

Nick Sullivan, RASPRG, IETF 120



LLMS AND RFCGPT

Leveraging large language
model platforms to
understand standards

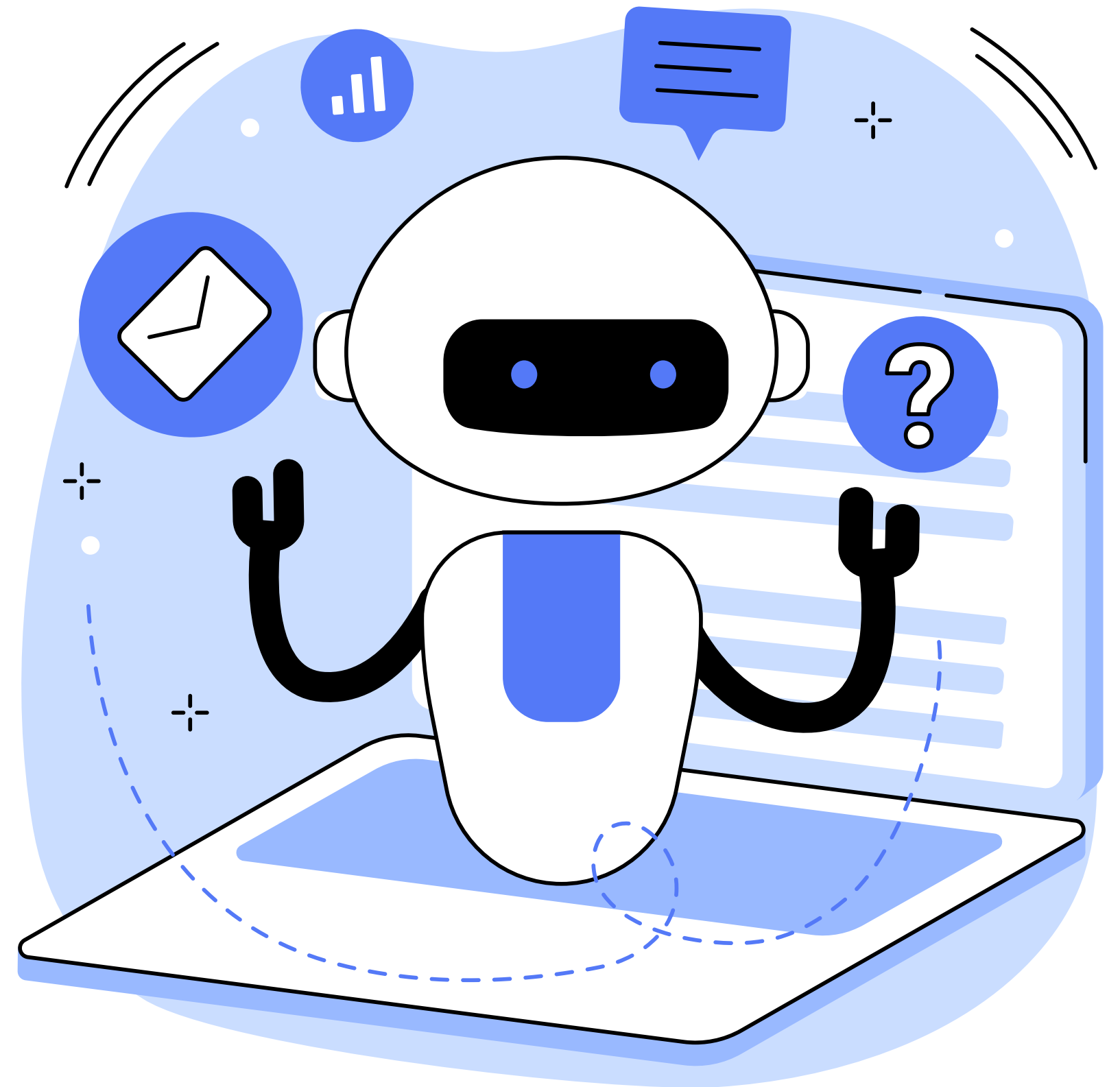
CONTENTS



- 1** What is a Large Language Model?
- 2** Overview of training and RAGs
- 3** Creating a custom GPT with OpenAI
- 4** Publishing RFCGPT
- 5** Outcomes and next steps

WHAT IS A LARGE LANGUAGE MODEL?

A Large Language Model (LLM) is a type of artificial intelligence algorithm designed to understand, generate, and interact with human language at a vast scale. Examples include models like GPT (Generative Pretrained Transformer).



LLMS IN ACTION

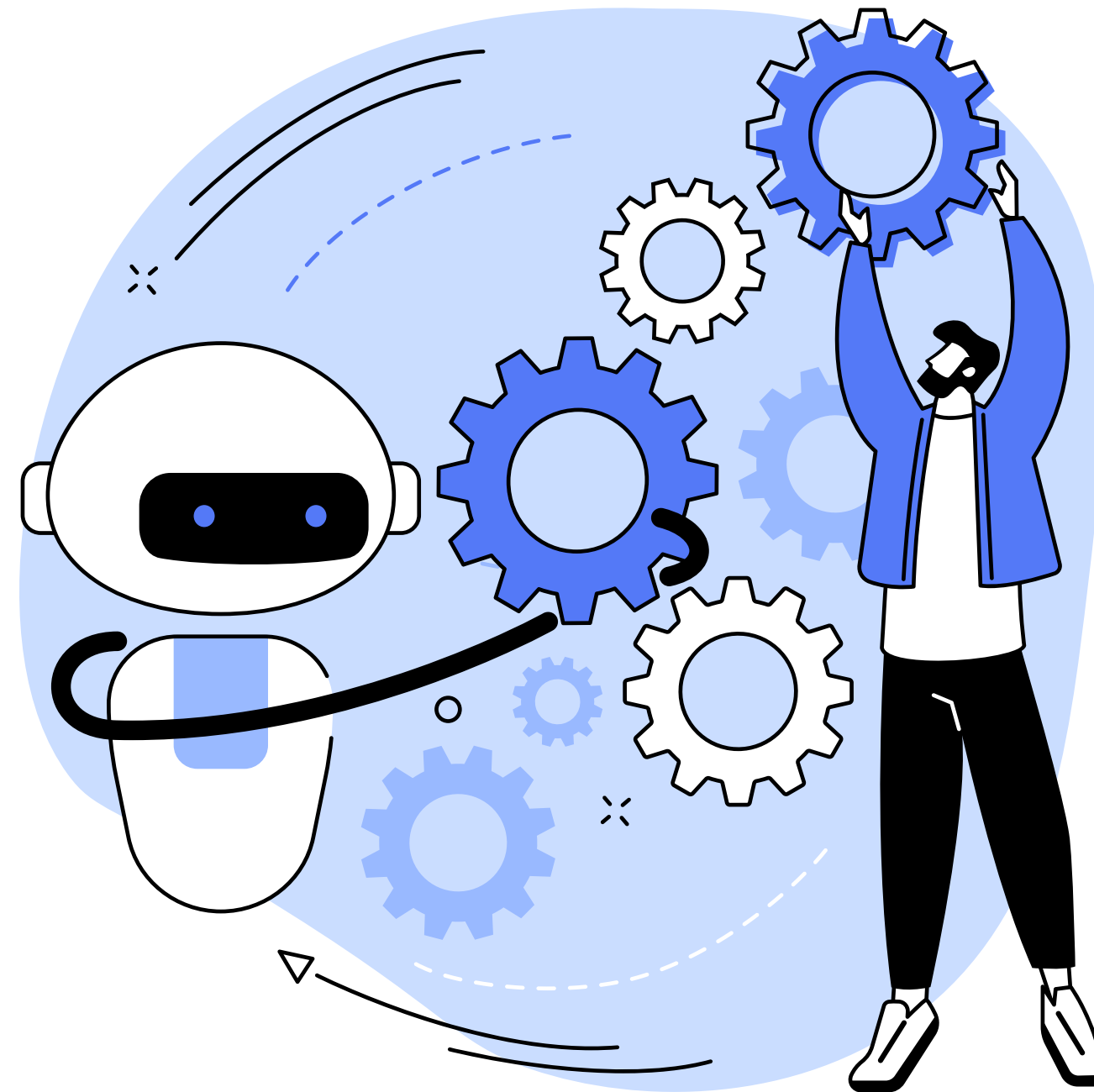
Applications

Smart Assistants

Text Classification

Text Generation

Expert Systems



Underlying Tech

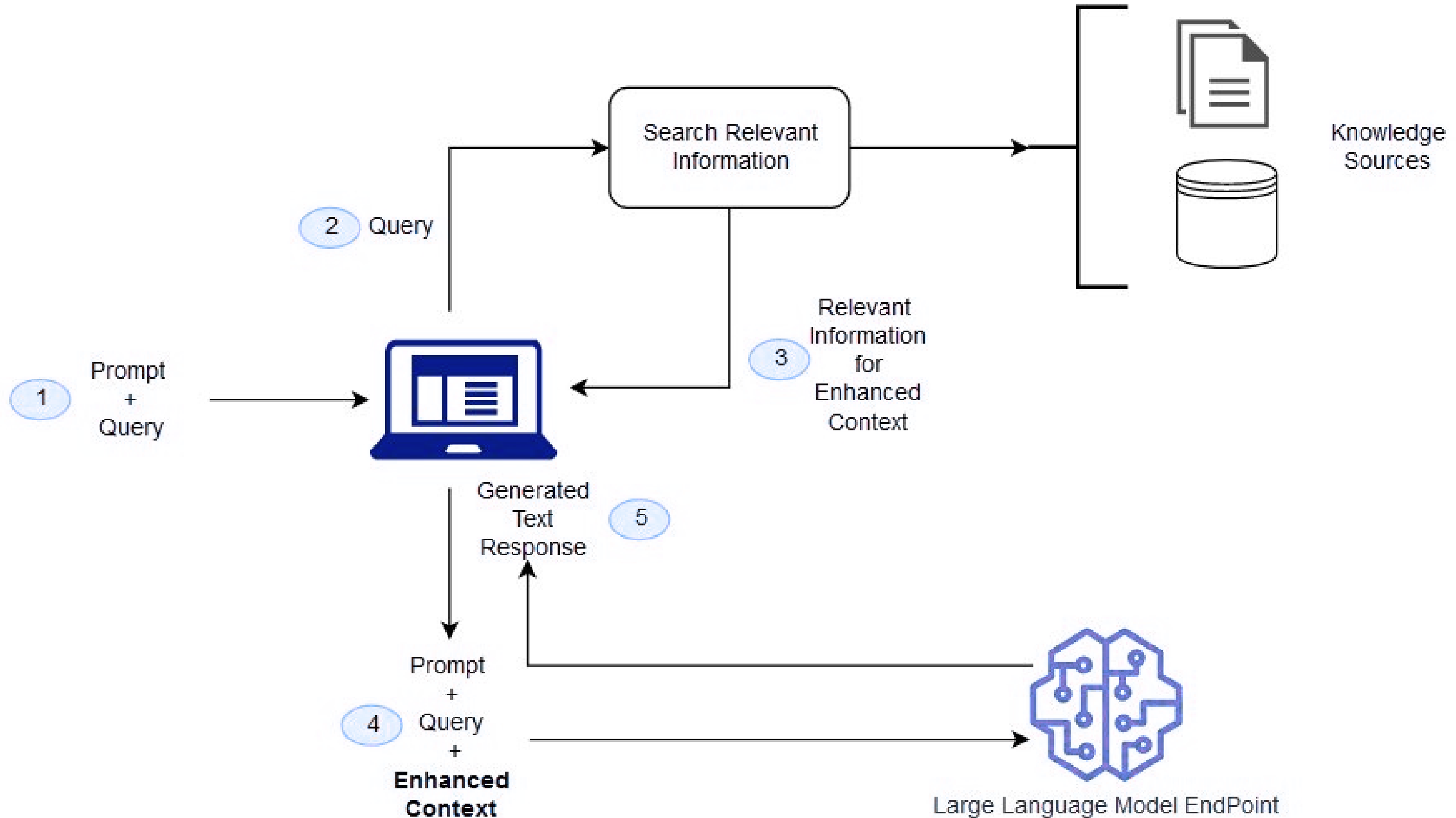
Training Data

Natural Language Processing

Models & Platforms

Retrieval-Augmented Generation

RETRIEVAL-AUGMENTED GENERATION



DESIGNING A RFC EXPERT SERVICE

Objective

Provide a user-friendly platform that simplifies the process of understanding and navigating the complexities of RFC documents.

Interaction Model

An intuitive interface that allows users to ask complex questions and receive precise, context-aware explanations and summaries of RFC content.

Base Model Requirement

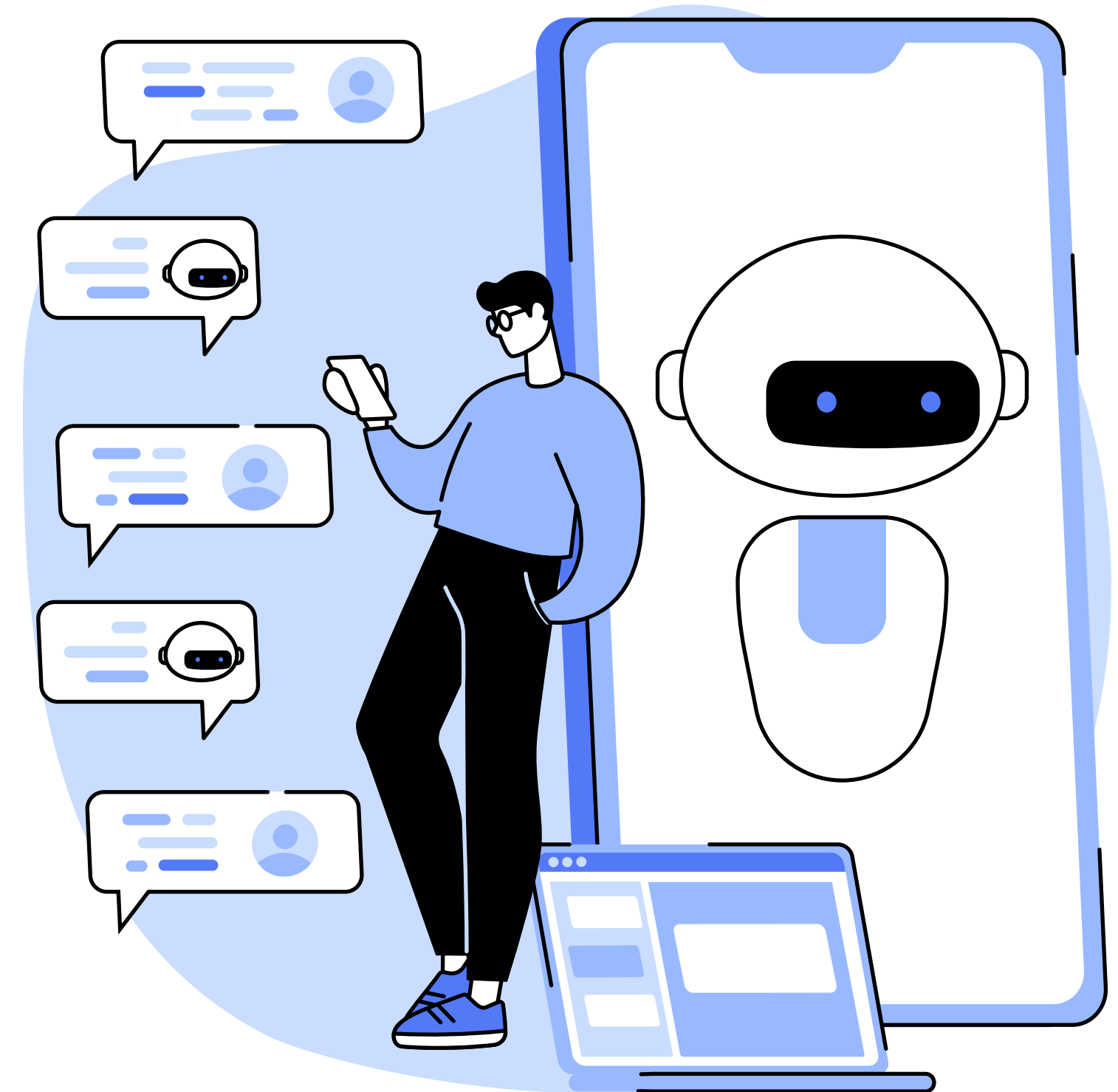
Enough information and context to be able to reason about broader technical topics and converse.

Domain-Specific Data

Augment GPT model on the RFC database to understand and interpret technical specifications, protocols, and standards accurately.

CREATING A CUSTOM GPT WITH OPENAI

The **My GPT** feature, allowing the creation of custom versions of ChatGPT for specific purposes, was announced by OpenAI on November 6, 2023. Engine: GPT4.



USING THE AVAILABLE FEATURES

Intructions

Custom context window.
Primes the chat with directions on how to answer.

Examples include

- Instructions on how to recover by searching the web.
- Where to find key information like author names

Knowledge

Configuration of RAG-style query enhancement. This is where the RFC archive can be uploaded.

Allows only

- UTF-8 text
- 20 files
- 2 million tokens per file
- 50MB per file

Code Interpreter

Feature by OpenAI that enables ChatGPT to write and execute Python code within a secure, sandboxed environment, facilitating tasks like solving math problems, performing data analysis, and more.

Enables large zip archives for search.

MAKING THE DATA FIT

Intructions

- Bias of underlying model is hard to avoid
- Additional instructions to prevent hallucination are ineffective

Knowledge

The RFC archive is full of text files that do not conform fully to UTF-8.

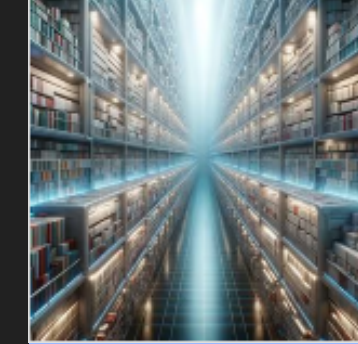
- Errors are not forthcoming.
- Pre-processing results in incomplete tokens.

50MB files failed upload. Split RFCs into groups of 100 (approx 5MB): only 2000 RFCs


Code Interpreter

Awkward python code created on the fly to parse archive and retrieve individual files. Frequent failures.

ZIP feature disabled later after deployment



RFCGPT

By cryptography.consulting 

RFC expert service

What does RFC 2616 cover?

Details about RFC 768, please.

Can you explain RFC 2549?

Who is the author of RFC 791?

 Message RFCGPT...



**SOFT LAUNCH
NOVEMBER '23**

**CHATGPT+ ONLY
400+ CONVOS**

EARLY RESULTS

Feedback:

“It was very useful. In fact we discovered a decoding bug in native ModSecurity and your tool helped us defining the correct behavior.”



You

SRTP について教えてください。



RFCGPT

The Secure Real-time Transport Protocol (SRTP) is defined in several Request for Comments (RFC) documents. Here are some key RFCs related to SRTP:

1. **RFC 3711:** "The Secure Real-time Transport Protocol (SRTP)" by M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, published in March 2004. This RFC outlines the core specifications of SRTP. It has been updated by RFCs 5506, 6904, and 9335.
2. **RFC 5763:** "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)" by J. Fischl, H. Tschofenig, E. Rescorla, published in May 2010. Updated by RFC 8842, this document provides a framework for establishing SRTP security contexts using DTLS.
3. **RFC 6904:** "Encryption of Header Extensions in the Secure Real-time Transport Protocol (SRTP)" by J. Lennox, published in April 2013. This RFC updates RFC 3711 and focuses on the encryption of header extensions in SRTP.
4. **RFC 7714:** "AES-GCM Authenticated Encryption in the Secure Real-time Transport Protocol (SRTP)" by D. McGrew, K. Igoe, published in December 2015. It describes the use of AES-GCM with SRTP for authenticated encryption.
5. **RFC 7983:** "Multiplexing Scheme Updates for Secure Real-time Transport Protocol (SRTP) Extension for Datagram Transport Layer Security (DTLS)" by M. Petit-Huguenin, G. Salgueiro, published in September 2016. It updates RFC 5764 and discusses the multiplexing schemes for SRTP and DTLS.

SRTP is designed to provide encryption, message authentication, and integrity verification for the Real-time Transport Protocol (RTP) and RTP Control Protocol (RTCP) streams. The protocol is widely used in VoIP (Voice over IP) and multimedia streaming applications to ensure secure communication.



SUCCESS? NOT YET

The limitations of the My GPT platform default features limits its current usefulness.



FUTURE DIRECTIONS

Using an External RAG

Set up an external vector database with the tokenized RFC archive. Automate addition of new RFCs when published.

Expanded Training Data

Add mailing list data and datatracker history to the archive.

Prompt engineering to enable tool to “think like a chair”

Open Source Models and Fine-tuning

Leverage open source models like LLaMa (Meta) and fine-tuning and run independently of OpenAI

Quality Measurement

Set up automated tests to measure response quality and debug errors.



THANK YOU FOR LISTENING!