Credal Take-home Engineering Screen: PII Redaction

Time allotted: 120 mins.

Introduction

At Credal, one thing we do is help prevent sensitive data from being exposed to an LLM. For example, if an Enterprise wants to prevent Social Security Numbers (SSN) from ever being sent to an LLM, they route their LLM request through our API, we automatically detect when the prompt contains an SSN, and “redact” the SSN by replacing the SSN in the prompt with placeholder text, e.g. “SSN_0001”.

However, to prevent disrupting the user experience too much, if the LLM response contains the placeholder text “SSN_0001”, we want to “unredact” this response: i.e. replace that placeholder with the original SSN in the response we show the user.

Your job is to take incoming user requests, redact sensitive content before generating a chat completion, then reconstruct redacted data.

We estimate that this should take you around 70 minutes, but you have the full 120!

Given

You can complete the project in your language of choice. We use TypeScript and Python at Credal.

You can assume code that detects SSNs, emails and phone numbers in strings and returns their positions. A scaffold in Python is attached.

You have been given a CSV of incoming user requests. Below is the schema for user requests as a notional TypeScript type,

```
type UserRequest: {
    system_prompt: string; // instructions and context retrieved from RAG pipeli
ne
    prompt: string; // task user is requesting
}
```

What your code needs to do

1. **Redact Incoming Requests**

Redact all sensitive data. Below are the entities we care about redacting in the scope of this problem:

```
ENTITIES = [
    "US_SSN",
    "PHONE_NUMBER",
    "EMAIL_ADDRESS",
]
```

For example, if the user request is

```
{
    'system_prompt': 'You are helping a user with tax filing. Their SSN is 123-45
-6789.',
    'prompt': 'Hi, my email is john.doe@email.com and my phone number is (55
5) 123-4567. Can you help me file my taxes?'
}
```

The redacted context would then be:

```
{
    'system_prompt': 'You are helping a user with tax filing. Their SSN is SSN_0
001. ',
    'prompt': 'Hi, my email is EMAIL_1 and my phone number is PHONE_NUMBE
```

```
R_0001. Can you help me file my taxes?'
}
```

Note that there may be more than 1 of each entity in the text, it is up to you to distinguish between them.

We recommend using the presidio_analyzer python library for PII redactions, however, you may use Regex if you prefer (or if you're using node).

2. **Make the Streaming LLM Requests**

You will have to use the OpenAI (or any provider of your choice) API to make those requests. Your final submission should use **streaming**. We recommend that you find a solution without streaming enabled, and then add the functionality later.

3. **Unredact the Reponse:** Given the list of redactions and LLM response, construct the un-redacted user-facing text.

If streaming is enabled, you will be given chunks of the response at a time, it is up to you to figure out a way to never return a partially redacted string. To the user, it should feel as if none of this were happening behind the scenes! To simulate streaming in the "frontend," print chunks or partially completed responses in the console. This will help you (and us) visualize your work.

Implement all necessary functions to complete the above tasks. The test cases provided are not exhaustive, so please include any additional test cases or validation steps you took.

As time permits, please provide explanations for your design decisions either in the code or in a separate document. Include this document in your repository. This will help us understand your thought process, although it is completely optional!

How to submit

(Don’t worry about the minuscule differences between submitting at 108 minutes or 113 minutes)

Zip your project and reply in the email thread, or send to hiring@credal.ai

That’s it! Thank you so much for taking the time to redact sensitive information. 🙂