Using ChatGPT API in Applications

Understanding Large language Models (LLM)

# Cover

Hello everyone. My name is Konstantin Voloshenko, and I am a BA from St. Petersburg.

One of my hobbies is neural networks.

Last time I told you about neural networks from three sides: how it looks for a PM, for a BA, and for a developer.

<https://bass.netcracker.com/display/VID/Neural+Networks+Overview>

Today, I will talk about how you can integrate ChatGPT into your application.

# Agenda

I will provide an overview of what ChatGPT is and how it works, and then I will explain how you can use the ChatGPT API to incorporate it into your software.

# Warning!

Please do not send our Сompany inside information to ChatGPT.

It will be a security incident!

# Overview. What is and how ChatGPT works

ChatGPT is a language model developed by OpenAI. It uses a technique called Generative Pre-trained Transformer (GPT) to generate text responses that resemble human speech. This model has been trained on a large amount of text from the internet, allowing it to learn patterns, grammar, and even some understanding of various topics. It operates in a conversational way, where users can input prompts or questions, and ChatGPT generates a relevant response.

# Step 1: Simple Request. Using ChatGPT API in Applications

First of all, in order to integrate with the language model, we should find a suitable API. Yes, there is such API for ChatGPT.

Let me explain how you can interact with GPT through the API. First, we will send a message to the model and ask for a response. ChatGPT uses the concept of "roles" to represent different participants in the conversation. Each message in the messages list is like a dictionary with two parts: role and content. Role can be one of three values: "system", "user", or "assistant", which tells us who is speaking the message. Content contains the actual message. In a typical chat, there are three main roles:

• "**system**": The system role is used to set the context of the conversation. It gives general instructions to the model, telling it how to behave during the conversation. System messages are usually placed at the beginning and may provide information on how the model should act as an assistant.

• "**user**": This role represents the human user interacting with the model. Your requests to the model will be under this role.

• "**assistant**": This role represents the AI model that responds to the user. When you create a chat with the model, you provide a list of messages.

See the full API reference documentation here

<https://platform.openai.com/docs/guides/gpt/chat-completions-ap>

# Step 2: Prompt and Knowledge base

Secondly, we should decide how to send exactly our actual data to the language model. So, we will use the trained language model as a transformer, transforming our data into the necessary response form.

To do this, we prepare files with a prompt and a knowledge base in advance.

## Prompt

Guidelines for creating prompts for the ChatGPT model:

1. **Be specific:** The more specific your instructions, the more accurate answer you will receive. Make sure your instructions are clear. If the instructions are too general or unclear, the model may struggle to understand what you want from it.
2. Use the system message **to outline the behavior of the model**. For example, you can specify in the system message: **"You are an assistant with expertise in art history, always striving to provide detailed and accurate answers."** This message helps set the context for the model. The system message allows you to assign tasks or give instructions to the model. The model sees this message, but it is not considered part of the conversation with the user.

## Knowledge base

The structure of the company's knowledge base plays a vital role in providing responses to client inquiries in written form. The primary goal of the knowledge base structure is to ensure its effective functioning. Here are some important principles to consider when organizing the structure:

1. **Categorization and classification**: The knowledge base should be divided into clearly defined and logically organized sections that correspond to different knowledge areas or types of client questions. For instance, there can be sections related to products, services, support, payment inquiries, and more. Each section can contain subsections for better organization and detail.
2. **Hierarchy**: It is helpful to arrange the knowledge base in a hierarchical structure, with broader topics at the top level and more specific questions and answers at lower levels. This simplifies the management of information relevance and allows for easy updates when necessary.

By these principles, we can ensure that the knowledge base is designed to be organized, logical, and model - friendly.

However, we cannot transfer a large knowledge base to the language model -

The model has restriction on the input message size.

And this model is not available to us for re-training.

So, we need to find a way to split the knowledge base into chunks, find the most relevant to the question chunks, and then submit only those relevant chunks to the language model.

## Main Questions

1. How to find relevant chunks?
2. What is Embedding?
3. How does vector search work?

Embeddings. In the context of language models, embeddings are numerical representations of words. They are created by mapping words to vectors in a mathematical space. These vectors capture the meaning or semantic relationships between words.

<https://bass.netcracker.com/display/YAAIML/GenAI+Useful+Information>

<https://towardsdatascience.com/all-you-need-to-know-to-build-your-first-llm-app-eb982c78ffac>

# Step 3: TG bot and all together

Thirdly, we need to integrate interaction with the language model into our application.

To do this, the easiest way is to use Telegram and create a TGbot that interacts according to our principles with ChatGPT.

The diagram shows the flow of information and the interaction of various components.

1. The Application downloads two files from Google Drive: one file contains prompt, and the other contains a knowledge base.
2. The Application splits the knowledge base into chunks using a CharacterTextSplitter component.
3. The Application creates an index database and converts the knowledge base fragments into embeddings.
4. The diagram shows a group labeled "async text(update, context) # TG bot function for text messages" which represents the asynchronous process of handling text messages in the Telegram Bot. Within this group, the following steps occur:

* The User sends a question to the Telegram Bot.
* The Application receives the user's question.
* The Application searches for relevant chunks in the knowledge base based on the user's question.
* The Application sends a request to ChatGPT, including the prompt, message content (question and relevant chunks), and a temperature parameter controlling the randomness of the response.
* ChatGPT generates a response and sends it back to the Application.

1. The response from ChatGPT is then sent back

**Embeddings.** In the context of language models, embeddings are numerical representations of words. They are created by mapping words to vectors in a mathematical space. These vectors capture the meaning or semantic relationships between words.

## Bot question examples

who are you?

price?

any discounts?

can I bring alcohol?

What about smoking?

how r u

all your visitors are idiots

you are a fool

what games do you have

que juegos tienes

|  |  |
| --- | --- |
| Question | Type |
| who are you? | General |
| price? | Content |
| any discounts? | Content |
| can I bring alcohol? | Content |
| What about smoking? | Content |
| how r u | General |
| all your visitors are idiots | Stress resistance |
| you are a fool | Stress resistance |
| what games do you have | Content |
| que juegos tienes | Another language |

# Summary

Summary

1. The ChatGPT API is a crucial aspect of the integrated application.
2. It is necessary to have your own Knowledge Base.
3. Telegram bot integration is the easiest approach.
4. Apart from ChatGPT, there are other LLM options available, with the most well-known being the Llama models family.

# Additional materials

## Github reference to NC\_Attic branch

<https://github.com/kvoloshenko/LLMT_01/tree/NC_Attic>

## Structure of the .env file:

TOKEN = '???' # TG bot token

API\_KEY = '???' # Open AI API Key

## Links to openai documentation:

Chat Completions API:

<https://platform.openai.com/docs/guides/gpt/chat-completions-api>

Chat completions response format:

<https://platform.openai.com/docs/guides/gpt/chat-completions-response-format>

## Token Limit in GPT-3.5

see <https://www.scriptbyai.com/token-limit-openai-chatgpt/>

|  |  |
| --- | --- |
| Model | Max Tokens |
| gpt-3.5-turbo | 4086 |
| gpt-3.5-turbo-0613 | 4086 |
| gpt-3.5-turbo-16k | 16384 |

# Meeting info

1. Topic and Agenda

Using ChatGPT API in Applications

Understanding Large language Models (LLM)

* Overview
* Step 1: Simple Request
* Step 2: Prompt and Knowledge base
  + Prompt
  + Knowledge base
* Step 3: TG bot and all together
* Summary
* Q&A

2. Meeting description (not agenda!): In this meeting you will know about/how to....

On the example of ChatGPT you will learn how to integrate the Large Language Model (LLM) into your application.

3. Your NC experience (3-4 sentences)

Previous video <https://bass.netcracker.com/display/VID/Neural+Networks+Overview>

Konstantin Voloshenko, BA from St. Petersburg. One of my hobbies is Artificial Intelligence (AI) and Neural Networks (NN).

4. Follow up materials with links - link to the deep dive into the topic. Bass spaces, articles, useful resources.

My sources <https://github.com/kvoloshenko/LLMT_01/tree/NC_Attic>

Links to Open AI documentation:

Chat Completions API: <https://platform.openai.com/docs/guides/gpt/chat-completions-api>

Chat completions response format: <https://platform.openai.com/docs/guides/gpt/chat-completions-response-format>