To use DeepSeek to process your own data and retrieve token IDs for each batch, you can follow these steps. This involves sending your data to the DeepSeek API, processing it, and receiving the tokenized output. Below is a Python example to help you achieve this:

Step 1: Install Required Libraries

Make sure you have the `requests` library installed to interact with the DeepSeek API:

bash Copy

```
pip install requests
```

Step 2: Set Up Your API Key

Obtain your DeepSeek API key from your account dashboard.

Store it securely in your environment or configuration file.

Step 3: Prepare Your Data

Organize your data into batches (e.g., lists of strings or JSON objects).

Ensure your data is in a format compatible with the DeepSeek API.

Step 4: Send Data to DeepSeek and Retrieve Token IDs

Here's an example Python script to send your data to DeepSeek and get token IDs for each batch:

python Copy

```
import requests

# Replace with your DeepSeek API key
API_KEY = "your_api_key_here"
API_URL = "https://api.deepseek.com/v1/tokenize"  # Example endpoint (check DeepSeek docs for the correct one)

# Example data batches
data_batches = [
    "This is the first batch of text data.",
    "Here is the second batch of text data.",
    "And this is the third batch of text data."
]

# Headers for the API request
headers = {
    "Authorization": f"Bearer {API_KEY}",
    "Content-Type": "application/json"
}

# Function to send data and get token IDs
def get_token_ids(data):
    payload = {
        "input": data,
        "model": "deepseek-v3"  # Specify the model you're using
    }
    response = requests.post(API_URL, headers=headers, json=payload)

    if response.status_code == 200:
        return response.json().get("token_ids", [])
    else:
        print(f"Error: {response.status_code}, {response.text}")
```

```
    return None

# Process each batch and retrieve token IDs
for i, batch in enumerate(data_batches):
    token_ids = get_token_ids(batch)
    if token_ids:
        print(f"Batch {i + 1} Token IDs: {token_ids}")
    else:
        print(f"Failed to process Batch {i + 1}")
```

Step 5: Handle API Responses

The `get_token_ids` function sends a batch of data to the DeepSeek API and retrieves the token IDs.

If the API call is successful, it returns the token IDs; otherwise, it prints an error message.

Step 6: Batch Processing

Loop through your data batches and process them one by one.

Store or use the token IDs as needed for your application.

Step 7: Error Handling and Retries

Add error handling and retry logic to manage API rate limits or temporary failures.

Example:

python Copy

```
import time

def get_token_ids_with_retry(data, retries=3):
    for attempt in range(retries):
        token_ids = get_token_ids(data)
        if token_ids:
            return token_ids
        time.sleep(2 ** attempt)  # Exponential backoff
    return None
```

Step 8: Save or Use Token IDs

Save the token IDs to a file, database, or use them directly in your application.

Example of saving to a file:

python Copy

```
with open("token_ids.txt", "w") as f:
    for i, batch in enumerate(data_batches):
        token_ids = get_token_ids(batch)
        if token_ids:
            f.write(f"Batch {i + 1}: {token_ids}\n")
```

Notes:

. **API Endpoint**: The `API_URL` in the example is a placeholder. Check theDeepSeek API documentationfor the correct endpoint for tokenization.

. **Rate Limits**: Be mindful of API rate limits and adjust your batch sizes or add delays if necessary.

. **Data Privacy**: Ensure your data complies with DeepSeek's privacy policy and terms of service.