Blog

Product 1 min read

Introducing Usage Metering for ChatGPT-Powered LLM Apps

Add usage metering to your ChatGPT-powered products with five lines of code. Quickly implement functionality to enforce limits, detect abuse, act on user feedback, and visualize usage insights and get to market faster.

Nico Acosta

• Published: May 17, 2023
• Last updated: May 25, 2023

Photo: Propel

Today, we are excited to announce the preview of Propel's usage metering solution for ChatGPT-powered applications. It enables developers to streamline building and operating usage metering functionality, such as token usage tracking, latency monitoring, and user feedback collection, through a highly scalable analytics backend and a blazing-fast data API.

With Propel, developers can quickly implement functionality to enforce limits, detect abuse, act on user feedback, and visualize usage insights in their user-facing product or internal tools. The preview supports OpenAI's ChatGPT API (GPT-3.5 and GPT-4). We plan to add support for more APIs and custom-built LLMs based on customer demand in the future. To get access, join our waitlist.

Large Language Models (LLMs), such as ChatGPT, have been the biggest advance in computing since the graphical user interface. They fundamentally change how we interact with computers. LLM apps are being built into every consumer or B2B product experience. However, taking ChatGPT-powered products to market is not an easy task.

The Problem: Generative AI Apps Need to Track Usage

Each ChatGPT API call consumes tokens that cost money. Therefore, usage needs to be tracked and communicated to end users. It must be designed and built into the product experience. With all the excitement around LLMs, prompt design and tuning, preventing prompt injections, and model benchmarking, tracking usage is often an afterthought. However, it's a critical aspect that can cause delays in bringing generative AI products to market.

There are different ways to productize GPT-powered features. Some products will include GPT-powered features as part of their current offerings. In such cases, they need to keep track of usage and enforce limits to avoid runaway costs. Other products will pass the cost onto the end user. In such cases, they need to keep track of usage to be able to charge at the end of the month. In either case, usage metrics must be available via APIs for developers to surface to end-users, and to integrate into the rest of their application logic that enforces limits, bills, and sends user notifications.

Why Is It Hard to Track Usage?

Tracking usage is harder than it seems. Why not just store it in the database and read it from there? There are three main challenges:

1. **Aggregates and query latency:** Usage queries are analytical in nature, meaning they do not retrieve individual records from a database. Instead, they retrieve a set of records for a given time range based on a given aggregation. For example, your customers may want to know their token consumption by day, broken down by user, for the last 30 days. This query requires a SUM aggregation over the last 30 days, grouped by user, and filtered by customer ID. As usage data grows, your transactional database will become increasingly slower at serving these queries, which can slow down your entire application.
2. **Ever-evolving metadata:** Your product is constantly evolving, with new GPT-powered features being shipped to different parts of the product. This means that the metadata contextualizing usage data is constantly changing. The nature of this data is unstructured, so you will need the

ability quickly add new metadata and query your usage data with the ability to slice and dice by the metadata you provided.

3. **Custom metrics:** Every product is unique, and how you account for usage and productize your GPT features will depend on your product, customer base, and the experience you want to provide. Therefore, you need the ability to define custom metrics to track your primary usage unit. Some products track usage based on requests, while others track usage on input words or input tokens. Whatever method you choose, you need a centralized metric definition that you define once and use everywhere.

Propel's Usage Metering for LLM Apps

Propel's LLM usage metering solution equips generative AI developers with a powerful analytics backend, low-latency data API, and a React component library to easily capture usage data, model it into metrics, and expose usage data via a fast and responsive GraphQL API. This API can be used to drive product behaviors, such as enforcing limits or usage accounting or building in-product usage visualizations like reports and dashboards.

Key Features

The solution consists of the following features:

Data collection endpoint

The data collection endpoint enables developers to send the response of the OpenAI API directly to Propel with a simple HTTP POST.

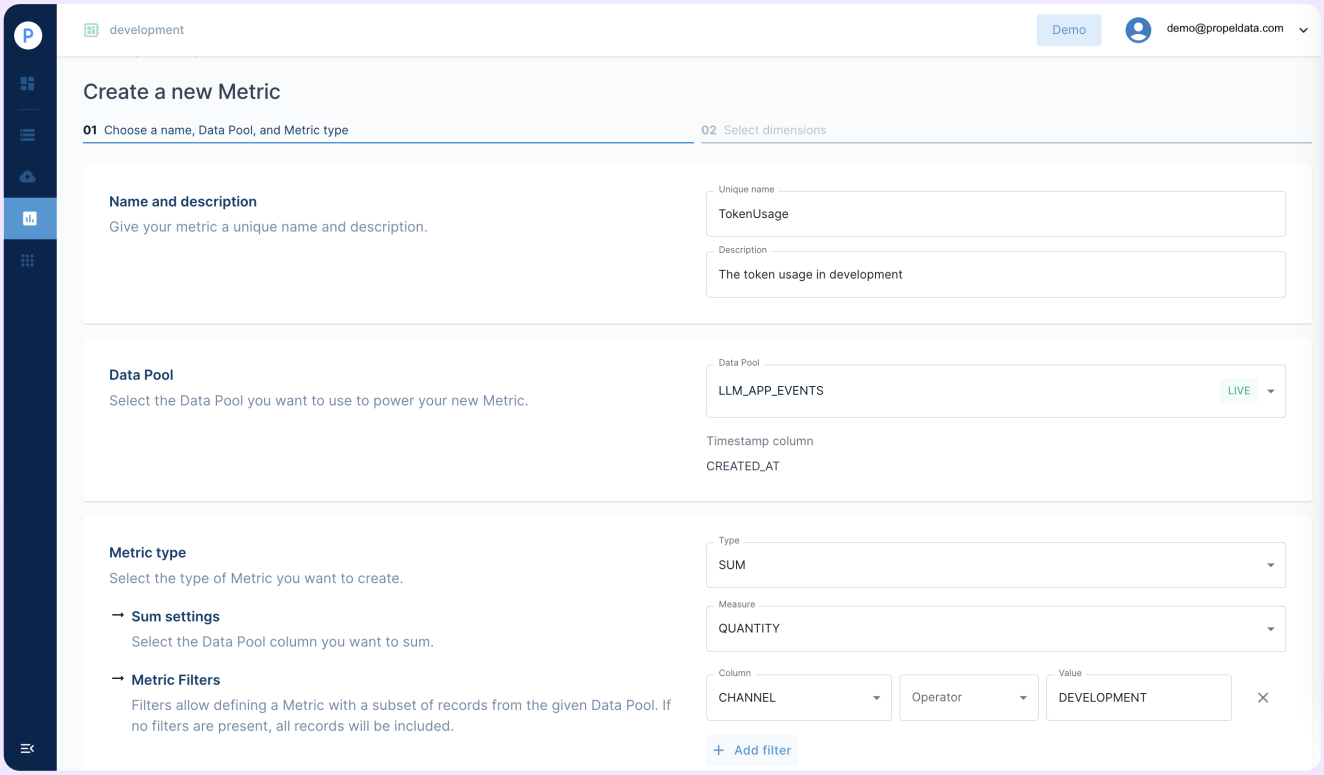
```
const openAICompletion = await openai.createChatCompletion({
  model: "gpt-4",
  messages: [{role: "user", content: "Hello world"}],
});

// Log response with Propel
const logResponse = await fetch("PROPEL_URL", {
    method: "POST",
    headers: { "Authorization": "Bearer YOUR_TOKEN"},
```

```
    body: JSON.stringify(openAICompletion),
});
```

Define custom metrics

Once the data is in Propel, developers can define custom metrics that give meaning to the data. The example below illustrates how to configure a `TokenUsage` metric.



Query data via a GraphQL API

Once the Metric is defined, developers can query the data with a simple GraphQL API request. The example below queries the token usage metric when the `gpt-4` was used over the last 30 days.

```
query TokenUsage {
  metricByName (uniqueName: "TokenUsage") {
    counter ({
      timeRange: { relative: "LAST_N_DAYS"  n:"30"}
      filters: [{
```

```
        column: "model"
        operator: EQUALS
        value: "gpt-4"
      }]
    }) {
      values
      labels
    }
  }
}
```

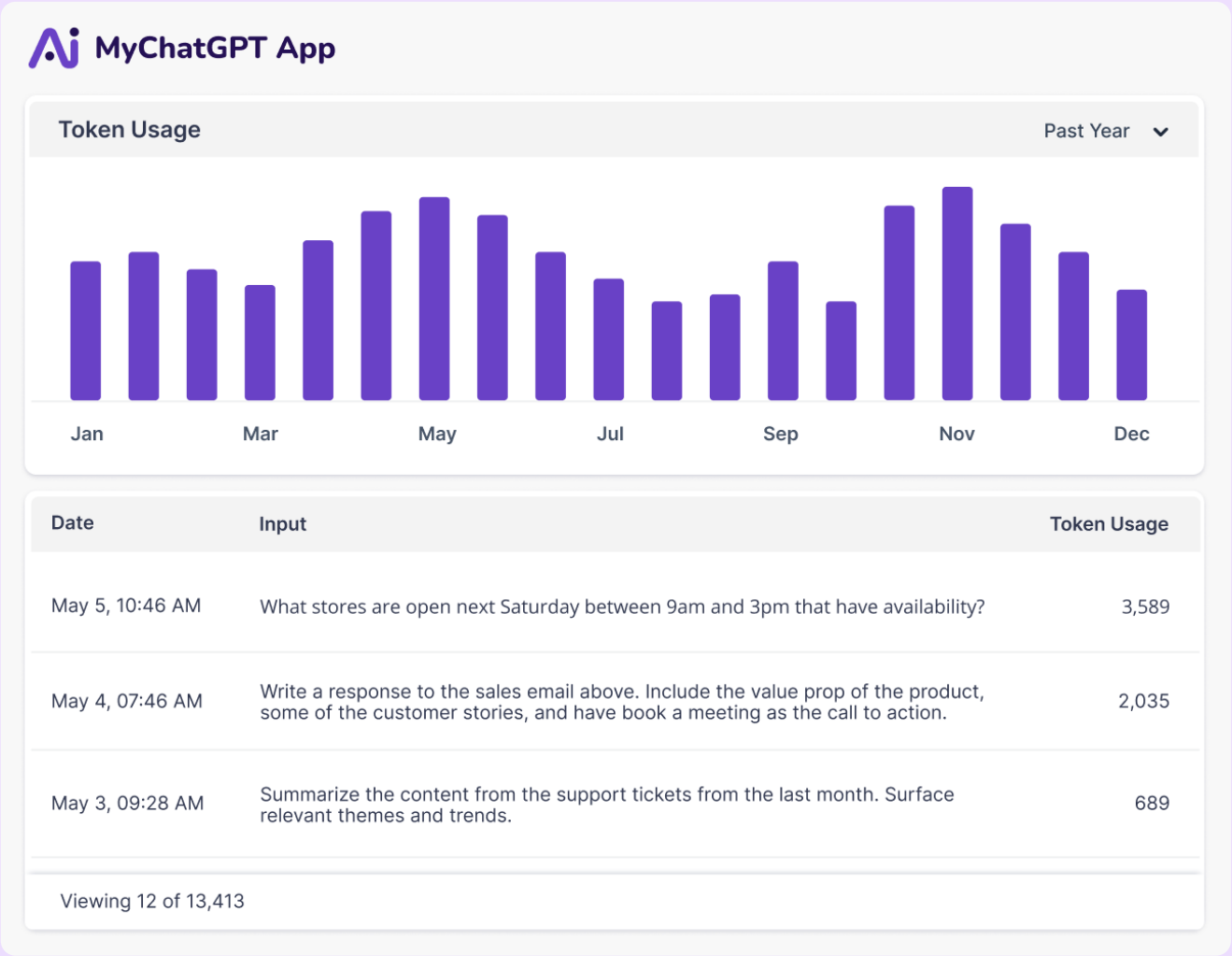
Slice and dice on your own metadata

If you added custom metadata to the payload, you can slice and dice the metric values by your own metadata. The example below adds a filter based on a `customerId` value provided.

```
query TokenUsage {
  sales: metricByName (uniqueName: "tokenUsage") {
    timeSeries ({
      timeRange: { relative: "LAST_N_DAYS"  n:"30"}
      granularity: "DAY"
      filters: [{
        column: "metadata.customerId"
        operator: EQUALS
        value: "ac5bc654-6449-4003-ac72-6f07e761b152"
      }]
    }) {
      values
    }
  }
}
```

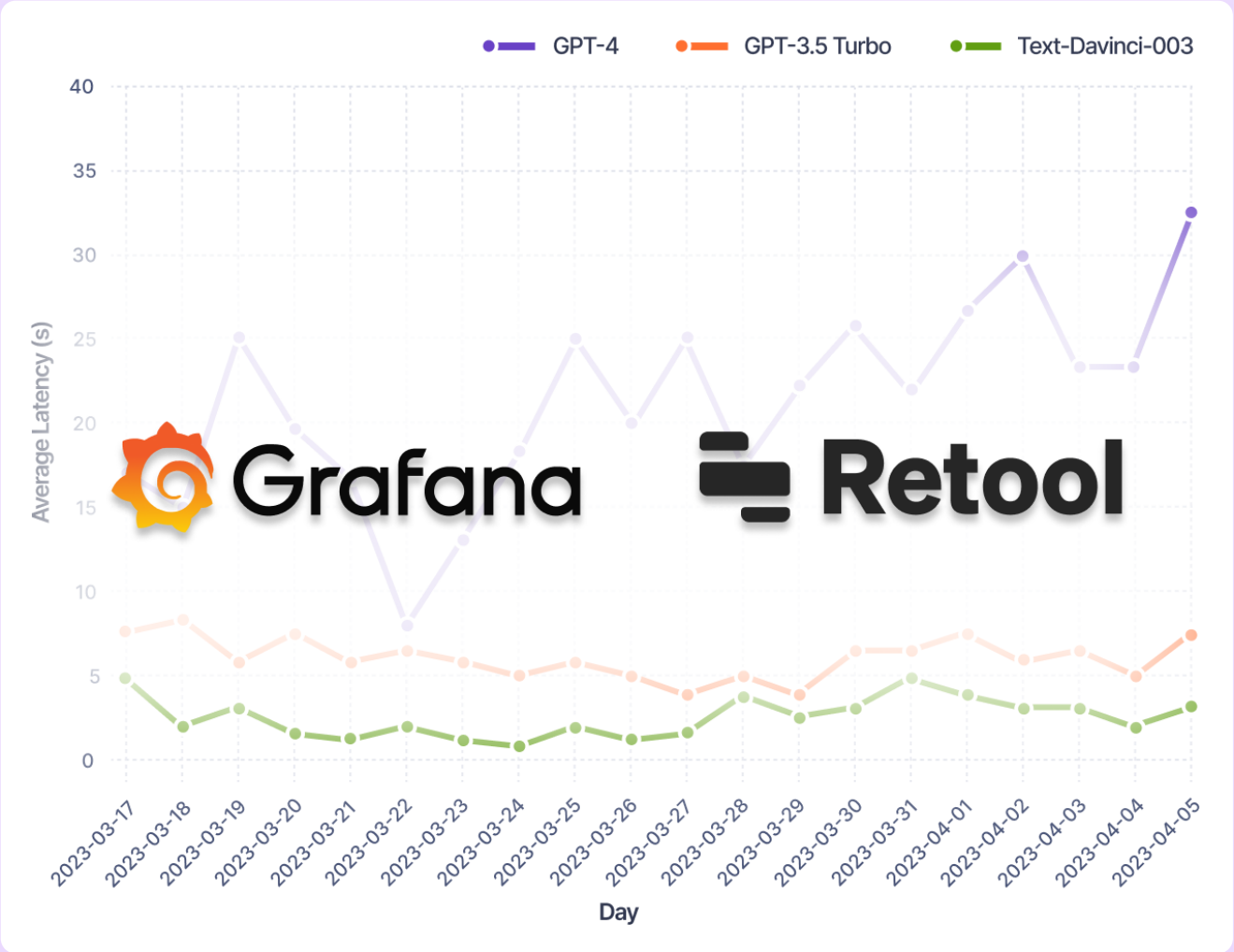
Build user-facing usage dashboards

By querying the API and using our UI components (optional), you can quickly build dashboards, log interfaces, and in-product metrics.



Integrate's with the observability tools you already use

Lastly, you can easily integrate with your existing observability and internal tools stack, such as Grafana and Retool to manage and monitor your ChatGPT-powered apps. No need to reinvent the wheel.



Final thoughts

With Propel's usage metering solution, developers can focus on what they do best - building great ChatGPT-powered products - without having to worry about the complexities of usage tracking. By streamlining the development and operation of usage metering functionality, Propel is helping to bring ChatGPT-powered products to market faster and more efficiently than ever before.

Join our waitlist now and start getting the benefits of Propel's usage metering solution for ChatGPT-powered applications.

Nico Acosta
Co-founder and CEO at Propel

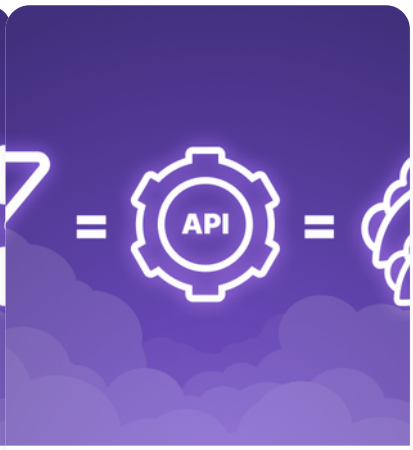
Related posts

Product

JSON Data Type and Query support

Product

Introducing new synchronization controls for Snowflake data warehouse integrations



Product

Introducing OR and Filter Group Support to our GraphQL API: Build powerful filters for your reports

Dave Ganly
June 23, 2023

Dave Ganly
May 31, 2023

and dashboards

Dave Ganly
May 1, 2023

Return to blog

You could be building more

Get a product demo to see how Propel helps your product dev team build more with less.

Try for free | Book Demo

Propel

Company

- Product
- Pricing
- About
- Contact
- Legal
- Snowflake

Resources

- Docs
- Changelog
- Blog
- Case Studies
- Podcasts

Follow us

- Twitter
- LinkedIn
- Github
- Reddit