPT from the data that we wanted it to analyze? Well, this is now no longer needed because the command would be the 1st **User** message, the data would be the 2nd **User** message.

5. View Code (Optional)

After you have submitted your prompt using the playground, you can click the **View code** button (Figure 1-5, item 5) in order to see the code necessary to send the same prompt using any of the languages that they support.

You may notice that Java is not an officially supported language, but we'll fix that in Chapter 2, when we use ChatGPT as a pair programmer and port their REST interface to Java ourselves.

6. Model (Optional)

Earlier in this chapter, we talked about the various models that are available for developers. Click the model field in order to see a list of models that are available.

You may also see that some models have a month and day associated with their name, which is simply a snapshot of that model. Programmatically selecting a snapshot enables developers to have some sort of predictability in the responses that they will receive from ChatGPT, because the current models are always updated.

7. Temperature (Optional)

As noted earlier in this chapter, the temperature selector ranges between 0 and 2 and allows you to select the “randomness” of the response.

8. Maximum Length (Optional)

Do you remember the discussion earlier in this chapter about tokens? By selecting anything in the range for this item, you can adjust the

number of tokens (which directly affects the number of words) in the response.

Try It Now! Experimenting with the “System” Role

Now that we’re familiar with several of the features of the Chat Playground, let’s send our first prompt using the settings discussed above. Listings 1-7 and 1-8 use the same prompt asking ChatGPT to give a few paragraphs on telemedicine, but the role of the system is vastly different from each other.

System: You are a strictly factual researcher

User: Write 3 paragraphs on pros and cons of telemedicine

Listing 1-7 PROMPT. The Pros and Cons of Telemedicine as a Researcher

System: You are a highly opinionated health blogger who always has stories with first hand experience

User: Write 3 paragraphs on pros and cons of telemedicine

Listing 1-8 PROMPT. The Pros and Cons of Telemedicine as an Opinionated Health Blogger

You are encouraged to try these two prompts yourself and see what the responses are. Adjust the settings for the temperature and token length to get familiar with how those parameters affect the outcome.

Conclusion

You just learned more about how ChatGPT can be used by developers. We covered some of the basics of the Chat Playground, which is a web interface for developers to interact with the ChatGPT API.

We talked about how to set the system, user, and assistant roles in the Chat Playground and how to adjust settings such as the temperature and maximum length of output.

You learned about some of the parameters and terminology necessary to use the Chat Playground, such as the model, the temperature, and tokens. Getting familiar with the parameters of the Chat Playground is essential to knowing how to use the REST API since the Playground is a subset of capabilities offered by the REST API.

In the next chapter, we'll see how to use ChatGPT as your "pair programmer" and port the officially supported ChatGPT REST interface to Java.

2. Using ChatGPT As Your Java Pair-Programmer

Bruce Hopkins¹✉
(1) Beaverton, OR, USA

I'm a big fan of some of the practices of XP (eXtreme Programming), and especially pair-programming. No matter what flavor of pair-programming that you prefer, it involves two engineers sitting down at the same screen and solving the same problem together. One of the biggest benefits that you get is a fresh set of eyes on a problem, and of course, you now have two engineers who have "touched" the codebase instead of one. Sometimes you can have one engineer write the code and the other write the tests and the comments. No matter how you slice it, it's all good stuff.

Now, the OpenAI REST APIs for ChatGPT and their other models are officially supported in Python, Typescript, and of course, cURL (which is the de facto standard for REST APIs).

There are a few Java APIs that exist that were created by third-party developers, but the biggest problem (in my opinion) is that this space is rapidly changing. OpenAI is constantly updating both their models and their HTTP interfaces, and as a result, they are adding or deprecating features and functionality frequently. If you choose to use a third-party Java API for your project, you risk the problem of using an API that is outdated or out of sync with the OpenAI REST API.

So, in this chapter, we're going to use ChatGPT as our pair-programmer and simply port the official OpenAI REST APIs directly to Java. Every time OpenAI makes any changes to their officially supported languages and interfaces, we have everything needed to update our own library instantly. Let's do this!

Creating Your First Java ChatGPT App: `ListModels.java`

We're actually going to accomplish two tasks at once here. We're going to create a basic application in Java using the OpenAI APIs, and in the process, we're going

to verify that we've properly obtained an API key. So, needless to say, in case you haven't done so already, follow the instructions in Chapter 1 to create your OpenAI developer account and obtain your API key. Going forward, all the code samples in this book require a valid API key.

List Models Endpoint

One of the most basic (but also essential) services that we can call is the **List Models** endpoint. Why, you may ask? The List Models endpoint allows you to get a list of all the AI models that are currently available for use by developers via the REST API.

Creating the Request

Table 2-1 lists all the HTTP parameters necessary to call the List Models endpoint.

Table 2-1 The HTTP Parameters Necessary to Call the List Models Endpoint

HTTP Param	Description
Endpoint URL	https://api.openai.com/v1/models
Method	GET
Header	Authorization: Bearer \$OPENAI_API_KEY

Handling the JSON Response

After successfully invoking the List Models endpoint, the service will provide a JSON response with the structure shown in Table 2-2.

Model (JSON)

Table 2-2 The Structure of the Model JSON Object

Field	Type	Description
object	String	This always returns the literal, "list"
data	Array	A list of AI models offered by OpenAI
↳ id	String	The unique ID of the AI model, which is essentially the full name of the model
↳ object	String	This always returns the literal, "model"
↳ created	integer	The creation date for the model
↳ owned_by	String	The name of the organization that owns the model

Field	Type	Description
permission	Array	This is an array of metadata for the permissions for the model
root	String	This is typically a copy of the data.id field, which is the model's name
parent	String	This is currently an unused field

Note Since JSON objects can contain arrays (which can be hard to represent in a table), we're using the notation “↳” to indicate the elements of an array. As you can see from the previous table, “id,” “object,” “created,” and “owned_by” are all elements of the “data” array in the JSON response.

Now that we have the details of the HTTP request and response, we can now create the ListModel.java app using the prompt shown in Listing 2-1.

System: You are a Java developer

User: Take the following cURL command and convert it to Java. Name the file, "ListModel.java"

User: curl https://api.openai.com/v1/models \
-H "Authorization: Bearer \$OPENAI_API_KEY"

Listing 2-1 PROMPT. Using ChatGPT to Create the ListModels.java App

So, after sending the prompt, ChatGPT will provide you an answer as shown in Listing 2-2.



```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;

public class ListModels {

    public static void main(String[] args) {
        try {
            // Create URL object for the API endpoint
            URL url = new
URL("https://api.openai.com/v1/models");

            // Create HttpURLConnection object
            HttpURLConnection conn = (HttpURLConnection)
url.openConnection();

```

```
// Set request method
conn.setRequestMethod("GET");

// Set Authorization header
conn.setRequestProperty("Authorization",
"Bearer $OPENAI_API_KEY");

// Get response code
int responseCode = conn.getResponseCode();

// If the request was successful (status code
200)
if (responseCode ==
HttpURLConnection.HTTP_OK) {
    // Read the response from the API
    BufferedReader in = new
    BufferedReader(new
    InputStreamReader(conn.getInputStream()));
    String inputLine;
    StringBuilder response = new
    StringBuilder();
    while ((inputLine = in.readLine()) !=
null) {
        response.append(inputLine);
    }
    in.close();

    // Print the response
    System.out.println(response.toString());
} else {
    System.out.println("Request failed.
Response Code: " + responseCode);
}

// Close the connection
conn.disconnect();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Listing 2-2 RESPONSE. ListModels.java

Fantastic! Now all you have to do is to replace “\$OPENAI_API_KEY” with the value of your actual API key.

After running this class, you’ll get a JSON file that contains all the models offered by OpenAI’s API, as shown in Listing 2-3.



```
{  
    "object": "list",  
    "data": [  
        {  
            "id": "text-search-babbage-doc-001",  
            "object": "model",  
            "created": 1651172509,  
            "owned_by": "openai-dev"  
        },  
        {  
            "id": "gpt-4",  
            "object": "model",  
            "created": 1687882411,  
            "owned_by": "openai"  
        },  
        {  
            "id": "gpt-3.5-turbo-16k",  
            "object": "model",  
            "created": 1683758102,  
            "owned_by": "openai-internal"  
        },  
        {  
            "id": "curie-search-query",  
            "object": "model",  
            "created": 1651172509,  
            "owned_by": "openai-dev"  
        },  
        {  
            "id": "text-davinci-003",  
            "object": "model",  
            "created": 1669599635,  
            "owned_by": "openai-internal"  
        },  
        {  
            "id": "text-search-babbage-query-001",  
            "object": "model",  
        }  
    ]  
}
```

```

    "created": 1651172509,
    "owned_by": "openai-dev"
},
{
    "id": "babbage",
    "object": "model",
    "created": 1649358449,
    "owned_by": "openai"
},
...

```

Listing 2-3 RESPONSE. The Partial JSON Response After Running ListModels.java

Listing 2-3 is a partial list due to the sheer size of the number of models available for developers to use! The good news, however, is that the full response is provided as a table in the Appendix.

Now that we can programmatically get a list of models available to use, it's time to send prompts to ChatGPT using Java. This is accomplished using the Chat Endpoint.

Chat Endpoint

The Chat Endpoint (formerly called, “Chat Completion”) is a REST service that’s basically a 1-to-1 representation of what you can do in the Chat Playground; therefore, this service should feel like second nature to you.

Creating the Request

Table 2-3 lists all the HTTP parameters necessary to call the Chat Endpoint.

Table 2-3 The HTTP Parameters for the Chat Endpoint

HTTP Param	Description
Endpoint URL	https://api.openai.com/v1/chat/completions
Method	POST
Header	Authorization: Bearer \$OPENAI_API_KEY
Content-Type	application/json

Table 2-4 describes the format of the JSON object necessary for the request body for the Chat Endpoint. After a quick glance, you can see that only a few fields are actually required in order to successfully invoke the service.

Chat (JSON)

Table 2-4 The Structure of the Chat JSON Object

Field	Type	Required?	Description
model	String	Required	<p>The ID of the model you want to use for Chat Completion. Compatible models include</p> <ul style="list-style-type: none"> • gpt-4 • gpt-4-0613 • gpt-4-32k • gpt-4-32k-0613 • gpt-3.5-turbo • gpt-3.5-turbo-0613 • gpt-3.5-turbo-16k • gpt-3.5-turbo-16k-0613
messages	Array	Required	<p>An array of messages that are a part of the ongoing conversation.</p> <p>Each message in the array has two properties: "role" and "content."</p>
↳ role	String	Required	<p>Specifies the role of the message, which can be any of the following:</p> <ul style="list-style-type: none"> • "system" • "user" • "assistant" • "tool"
↳ content	String	Required	Contains the text of the message for the specified role.
tools	Array	Optional	<p>This allows you to specify a list of tools that the model can call. Currently, the only supported type of tool is a function.</p> <p>This parameter enables you to define a set of functions for which the model can generate JSON inputs.</p>
↳ type	String	Required	This is type of the tool, which can be any of the

Field	Type	Required?	Description
			<p>following:</p> <ul style="list-style-type: none"> • "function"
↳ function	Array	Optional	An array of functions that the model may use to call in the Chat Completion.
↳↳ name	String	Required	<p>The name of the function to be called. Valid names must be a-z, A-Z, 0-9, or contain underscores and dashes.</p> <p>The maximum length is 64 characters.</p>
↳↳ description	String	Optional	A description of what the function does. This helps the model to decide whether to call the function in the Chat Completion.
↳↳ parameters	JSON object	Required	The parameters the function accepts in the format of a JSON Schema object.
tool_choice	<p>String or JSON object</p> <p>default:</p> <p>"none" when no functions are included in the request</p> <p>"auto" when functions are included in the request</p>	Optional	<p>This allows you to determine which, if any, function the model should invoke.</p> <p>When set to "none," the model will refrain from calling any function and will solely generate a message response.</p> <p>When set to "auto," the model has the flexibility to choose between generating a message response or invoking a function based on its internal decision-making process.</p>
temperature	<p>Number or null</p> <p>default: 1</p>	Optional	<p>Valid values range between 0 and 2.</p> <p>Controls the randomness of the model's output.</p> <p>The best practice is to adjust the top_p or temperature, but not both.</p>
top_p	<p>Number or null</p> <p>default: 1</p>	Optional	Valid values range between 0 and 1. Indicates whether to

Another random document with
no related content on Scribd:

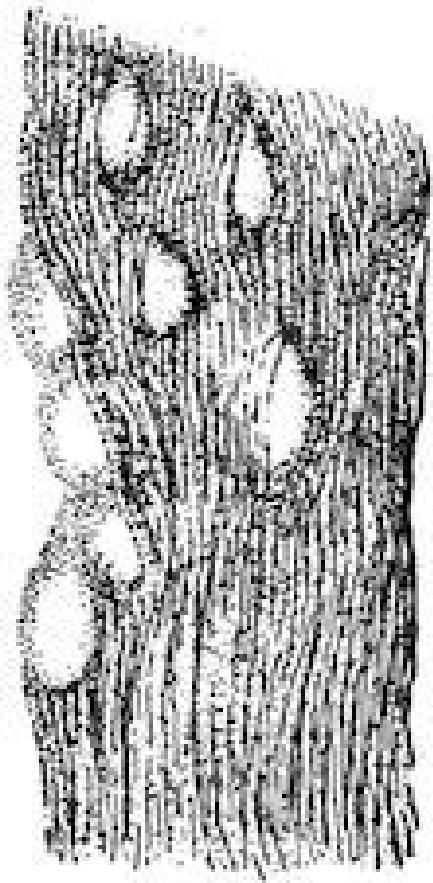


FIG. 30.—A piece of pork heavily infected with pork measles (*Cysticercus cellulosæ*), natural size. (Stiles, Report U.S.A. Bureau of Agriculture, 1901.)

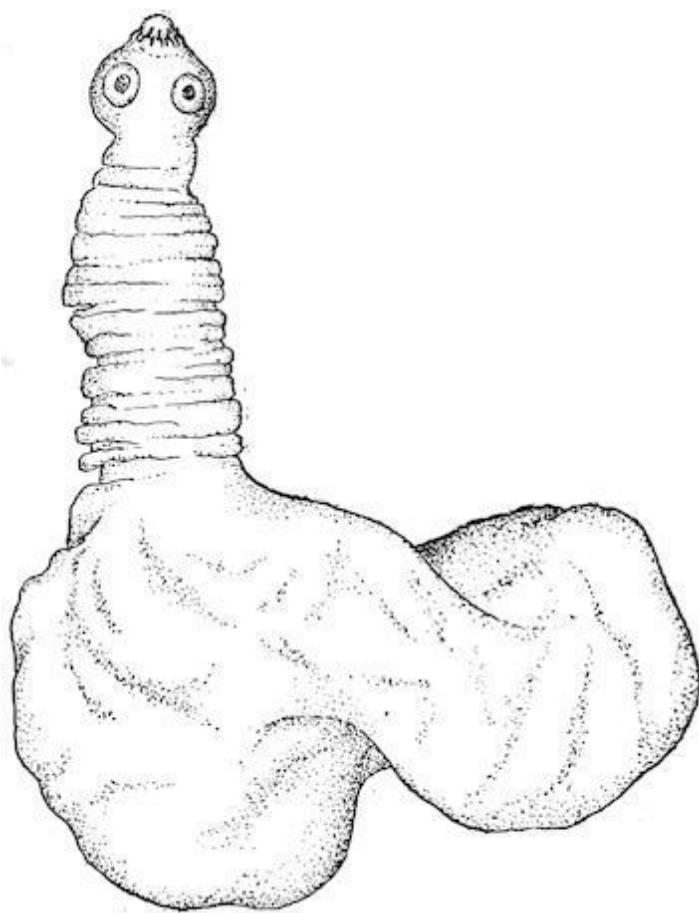


FIG. 31.—An isolated pork-measle bladder-worm (*Cysticercus cellulosæ*), with extended head, greatly enlarged. (Stiles, Report U.S.A. Bureau of Agriculture, 1901.)

Causation. The cause of cysticercus disease in the pig may be summed up in one phrase—viz., ingestion of eggs or embryos of *Tænia solium*.

Young animals alone seem to contract the disease. After the age of eight to ten months they appear almost entirely proof against it.

It is very rare in animals reared in confinement, but is relatively common in those roaming at liberty; because they are much more likely to discover human excrement and the embryos of *tænia*. The eggs having been swallowed, the six-hooked embryos are set at liberty in the intestine, perforate the tissues, enter the vessels, and are carried by the blood into all parts of the body. Those alone develop well which reach the interstitial and intermuscular

connective tissue. The others in the viscera usually disappear. Their presence in the depths of the muscles produces slight general disturbance and signs of local irritation, due to the development of the cyst itself. At the end of a month the little vesicle is large enough to be visible to the naked eye; in forty to forty-five days it is as large as a mustard seed, and in two months as a grain of barley. Its commonest seats are the abdominal muscles, muscular portions of the diaphragm, the psoas, tongue, heart, the muscles of mastication, intercostal and cervical muscles, the adductors of the hind legs, and the pectorals.

Symptoms. The symptoms of invasion are so little marked as usually to pass undetected. Occasionally, when large quantities have been ingested, signs of enteritis may occur, but these are generally ascribed to some entirely different cause. In some cases there is difficulty in moving, and the grunt may be altered.

Certain authors declare that the thorax is depressed between the front limbs, but this symptom is of no particular value, and is also common to osseous cachexia and rachitis. Paralysis of the tongue and of the lower jaw is of greater importance. In exceptional cases, where the cysticerci are very numerous and penetrate the brain, signs of encephalitis, vertigo, and turning sickness (gid, sturdy) may be produced. These signs, however, disappear, and the cysticerci undergo atrophy. Interference with movement may give rise to suspicion when the toes of the fore and hind limbs are dragged along the ground, and thus become worn. This peculiarity is due to the presence of cysts in the muscles of the limbs, but it occurs in an almost identical form in osseous cachexia.

One symptom alone is pathognomonic, and it appears only at a very late stage—viz., the presence of cysts under the thin mucous

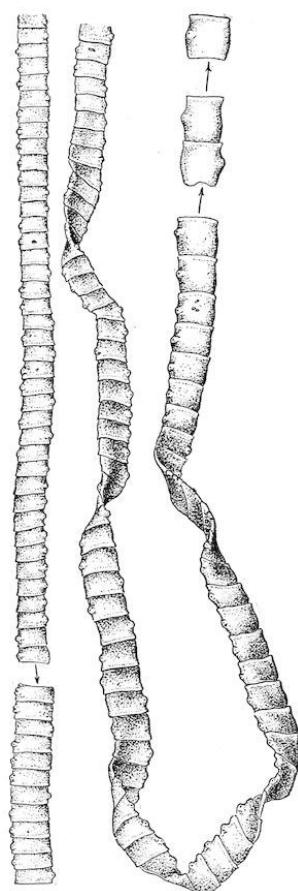


FIG. 32.—Several portions of an adult pork-measle tapeworm (*Tænia solium*), natural size.

membranes which are accessible to examination, such as those of the tongue and eye.

Visual examination then reveals beneath these mucous membranes the presence of little greyish-white, semi-transparent grains the size of

(Stiles, Report
U.S.A. Bureau of
Agriculture,
1901.)

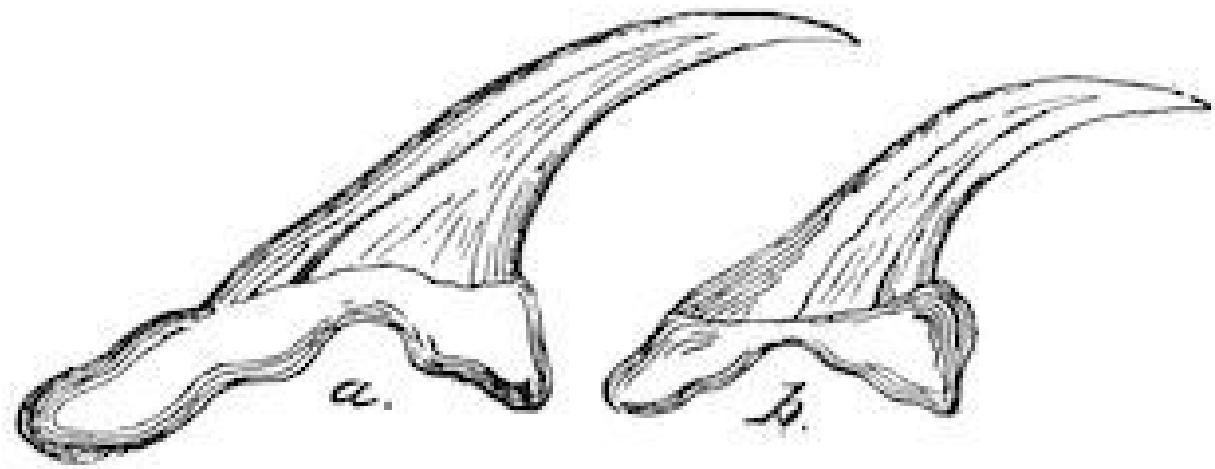


FIG. 33.—Large (a) and small (b) hooks of pork-measle tapeworm (*Taenia solium*). $\times 280$. (After Leuckart.)

a grain of barley, or even larger. Unfortunately, in an animal so difficult to handle as the pig, this visual examination is decidedly troublesome, and is usually replaced by palpation. In many instances the disease does not attract attention during the patient's life, and is only discovered on slaughter in consequence of the lesions by which it is characterised.

Diagnosis. As the characteristic lesions of cysticercus disease are to be found in the depths of the muscular and connective tissues, and as the external symptoms may be regarded as of doubtful significance, the diagnosis can only be confirmed during life by manual examination of the tongue. This examination of the tongue has been practised since the earliest times. Aristophanes even speaks of it, and in the Middle Ages it was performed under sworn guarantees. The regulations concerning the inspection of meat have finally led to the suppression of this calling.

In this method of examining the tongue, the operator commences by throwing the animal on its side, usually on the right side, and holding it in this position by placing his left knee on its neck. He then

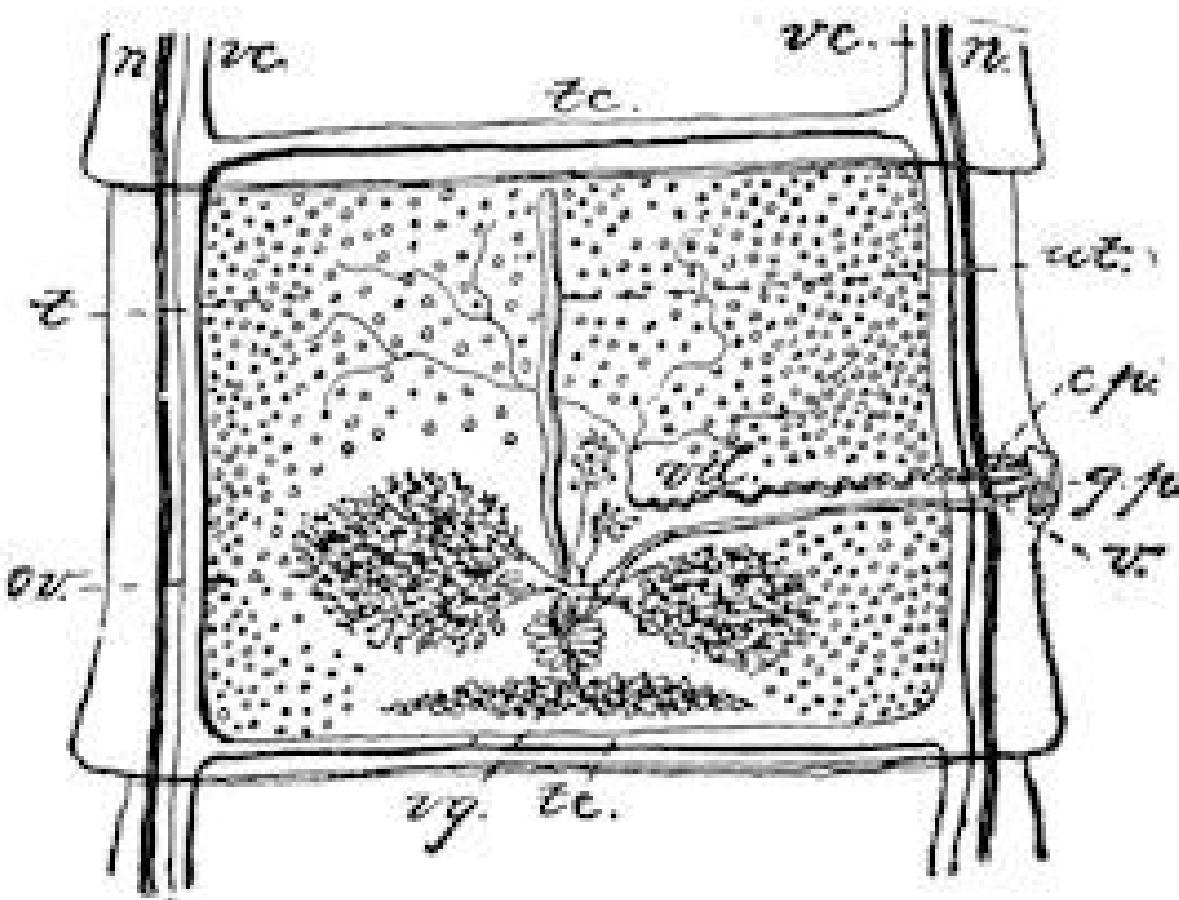


FIG. 34.—Mature sexual segments of pork-measle tapeworm (*Tænia solium*), showing the divided ovary on the pore side. *cp*, Cirrus pouch; *gp*, genital pore; *n*, nerve; *ov*, ovary; *t*, testicles; *tc*, transverse canal; *ut*, uterus; *v*, vagina; *vc*, ventral canal; *vd*, vas deferens; *vg*, vitellogene gland. $\times 10$. (After Leuckart.)

passes a thick stick between the jaws and behind the tusks, opens the mouth obliquely, raising the upper jaw by manipulating the stick. Finally he fixes one end of this last by placing his foot upon it, and holds the other extremity by slipping it under his left arm. In this position he is able to grasp the free end of the tongue and by digital palpation to examine the tongue itself, the gums, the free portions of the frænum linguæ, etc.

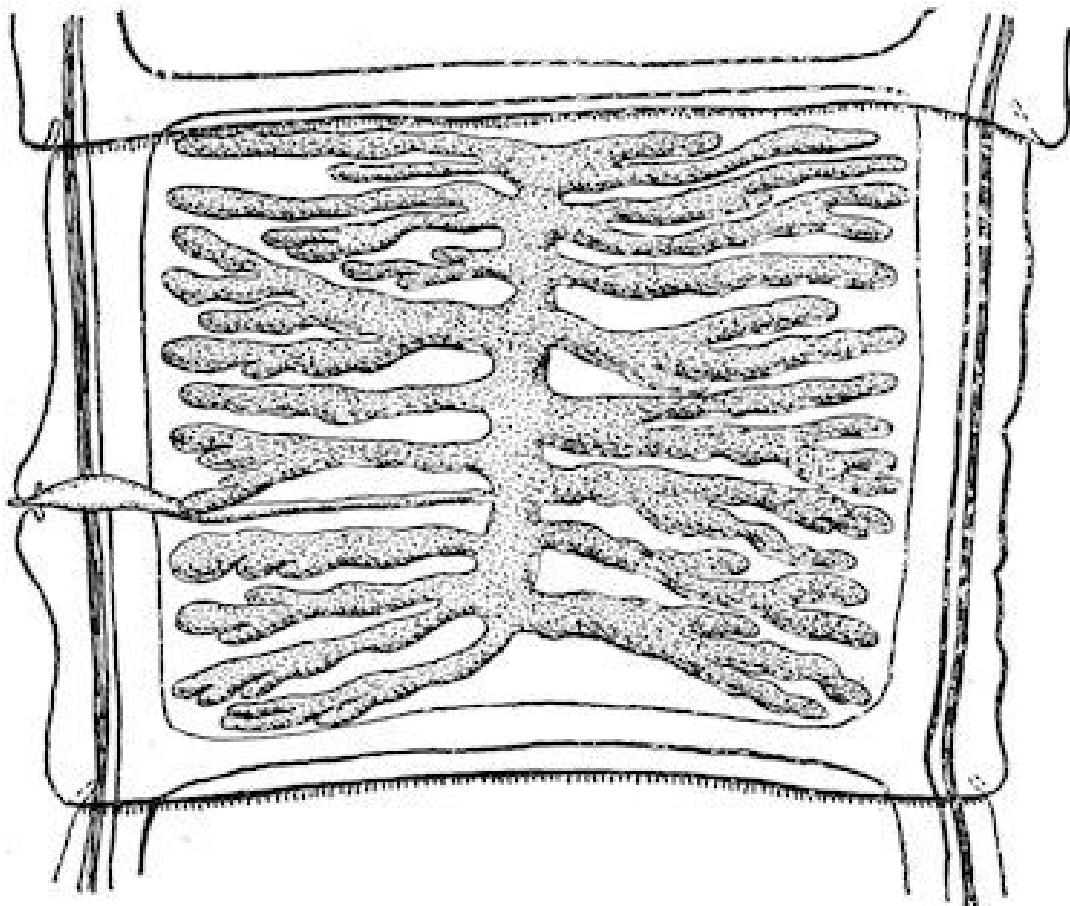


FIG. 35.—Gravid segment of pork-measle tapeworm (*Tænia solium*), showing the lateral branches of the uterus enlarged. (Stiles, Report U.S.A. Bureau of Agriculture, 1901.)

If he discovers cysts, the diagnosis is confirmed, but failure to do so by no means disposes of the possibility of infection. Railliet declares that about one animal in four or five shows no cysts beneath the tongue, and, moreover, fraud is possible in this connection, it being quite possible to prick the little cysts with a needle so that the liquid contents escape, and examination gives no positive result. For these reasons *intra-vitam* examination alone is now discounted, and the chief reliance is placed on post-mortem search.

Prognosis. The prognosis is very grave, not on account of danger to the lives of the infected, but because infected meat may be offered for human consumption. Should such meat, in an insufficiently cooked condition, be eaten by man, its ingestion is followed by the development of *Tænia solium*. If cooking were always perfect it

would destroy the cysticerci, but the uncertainty in this respect should prevent such meat being consumed. The cysticerci are killed at a temperature of 125° to 130° Fahr.

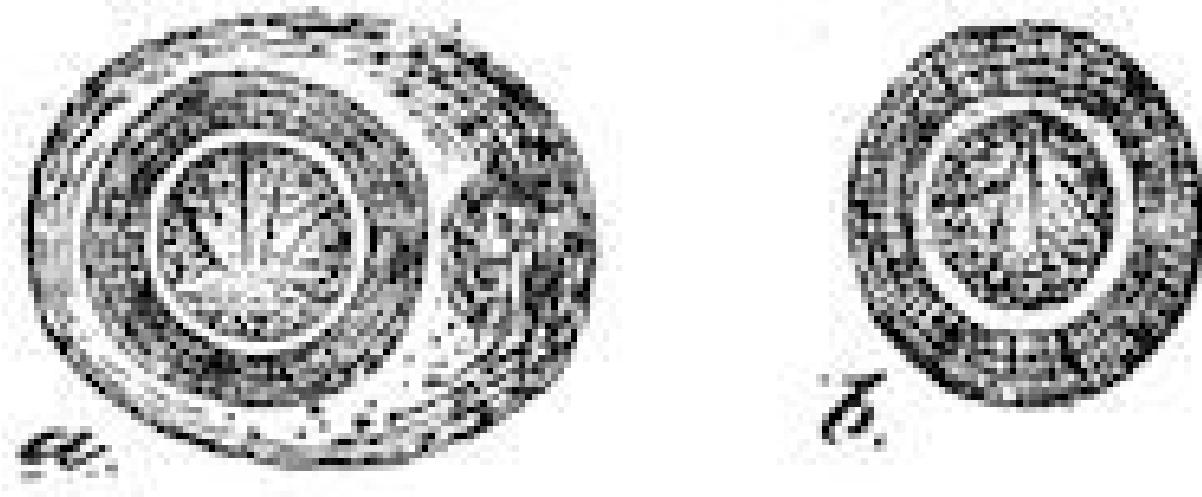


FIG. 36.—Eggs of pork-measle tapeworm (*Tænia solium*): *a*, with primitive vitelline membrane; *b*, without primitive vitelline membrane, but with striated embryophore. $\times 450$. (After Leuckart.)

Lesions. The lesions are represented by cysts alone—*i.e.*, by semi-transparent bladders, each of which contains a scolex or head armed with four suckers and a double crown of hooks. The little bladders are most commonly found in the muscles, lodged in the interfascicular tissue, which they slightly irritate.

The number present varies extremely, depending on the intensity of infestation and the number of eggs swallowed. Whilst in some cases difficult to discover, in others they are so numerous that the tissues appear strewn with them.

They are commonest in the muscles of the tongue, neck, and shoulders, in the intercostal and psoas muscles, and in those of the quarter.

The viscera—viz., the liver, kidneys, heart, lungs, etc.—are less commonly infested, and in these organs the cysts degenerate very rapidly. In animals which have been infested for a long time, the cysts may even have undergone caseo-calcareous degeneration, the liquid being absorbed and the lesions presenting the appearance of little oblong firm nodules.

On cutting through masses of muscle the vesicles protrude from between the bundles.

In young animals, infestation with cysticerci causes wasting and ill-health; subsequently the patients improve in appearance, later on fatten, and gain marketable condition.

Of the carcases examined in Prussian slaughter-houses between 1876–82, one in every 305 was found infested; between 1885–93, one in every 537.

Treatment. There is no curative treatment. Only preventive measures are of value. These are confined to rendering it impossible for animals to ingest eggs of the *Tænia solium*.

Cysticercus disease is rare in the north, centre, and east of France, and in districts where animals are reared in confinement. It is commoner where pigs are at liberty, such as Limousin, Auvergne, and Perigord. It is frequent in North Germany, where the custom of eating half-cooked meat contributes to the propagation of *Tænia solium*. It is also frequent in Italy.

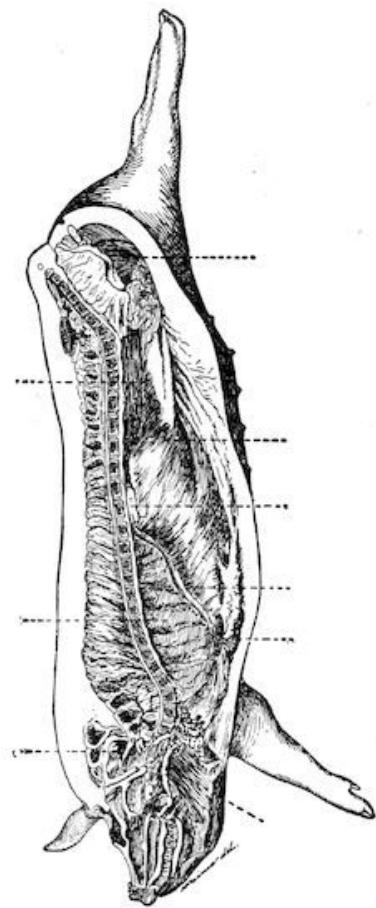


FIG. 37.—Half of hog, showing the portions most likely to become infested with pork measles. (After Ostertag.)

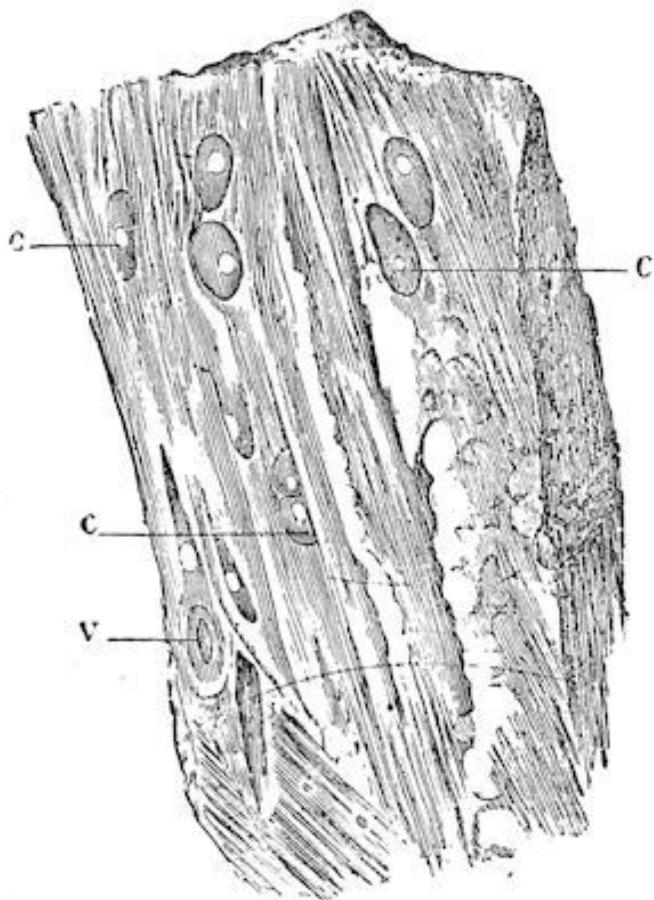


FIG. 38.—*Cysticercus cellulosæ* in pork. *c*, Cysts; *v*, fibrous tissue capsule which forms around the cyst.

BEEF MEASLES.

Causation. The disease of beef measles is due to the penetration into the connective and muscular tissues of embryos of the *Tænia saginata*, or unarmed tænia of man.

This disease, unlike that of the pig, has only been recognised within comparatively recent times, and only after Weisse's experiments (St. Petersburg, 1841) on feeding with raw flesh was attention drawn to it, although as early as 1782 the *Tænia saginata* had been described by Goëze.

Measles in the ox is rarely seen in France, but is common in North and East Africa. Alix has found it in Tunis, Dupuys and Monod in Senegal, and it is common in the south of Algeria. The disease is due

simply to oxen swallowing eggs or embryos of the unarmed tænia, a fact which explains the frequency of the disease in places where the inhabitants are of nomad habits, and consequently disregard the most elementary rules of public and general hygiene.

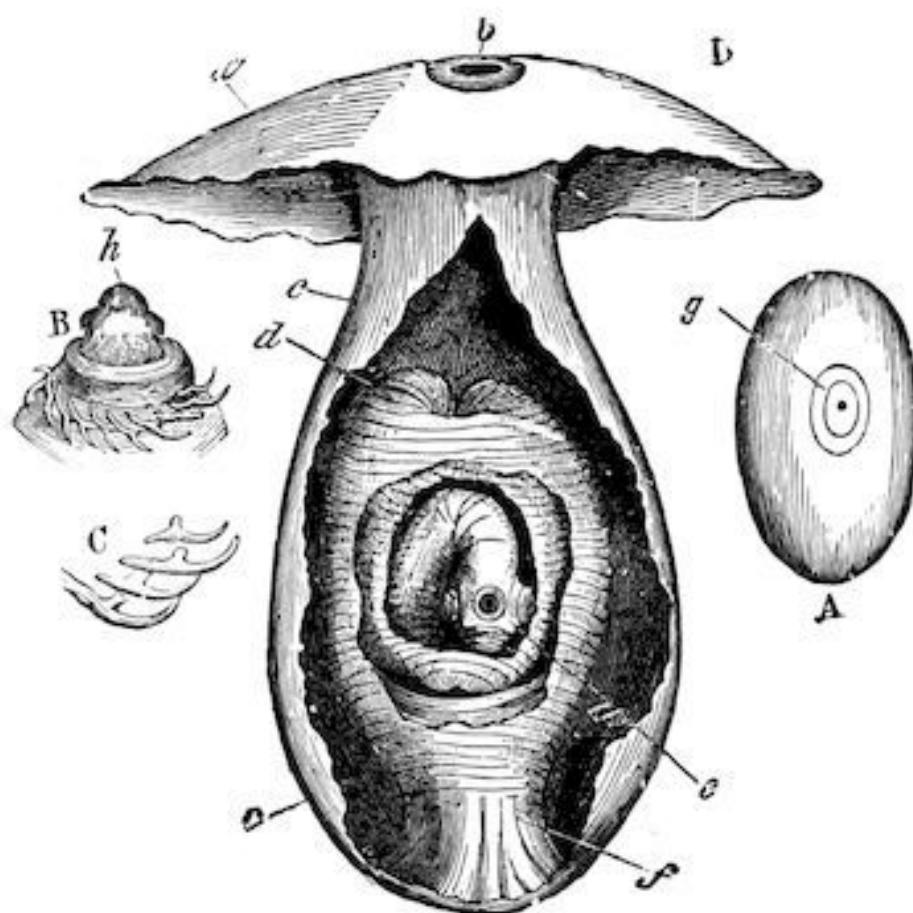


FIG. 39.—Anatomy of the *Cysticercus cellulosæ* (after Robin). A, Cyst; B, scolex with hooks; C, hooks; D, magnified fragment of cyst.

Furthermore, cattle in the Sahara, in Senegal and in the Indies, have a very marked habit of eating ordure, and as no attempts are made to prevent it, the risk to these animals is greatly increased.

As in the pig, the embryos which reach the stomach and intestine penetrate into the circulatory system, and are thereby distributed throughout the entire organism.

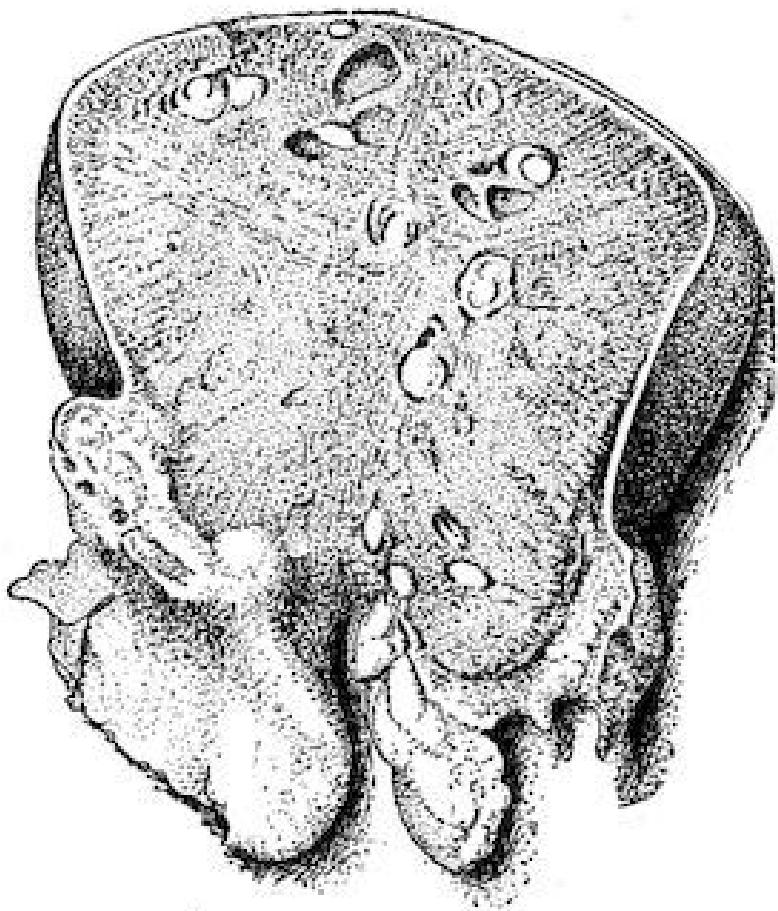


FIG. 40.—Section of a beef tongue heavily infested with beef measles, natural size (Stiles, Annual Report U.S.A. Bureau of Agriculture, 1901).

account of a case experimentally infected:—

The development of the cysticercus is complete in forty days, and if swallowed by man in infected meat after this period it again gives rise to the *Tænia saginata*.

The age of the animals seems of less importance than in the case of the pig, for Ostertag and Morot have seen cases of beef measles in animals of ten years old.

Symptoms. The symptoms are still less marked than in the pig, and in ordinary cases of infection always escape observation. Stiles, however, gives the following

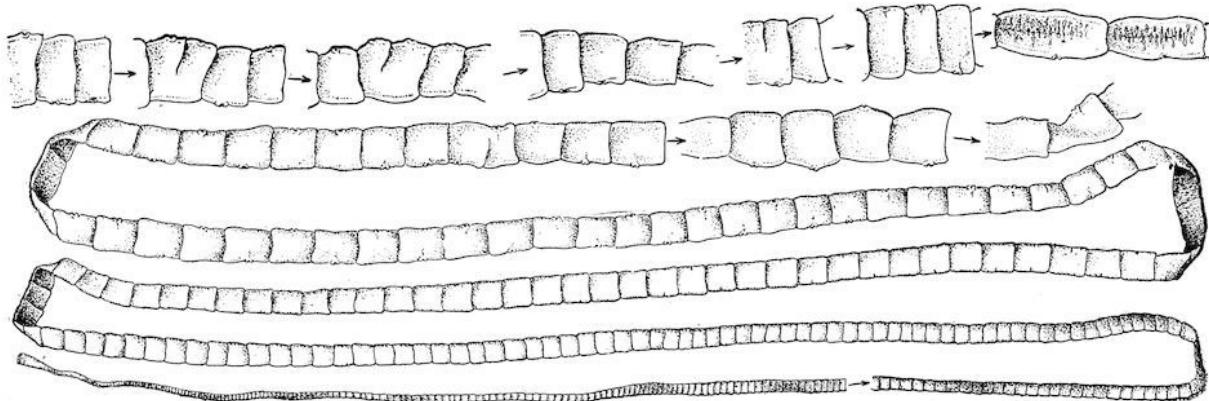


FIG. 41.—Several portions of an adult beef-measle tapeworm (*Taenia saginata*) from man, showing the head on the anterior end and the gradual increase in size of the segments, natural size. (Stiles, Annual Report U.S.A. Bureau of Agriculture, 1901.)

“Symptoms. Four days after feeding segments of *T. saginata* to a healthy three-months-old calf, the patient showed a higher temperature (the normal temperature was 39.2° C.). The calf ate but little on that day, showed an accelerated pulse, swollen belly, staring coat, and upon pressure on the sides showed signs of pain. The next day the animal was more lively, ate a little, and for nine days later did not show any special symptoms except pain on pressure of the abdominal walls, and a slight fever. Nine days after the infection the temperature was 40.7° C., pulse 86, respiration 22; the calf laid down most of the time, lost its appetite almost entirely, and groaned considerably. When driven it showed a stiff gait and evident pain in the side. The fever increased gradually, and with it the feebleness and low-spiritedness of the calf, which now retained a recumbent position most of the time, being scarcely able to rise without aid, and eating only mash with ground corn. Diarrhoea commenced, the temperature fell gradually, and on the twenty-third day the animal died. The temperature had fallen to 38.2° C. During the last few days the calf was unable to rise; in fact, it could scarcely raise its head to lick the mash placed before it. Pulse was reduced by ten beats. On the last day the heart-beats were very much slower, yet firm, and could be plainly felt. Several days before death the breathing was laboured, and on the last day there was extreme dyspnœa.”

Diagnosis. In forming a diagnosis we meet with the same difficulty as in the case of the pig. It is always easy to examine the

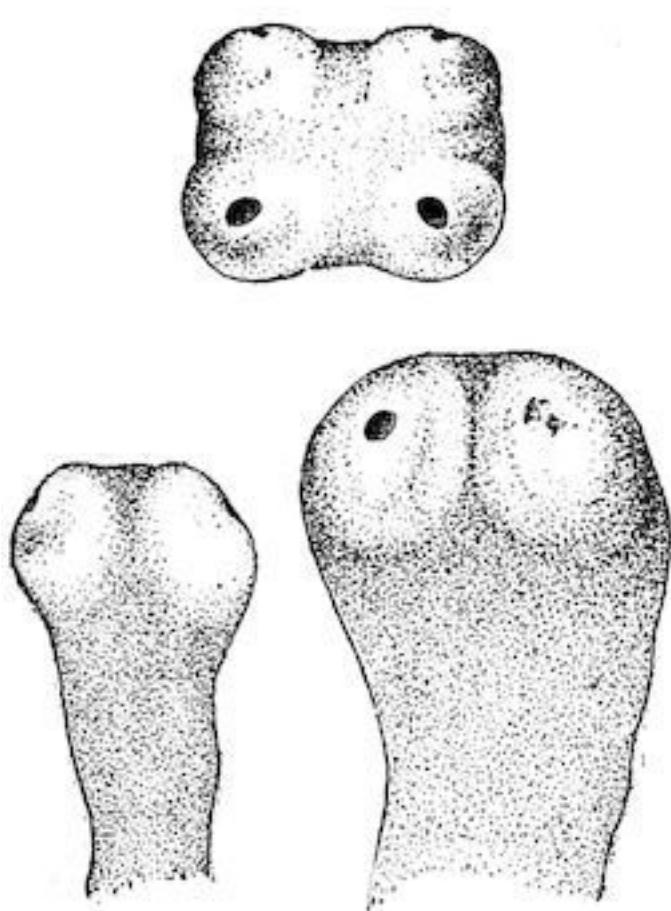


FIG. 42.—Apex, dorsal, and lateral views of the head of beef-measle tapeworm (*Tænia saginata*), showing a depression in the centre of the apex. $\times 17$. (Stiles, Report U.S.A. Bureau of Agriculture, 1901.)

twenty days destroys the vitality of the parasite.

Lesions. The lesions are confined to the presence of the cyst and of two little zones of chronic inflammation immediately surrounding it. Unless heavily infested the subjects fatten just as well as others.

The vesicles are semi-transparent, $3/16$ inch to $1/4$ inch in length, slightly ovoid in form, and contain a *tænia* head with four suckers, but without hooks.

tongue; but when visible lesions are absent diagnosis in the case of the ox remains doubtful and problematical even more than in the pig.

In the carcase, diagnosis is much easier. The cysts are sought for, as in the pig, by making sections of muscle, those usually selected being the pterygoid, cervical, cardiac, and psoas muscles, and those of the quarters.

Prognosis. The prognosis is grave, not indeed for the infected animals, which seem little injured by the parasite, but for human beings, who run the risk of contracting *Tænia inermis* by eating insufficiently-cooked meat.

A temperature of 115° to 120° Fahr. destroys the cysticerci, but in roast meats the central temperature of the mass always remains below this figure.

Salting for fifteen to

In seven to eight months the cysts undergo degeneration, the liquid is absorbed, and calcium salts are deposited throughout the mass. The lesions which remain have, in the ox, the appearance of interstitial disseminated tuberculosis.

There is no curative treatment. The infested animal recovers spontaneously with the lapse of time, for the cysticerci undergo degenerative processes, but the flesh of such animals is of little commercial value.

From a preventive standpoint we can only hope to improve matters by a gradual and progressive change in social and public hygienic conditions.

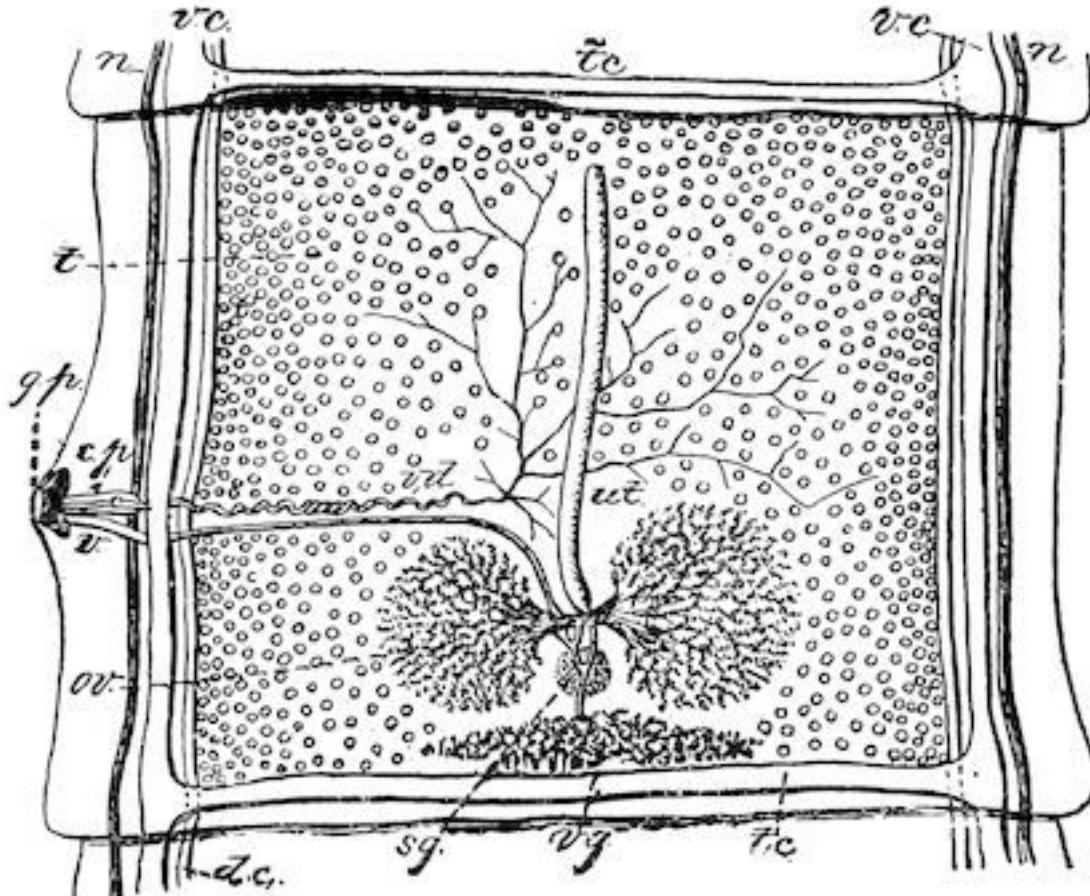


FIG. 43.—Sexually mature segment of beef-measle tapeworm (*Tænia saginata*). *c.p.*, Cirrus pouch with cirrhus; *d.c.*, dorsal canal; *g.p.*, genital pore; *n.*, lateral longitudinal nerves; *ov.*, ovary; *sg.*, shell gland; *t.*, testicles; *ut.*, median uterine stem, enlarged (in part after Leuckart); *v.*, vagina; *v.c.*, ventral canal, connected by transverse canal; *tc.*, *vd.*, vas deferens; *vg.*, vitellogene gland.

When the life of the nomad shall have been entirely replaced by that of the highly-civilised European and private hygienic precautions have rendered it impossible for animals to obtain access to segments or eggs of the *Tænia saginata*, beef measles will disappear.

At present, in the countries where the disease is common, one experiences a feeling of astonishment that it is not far more frequent; for experiment has shown that a person infected with one unarmed tapeworm expels with the faeces an average of four hundred proglottides per month, each proglottis or segment of the worm

containing about 30,000 eggs, each of which is capable of developing into a tapeworm.

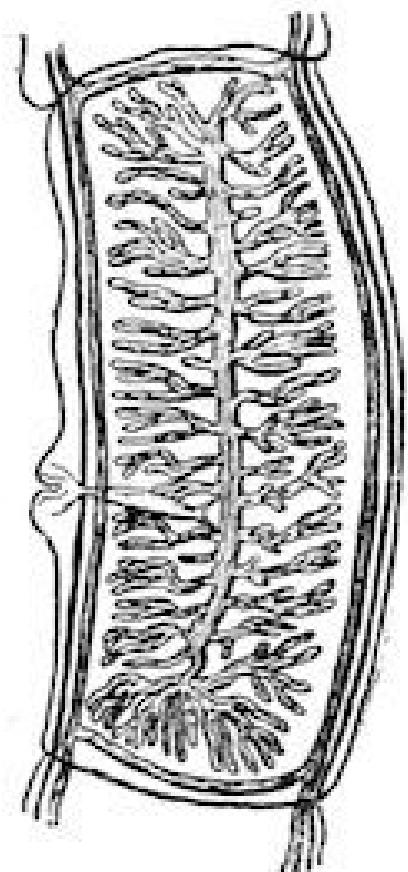
Beef measles is rather common in Germany, but rare in France, Switzerland, and Italy.

TRICHINIASIS—TRICHINOSIS.

Trichinosis is a disease caused by the entrance into the body of the *Trichina spiralis*. This parasite is swallowed in the larval form, and undergoes sexual changes in the intestine, at first producing intestinal trichinosis, which represents the first phase in the development of the disease.

The trichinæ breed rapidly. The embryos penetrate into or are directly deposited in the blood-vessels, which convey them to all parts of the body, thus setting up the second phase of the disease, known as muscular trichinosis.

Trichinosis as a disease has long been recognised. Peacock in 1828 and J. Hilton in 1832 mentioned the existence of the cysts of trichinæ; Owen in 1835 gave the name of *Trichina spiralis* to the parasites contained in the cysts. Trichinosis being common in Germany at that time, Virchow and Leuckart undertook its investigation, but mistook other nematodes of the intestine for the *Trichina spiralis*. In 1847 Leydy recognised that trichinosis occurred in American pigs.



In 1860 Zenker found muscular and intestinal trichinosis on post-mortem examination of a girl who had been suspected of suffering from typhoid fever, and a carefully conducted inquiry revealed the fact that this girl had some time previously eaten a quantity of raw ham. Virchow and Leuckart returned to their

FIG. 44.—Gravid segment of beef-measle tapeworm (*Tænia saginata*), showing lateral branches of the uterus, enlarged. (Stiles, Annual Report U.S.A. Bureau of Agriculture, 1901.)

investigations, and the life history of the parasite soon became definitely known.

Causation. Trichinosis is capable of attacking all mammifers without exception, from a man to a mouse; and most animals which can be made the subjects of experiment contract the disease in varying degrees.

The intestinal form is seen in birds, but

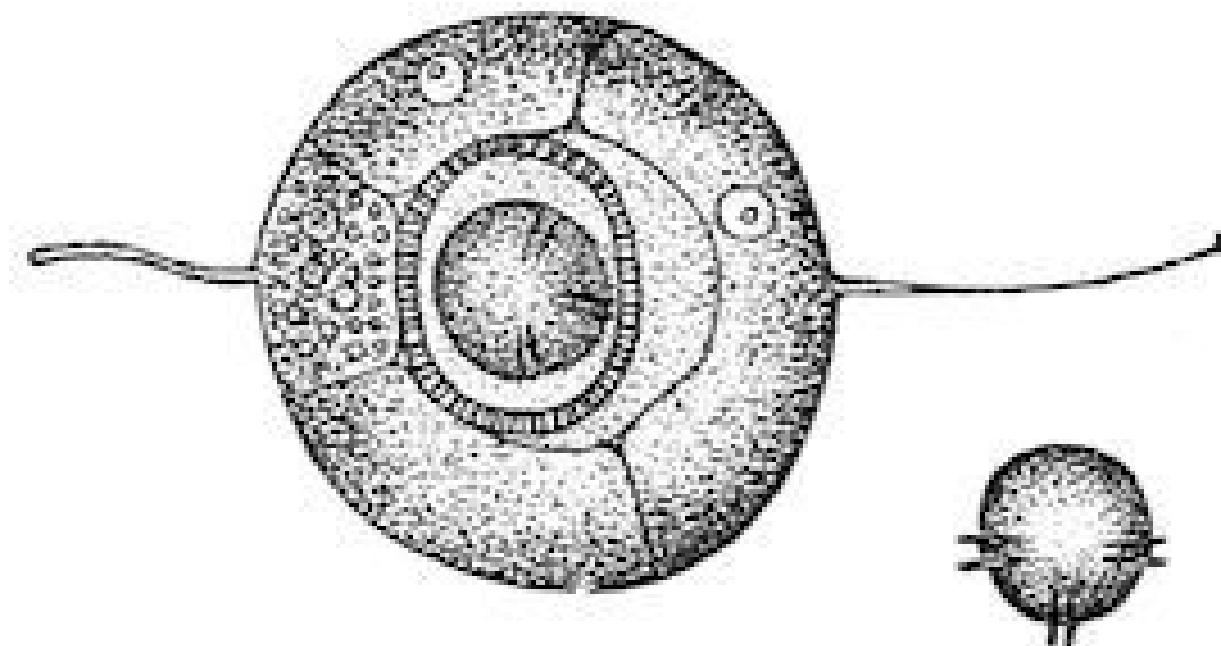


FIG. 45.—Egg of beef-measle tapeworm (*Tænia saginata*), with thick egg-shell (embryophore), containing the six-hooked embryo (oncosphere), enlarged. (After Leuckart.)

the muscles do not become infested by the embryos.

Cold-blooded animals are proof against the disease.

After the ingestion of meat containing cysts of the parasite, the processes of gastric and intestinal digestion set the larvæ at liberty. These larvæ become sexual at the end of four to five days, and the females, which are usually twice as numerous as the males, begin laying eggs from the sixth day, continuing for a month to six weeks. Each female lays approximately from 10,000 to 15,000 eggs. The

embryos perforate the intestinal walls, pass into the circulation, and are hurried into all parts of the system. This period of infestation constitutes the first phase of the disease.

Askanazy, in 1896, suggested that it was not the embryos which perforated the intestinal walls and thus reached the blood-vessels, but the fertilised female trichinæ themselves, which entered the terminal chyle vessels and laid their eggs directly within them.

This observation is of great interest, for it contradicts the view held by Leuckart and proves that treatment is useless even in the first phase.

The males are about $\frac{1}{16}$ inch in length, the females $\frac{1}{8}$ inch to $\frac{5}{32}$ inch, and are ovoviparous.

Symptoms. The symptoms lack precise character, even when the disease is known to be developing, and moreover they have only been carefully observed in experimental cases. As soon as the laying period begins, signs of intestinal disturbance may be observed, possibly due to embryos perforating the intestinal walls (if we accept Leuckart's view), or, according to Askanazy, to adult females penetrating the chyle vessels and disturbing intestinal absorption.

These symptoms are only appreciable in cases of "massive" infestation. If slight, the disturbance passes unperceived. In severe cases the symptoms consist of diarrhoea, loss of appetite, grinding of the teeth, abdominal pain in the form of dull colic, and sometimes irritation of the peritoneum. The embryos carried by the circulation then escape into the tissues and, like the cysticerci, become encysted, preferably in the muscles, in the interfascicular connective tissue towards the ends of the bundles. Each (asexual) parasite plays the part of a foreign body, causing infiltration of serum and exudation of

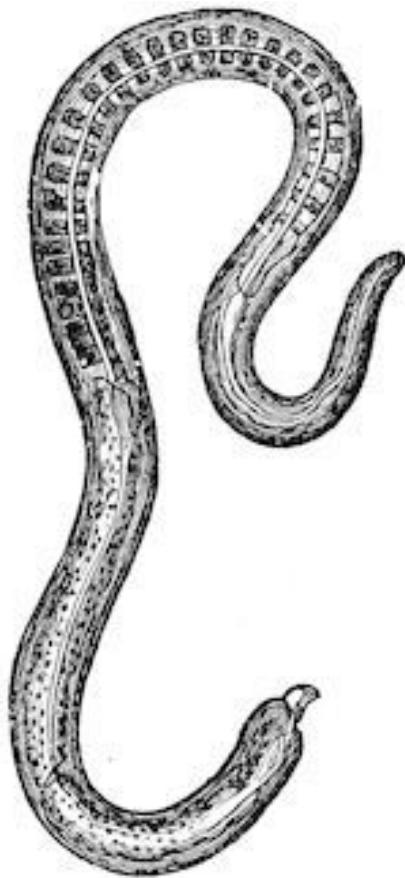


FIG. 46.—Male trichina from the intestine.
(Colin.)

leucocytes in its neighbourhood, and soon becoming encysted in the interior of a little ovoid space surrounded by a fibro-fatty wall. Fat granules accumulate at each end of the cyst.

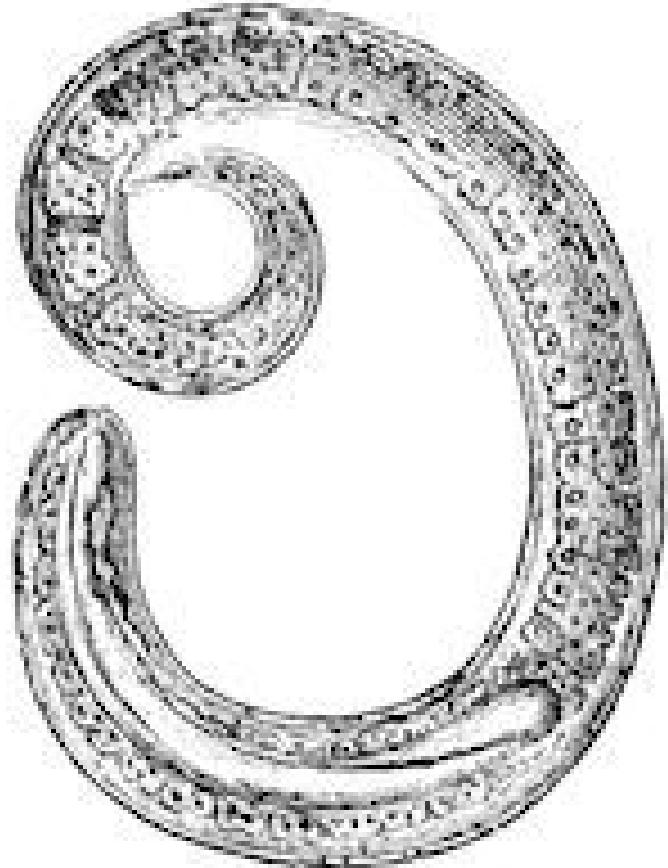


FIG. 47.—Free larval trichina. (Colin.)

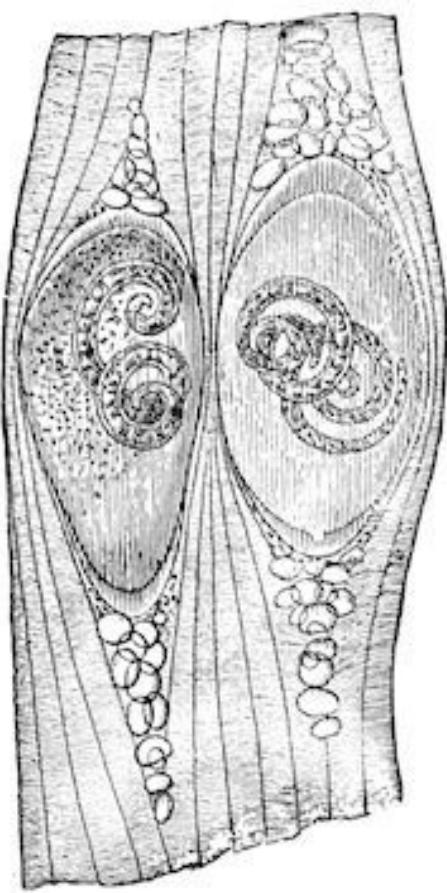


FIG. 48.—*Trichinæ*
encysted in the muscular
tissue. (Colin.)