LLM Text Masking to Protect Sensitive Data

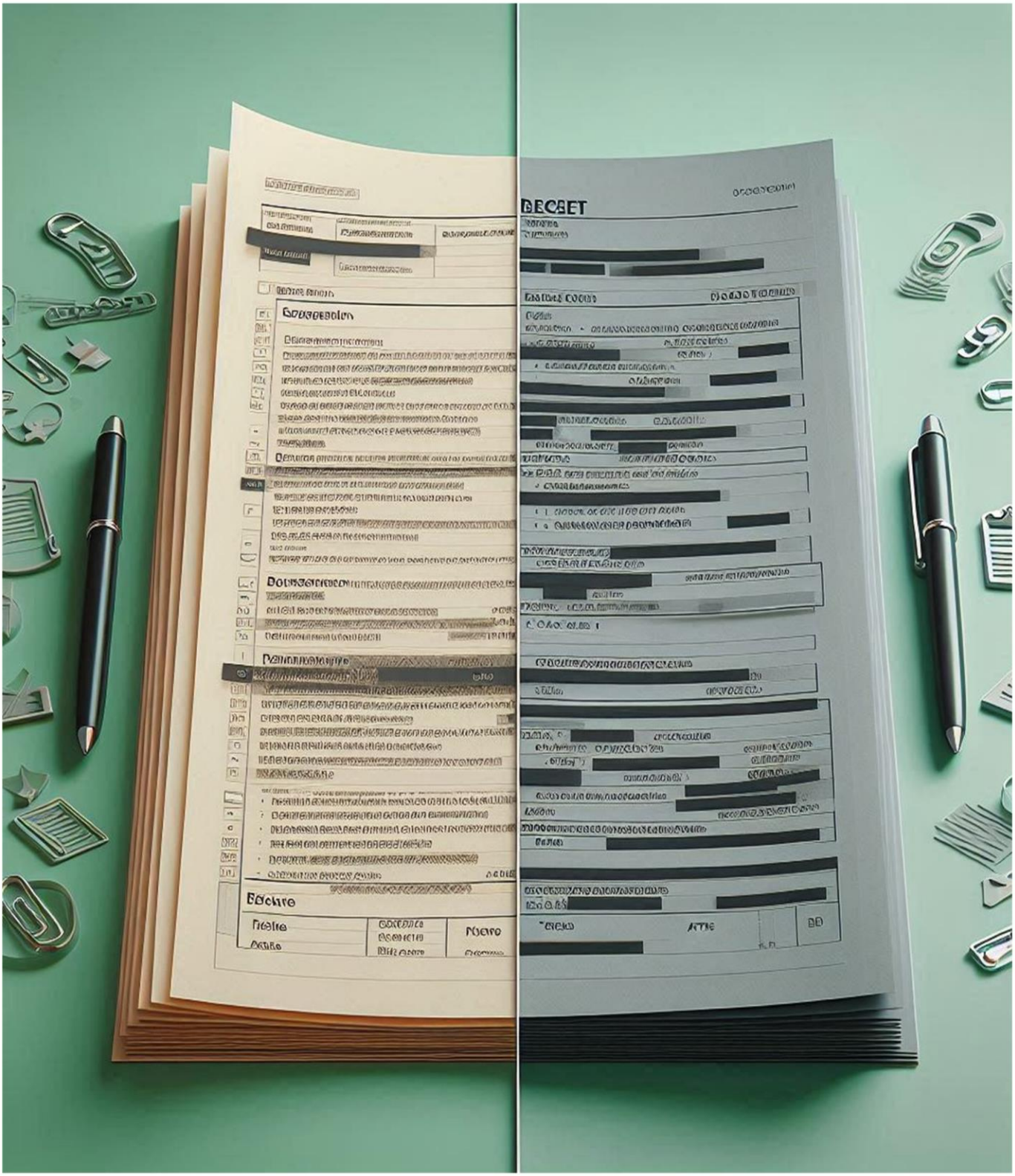


Table of Contents

1. Introduction

Large Unstructured documents often contain a mix of relevant and irrelevant information.

When working with sensitive data, it's crucial to mask or remove unwanted text to protect privacy or streamline content for analysis.

Large Language Models (LLMs) like GPT-3.5 can be effectively used to identify and mask these unwanted text segments in an automated fashion.

2. Understanding Masking and its Importance

2.1 What is Masking?

Masking refers to the process of obscuring or removing certain parts of a text.

This is particularly important in scenarios like data privacy, where sensitive information (e.g., personal identifiers, confidential details) needs to be hidden.

2.2 Why Use LLMs for Masking?

Traditional methods for masking involve regular expressions or rule-based systems, which can be brittle and hard to scale.

LLMs, on the other hand, can understand the context and meaning behind text, making them more effective at identifying which parts of the text should be masked.

3. Setting Up Your Environment

Before we dive into the code, let's set up the environment.

```
# Install required libraries

!pip install openai
!pip install pandas
```

We'll be using OpenAI's GPT model for this tutorial, so ensure you have an API key from OpenAI.

If you haven't installed it yet, you can do so using the following command:

```
import openai
import pandas as pd

# Set your API key
openai.api_key = 'your-openai-api-key'
```

4. Using LLMs to Identify and Mask Unwanted Text

Let's start by loading a sample unstructured document.

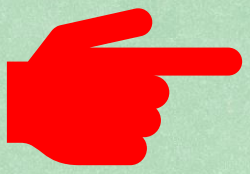
```
# Sample unstructured document
document = """
Dear John,

Your social security number is 123-45-
6789. Please keep it safe.

Also, note that your appointment with
Dr. Smith is scheduled for tomorrow at
10 AM.

Best regards,
Jane Doe
"""
```

Now, we'll use an LLM to identify and mask sensitive information.

This function sends a prompt to the LLM asking it to identify and mask sensitive information.

The model returns the modified document with sensitive information replaced by the mask token.

Next page



```
def mask_unwanted_text(document,
                       mask_token="[MASK]"):
    prompt = [
        {"role": "system", "content": "You
         are a helpful assistant that masks
         sensitive information in text."},

        {"role": "user", "content": f"Please
        mask any sensitive information like
        name, phone numbers, email addresses,
        or credit card numbers in the
        following text, replacing them with
        '{mask_token}':\n\n{document}"}
    ]

    response = openai.ChatCompletion.create(
       model="gpt-3.5-turbo", # or another suitable model
       messages=prompt
    )

    Return response.choices[0].message['content'].strip()

masked_document = mask_unwanted_text(document)
print(masked_document)
```

5. Customizing Masking with Fine-Tuning

For more specialized tasks, you might want to fine-tune an LLM on your specific dataset.

Here's a high-level overview of how you could approach this:

1. **Collect and Label Data:** Gather a dataset of documents with the text you want to mask labeled accordingly.

2. **Fine-Tune the LLM:** Use a platform like Hugging Face or OpenAI's fine-tuning API to train the model on your specific masking task.

3. **Deploy and Test:** Use the fine-tuned model to mask unwanted text in your documents.

Fine-tuning can greatly improve the accuracy of your masking, especially for industry-specific documents.

6. Best Practices and Considerations

- **Contextual Awareness:** Ensure that the model understands the context of the document. Overzealous masking might remove important information.

- **Testing and Validation:** Always validate the output manually or through automated tests to ensure the correct text is being masked.

- **Ethical Considerations:** Be mindful of privacy and ethical concerns when processing sensitive information. Always anonymize data where possible.

7. Conclusion

Masking unwanted text in unstructured documents is a crucial task for data privacy and content management.

LLMs provide a powerful tool for automating this process, offering flexibility and contextual understanding that traditional methods lack.

By following this tutorial, you should be able to set up a basic masking system using LLMs and customize it to your specific needs.

As you gain more experience, consider fine-tuning models for even better performance and efficiency.

Link of collab Notebook

LLM_Text_Masking_to_Protect_Sensitive_Data

August 21, 2024

1 LLM Text Masking to Protect Sensitive Data

1.1 Table of Contents

#**1. Introduction** ##Large Unstructured documents often contain a mix of relevant and irrelevant information.

##When working with sensitive data, it's crucial to mask or remove unwanted text to protect privacy or streamline content for analysis.

##Large Language Models (LLMs) like GPT-3.5 can be effectively used to identify and mask these unwanted text segments in an automated fashion.

#**2. Understanding Masking and its Importance**

##**2.1 What is Masking?** Masking refers to the process of obscuring or removing certain parts of a text.

This is particularly important in scenarios like data privacy, where sensitive information (e.g., personal identifiers, confidential details) needs to be hidden.

##**2.2 Why Use LLMs for Masking?** Traditional methods for masking involve regular expressions or rule-based systems, which can be brittle and hard to scale.

LLMs, on the other hand, can understand the context and meaning behind text, making them more effective at identifying which parts of the text should be masked.

#**3. Setting Up Your Environment** ##Before we dive into the code, let's set up the environment.

[ ]:
```
# Install required libraries

!pip install openai==0.28
!pip install pandas
```

```
Requirement already satisfied: openai==0.28 in /usr/local/lib/python3.10/dist-
packages (0.28.0)
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.10/dist-
packages (from openai==0.28) (2.32.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from openai==0.28) (4.66.5)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from openai==0.28) (3.10.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.20->openai==0.28) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28)
(2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.20->openai==0.28)
(2024.7.4)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (2.3.5)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->openai==0.28) (24.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->openai==0.28) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->openai==0.28) (4.0.3)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.1.4)
Requirement already satisfied: numpy<2,>=1.22.4 in
/usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

#Import necessary libraries and set your OpenAI API key: We'll be using OpenAI's GPT model for this tutorial, so ensure you have an API key from OpenAI

[ ]:
```
import os
import openai
import pandas as pd

openai.api_key = "your_API_key"  # Replace with your actual API key
```

2 4. Using LLMs to Identify and Mask Unwanted Text

Let's start by loading a sample unstructured document

[ ]:
```
# Sample unstructured document
document = """
Dear John,

Your social security number is 123-45-6789. Please keep it safe.

Also, note that your appointment with Dr. Smith is scheduled for tomorrow at 10␣
 ↪AM.

Best regards,
Jane Doe
"""
```

#Now, we'll use an LLM to identify and mask sensitive information.

#This function sends a prompt to the LLM asking it to identify and mask sensitive information.

#The model returns the modified document with sensitive information replaced by the mask token.

[ ]:
```
def mask_unwanted_text(document, mask_token="[MASK]"):
    prompt = [
        {"role": "system", "content": "You are a helpful assistant that masks␣
 ↪sensitive information in text."},
        {"role": "user", "content": f"Please mask any sensitive information␣
 ↪like name, phone numbers, email addresses, or credit card numbers in the␣
 ↪following text, replacing them with '{mask_token}':\n\n{document}"}
    ]

    response = openai.ChatCompletion.create(
      model="gpt-3.5-turbo", # or another suitable model
      messages=prompt
    )

    return response.choices[0].message['content'].strip()

masked_document = mask_unwanted_text(document)
print(masked_document)
```

```
Dear [MASK],

Your social security number is [MASK]. Please keep it safe.

Also, note that your appointment with Dr. Smith is scheduled for tomorrow at 10
AM.

Best regards,
Jane Doe
```

##This function sends a prompt to the LLM asking it to identify and mask sensitive information. The model returns the modified document with sensitive information replaced by the mask token.

2.1 5. Customizing Masking with Fine-Tuning

###For more specialized tasks, you might want to fine-tune an LLM on your specific dataset. Here's a high-level overview of how you could approach this:

2.2 1. Collect and Label Data: Gather a dataset of documents with the text you want to mask labeled accordingly.

2.2.1 2. Fine-Tune the LLM: Use a platform like Hugging Face or OpenAI's fine-tuning API to train the model on your specific masking task.

2.2.2 3. Deploy and Test: Use the fine-tuned model to mask unwanted text in your documents.

###Fine-tuning can greatly improve the accuracy of your masking, especially for industry-specific documents.

2.3 6. Best Practices and Considerations

###For more specialized tasks, you might want to fine-tune an LLM on your specific dataset. Here's a high-level overview of how you could approach this:

2.3.1 1. Contextual Awareness: Ensure that the model understands the context of the document. Overzealous masking might remove important information.

2.3.2 2. Testing and Validation: Always validate the output manually or through automated tests to ensure the correct text is being masked.

2.3.3 3. Ethical Considerations: Be mindful of privacy and ethical concerns when processing sensitive information. Always anonymize data where possible.

3 4. Conclusion

###Masking unwanted text in unstructured documents is a crucial task for data privacy and content management. LLMs provide a powerful tool for automating this process, offering flexibility and contextual understanding that traditional methods lack.

###By following this tutorial, you should be able to set up a basic masking system using LLMs and customize it to your specific needs. As you gain more experience, consider fine-tuning models

for even better performance and efficiency.