Batch Processing for Automated Grading System

Via Azure OpenAI – Efficient & Affordable Large-Scale Processing
12 June, 2025

Nafisa Islam

MS Candidate ECE UCF

GRA – UCF Research IT

nafisa.islam@ucf.edu

Dr. Fahad Khan

Research Cyberinfrastructure

Facilitator

fahad.khan@ucf.edu

UCF Office of Research Cyberinfrastructure

Azure & Azure OpenAI

Azure

- Microsoft's cloud platform that lets people run websites, apps, and store data without needing their own servers.
- Think of it as renting powerful computers over the internet to do work like hosting apps, storing files, or running big calculations. (You can remove this if needed)

Azure OpenAI

- A special service in Azure that gives access to advanced AI models like ChatGPT, DALL·E, and Codex.
- It lets you build smart apps that can talk, summarize text, write code, or generate images—powered by OpenAI's technology but with Microsoft's cloud.

What's the Challenge?

- Imagine being handed 500 student essays and asked to grade them manually — not just for accuracy, but also consistency and fairness. If we try to use AI by sending each essay one-by-one to a model, we'll quickly run into problems:
- It's slow — each response takes time.
- It's costly — every request adds to our bill.
- It's hard to manage —we have to match results manually.
- We needed a smarter way to handle large volumes of text — and that's where **batch processing** comes in.

The Power of AI for Batches

Azure OpenAI Batch Processing is a way to run powerful AI models (like ChatGPT) on *lots of data at once* — in the background without needing to sit and wait for each result one-by-one.

✅ **Asynchronous (Runs in the background)**
Start the job, walk away, and come back later for the results. You don't need to keep your app waiting or running.

💰 **Cost-Effective**
Batch jobs are much cheaper than real-time calls when you have thousands of tasks. Perfect for grading assignments, analyzing survey responses, or summarizing long documents.

⚙️ **Scalable and Efficient**
It's built for *big jobs*. Azure handles the heavy lifting, so your system doesn't slow down — even with millions of words.

🧩 **Simple Setup**
No complicated coding. You upload a file (like a spreadsheet or a list of prompts), choose the AI model, and hit "Start" on the Azure Portal.

The Post Office Analogy

Let’s think about how mail works.

- **Real-Time API = Courier Service**

  If we need something delivered urgently, we hire a courier. It’s fast and direct, but it costs more. That’s what real-time API calls are like — great for urgent or single tasks, but inefficient for large volumes.

- **Batch Processing = Standard Mail**

  Now imagine dropping all our letters in a mailbox. A postal worker collects them all and delivers them in bulk. It’s slower, but much cheaper and perfect for large quantities.

- Azure Batch works the same way — we send all our requests together and let Azure process them efficiently in the background.

How It Works – 3 Simple Steps

1.Prepare **our File**
Create a .jsonl file where each line contains one prompt (like a student answer) along with the grading instructions (system message).

2.Upload **to Azure**
Go to Azure OpenAI Studio or use a Python script to upload the batch. We can monitor the job's progress while it runs.

3.Download **Results**
Once complete, we'll get an output .jsonl file with AI-generated feedback for each input. We can link this to grading sheets using custom IDs.

What's Inside the Input File (`input.jsonl`)

- Each line in the input file contains:
- **Model name**: Tells Azure which model to use (e.g., `"gpt-4"`).
- **System message**: Your grading rubric — this is what tells the AI how to evaluate.
- **User message**: The student's response that needs to be graded.
- **custom_id**: A unique identifier for each response — like a student ID or label.

```
{
  "model": "gpt-4",
  "messages": [
    {"role": "system", "content": "Grade based on clarity, correctness, and depth."},
    {"role": "user", "content": "Photosynthesis is how plants turn sunlight into energy."}
  ],
  "custom_id": "student_023"
}
```


UCF

What's in the Output File (`output.jsonl`)

Each line in the output file contains:

- **custom_id**: A unique identifier that matches the input line — this links the feedback to the correct student.
- **AI feedback**: A detailed evaluation based on the rubric. The feedback explains whether each rubric item is satisfied or not.
- **Rubric breakdown**: Step-by-step analysis for each item (e.g., "Item 1: Satisfied (1 point), Item 2: Not satisfied").
- **Final grading vector**: A summary showing which items were satisfied. Example: {1, 0, 1} for a 3-item rubric.


```
{
  "custom_id": "student_023",
  "response": {
    "message": {
      "role": "assistant",
      "content": "Grade: 7/10. The answer is mostly correct but lacks detail. Mention glucose and
    }
  }
}
```


UCF

Demo Time! Let's See It in Action 🎬

We'll now walk through a full example:

1. **The Input File**
   - Simple JSONL format: model, system message, user message, custom_id
   - Example: "Evaluate this answer based on clarity and completeness."
2. **Using Azure OpenAI Studio**
   - Go to the Batch Processing tab
   - Upload the input file and start the job
   - Monitor the job status
3. **Review the Output**
   - Download the output JSONL file
   - Each result maps directly to the original prompt using the custom_id
   - We can visualize or import this into our grading system

Azure OpenAI : Access, Setup, and Considerations

- This presentation's workflow for batch uploads requires an Azure subscription with OpenAI services enabled
- Please coordinate with your institutional IT department to determine your available options

- If you are at UCF and have a research use case, please contact the ResearchIT@ucf.edu team
  - You will need to provide your own departmental or research funding to use Azure services at UCF
  - We can offer consultation support for developing various applications that use GPT or other LLM models
  - We can also provide access to other models, including Gemini (via Google Cloud Platform) and Claude (via GCP or Amazon Web Services)

- It is imperative to consult your institutional compliance and Information Security teams to ensure the proper handling of sensitive data (e.g., PII, FERPA)
  - They can help you with the appropriate architecture design and storage options to handle such data
  - An Azure OpenAI environment can be configured to handle sensitive data, such as PII or FERPA
  - They can also advise you on applying necessary privacy-preserving techniques like de-identification

What is Azure OpenAI?

- Azure OpenAI is a service by Microsoft that provides access to powerful OpenAI models (like GPT-4, GPT-3.5, Codex, and DALL·E) through Azure's cloud infrastructure

- It allows developers and organizations to build custom applications using language models — for tasks like summarization, document analysis, content generation, chatbots, code generation, and more

- It is designed for enterprise and research use, with capabilities to manage data privacy, scalability, and compliance

ChatGPT Is Just the Start — Build More with Azure OpenAI

❌ You can't automate anything — it's all copy-paste.

❌ You're limited to one conversation at a time — no integration, no workflows.

❌ You can't connect it to your files, systems, or data easily.

❌ You can't control where your data goes (not ideal for FERPA, HIPAA, etc.).

❌ You can't run it at scale (e.g., grading 1000 students, analyzing 10,000 documents).

✅ Automate workflows — no manual copy-pasting.

✅ Build your own GPT-powered tools (e.g., auto-grader, report summarizer, RAG)

✅ Connect to Microsoft tools like SharePoint, Teams, Outlook, etc.

✅ Keep your data private and compliant (FERPA, HIPAA, enterprise security)

✅ Run at scale — grade 10,000 answers in minutes, not hours.

Thank You

Supplemental Slides

UCF Office of Research Cyberinfrastructure

Research cyberinfrastructure – LLMs, AI, and more

ResearchIT@ucf.edu

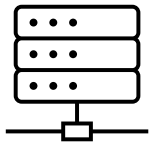
Office of Research Cyberinfrastructure's Mission

Our Mission:
Enable researchers through the effective use of research technology

Improve Research Computing and Data (RCD) Capabilities

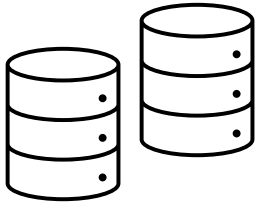
Improve Technology Support for Researchers

Research Computing and Data Capabilities

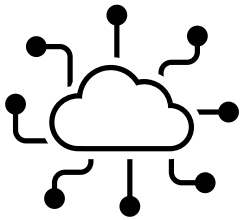


High performance computing

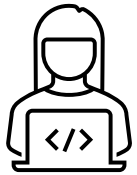
Training



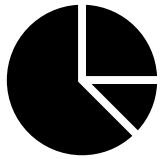
Research Data Storage



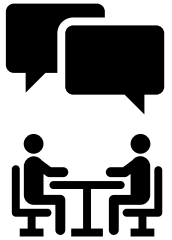
Cloud computing



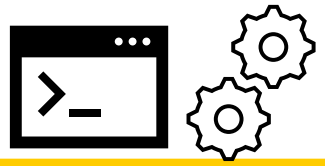
Tech talent for projects



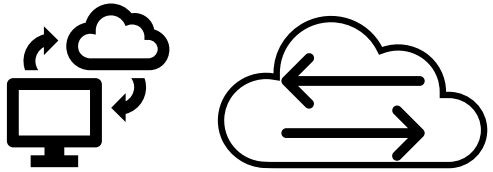
Research Data Visualization



Technical consultation



Research Software/tools



High speed research data transfers

Research Data Portals

Improving Support – Research Technology Facilitation

- Proposal Reviews
- Research technology support and setup at Award
- Technical Consulting and Cost Estimation
- Full support for Cloud Computing projects
- Training for existing people
- Hiring Technical Talent

Research Engagement Team

- Our research engagement team has a diverse range of expertise which includes computational science, high-performance computing (HPC), Linux, containerization, artificial intelligence (AI), Amazon Web Services (AWS), Google Cloud, Python, Git, Qualtrics, and more. To initiate a research project discovery call and explore how we can support your specific needs, please reach out to us via

Office Hours

Research Computing and Data Capabilities

- High Performance Computing (https://arcc.ist.ucf.edu)
- National Computing Resources (https://rci.research.ucf.edu/resources/category/national-computing-resources/)
- Cloud Computing (https://rci.research.ucf.edu/resources/category/cloud-computing/)
- Research Data Management (https://rci.research.ucf.edu/resources/category/research-storage-data-management/; https://guides.ucf.edu/data/dataresources; https://stars.library.ucf.edu/)
- Research Software (https://rci.research.ucf.edu/resources/category/research-software/)
- Regulated Research (https://rci.research.ucf.edu/resources/category/regulated-research/)

Things To Consider

Azure OpenAI – Features, Performance, Tradeoffs

Why use Batch Uploads

- **Time Efficient**: Grade hundreds of answers in parallel
- 💲 **Cost Effective**: Pay per token used, not per request
- 💡 Reduces educator workload while scaling across large classrooms
- ⏱️ Works faster than synchronous API calls with large data

Batch API Cost and Use Case

Feature	Real-Time API (Pay-as-you-go)	Batch Processing API
How it Works	Processes many separate, instant requests.	Submits one large job with many requests at once.
Typical Cost Per Unit	Standard price per 1,000 tokens.	50% discount per 1,000 tokens.
Estimated Total Cost	Higher relative cost (e.g., ~$1,000).	Lower relative cost (e.g., ~$500).
Best For	Time-sensitive, interactive applications.	Large-scale, non-urgent processing tasks.

Let’s Talk Cost – Why Batch Saves Money

We get access to the same powerful AI models, but at a fraction of the cost. By allowing Azure to process our data when it's most efficient, they pass the savings directly on to us.

Analogy: Taxi vs. The Bus

- **Real-time API (The Taxi 🚕):**
  - **Action:** Hailing a private taxi for an immediate, direct trip.
  - **Cost:** We pay a premium for on-demand service and the driver's dedicated time.
  - This is like real-time processing: Ideal for urgent, single tasks but expensive for many individual trips.

- **Batch Processing (The Bus 🚌):**
  - **Action:** Buying a bus ticket and traveling with other passengers.
  - **Cost:** The cost is shared among all passengers, making it dramatically cheaper per person.
  - This is like batch processing: Perfect for non-urgent, high-volume tasks where we can "share the ride" with other data jobs, leading to huge savings.

Performance Considerations

•**Parallel Jobs**: We can split our workload into smaller files and run multiple batch jobs at once (e.g., by class or assignment).

•**Batch Size Tradeoff**: Larger files are efficient but may take longer. Aim for under 100MB or ~50K tokens per file.

•**Token Throughput**: Azure throttles how fast it processes tokens to avoid overload. This isn’t usually a problem unless we're sending many huge files.

•**Prompt Engineering Matters**: The system message (our grading instructions) affects the quality of responses — be clear and consistent.

•**Content Filtering**: Azure applies filters to detect harmful or inappropriate content, so keep prompts clean and aligned with our use case. You can set thresholds or even disable these filters to fit your research needs.

Improving Responses

- Prompt Engineering
  - System Message & User Message
  - Formatting of Output
  - https://cookbook.openai.com/examples/gpt4-1_prompting_guide
  - https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-the-openai-api
  - https://platform.openai.com/docs/guides/text?api-mode=responses

- Model Parameters
  - Temperature
  - Top-P
  - https://medium.com/@1511425435311/understanding-openais-temperature-and-top-p-parameters-in-language-models-d2066504684f
  - https://community.openai.com/t/cheat-sheet-mastering-temperature-and-top-p-in-chatgpt-api/172683

UCF

GUI vs. Python SDK – Which Should You Use?

- **Azure OpenAI Studio (GUI)**
  This is the easiest way to get started. It's web-based and requires no programming. Great for testing small jobs or one-time uploads.

- **Python SDK or REST API**
  If you're building something more complex — like an automated grading tool or LMS integration — the SDK gives us full control. We can automate uploads, monitor jobs, and handle results programmatically.

- Choose what fits your comfort level and project needs.

Cost Assessment

Grading Scenario

How AI Pricing Works: Understanding Tokens

1) What is a Token?

Tokens are the small pieces of text that an AI model actually reads and processes. Think of them as the "building blocks" of language for the AI.

- A token can be a whole word ("apple"), a part of a word ("-ing"), or punctuation (".").
- Example: The word "tokenization" is split into two tokens: "token" and "ization".

2) The Word-to-Token Relationship

It's not a 1-to-1 relationship. The number of tokens is usually different from the number of words.

- Rule of Thumb: For English text, **100 tokens ≈ 75 words**.
- LLMs process tokens – This is why all pricing and model limits are measured in tokens, not words.
  - https://azure.microsoft.com/en-us/pricing/details/cognitive-services/openai-service/?cdn=disable

3) The Fundamental Cost Formula

The cost of every single API call is based on the number of tokens you send and receive.

Cost = (Input Tokens × Input Cost per Token) + (Output Tokens × Output Cost per Token)

- Input Tokens: The text you send to the model (including instructions and your query).
- Output Tokens: The text the model generates and sends back to you.

Cost Assessment for the Grading Scenario

- The Grading Scenario:
  - Before we calculate, let's define our project.
  - We want to automatically grade student assignments.
  - **S** number of Students
  - **Q** number of Questions per student
  - **I** number of grading Iterations for each answer
  - Total Items to Grade = S x Q
  - Total AI interactions = **S x Q x I**

- The total cost is determined by two simple things:
  - How many times we need to ask the AI for a grade.
  - How much each of those individual requests costs.
  - **Total Cost = (Total Number of API Calls) x (Cost Per Single API Call)**

Cost Assessment – Calculations

$$\text{Cost per API Call} = (\text{Input Tokens} \times \text{Input Cost per Token}) + (\text{Output Tokens} \times \text{Output Cost per Token})$$

$$\text{Input Tokens} = \text{System Message Tokens} + \text{User Message Tokens}$$

$$1 \text{ Word} = 1.33 \text{ Tokens}$$

System Message : Prompt for evaluation instructions or grading rubric
User Message : Student's response that needs to be evaluation
Input Message : System Message + User Message
Output Message : Response from LLM -- assessing student response based on rubric

R_{WT} : The Ratio converting Words-to-Tokens ≈ 1.33
$(W_{\text{rub}} + W_{\text{sub}})$: The total word count of the input (rubric + submission)
W_{eval} : The word count of the AI's evaluation output
C_{in} : The Cost per input token
C_{out} : The Cost per output token

Cost Assessment – Complete Picture

$$\text{Total Cost} = (\text{Total Number of API Calls}) \times (\text{Cost Per Single API Call})$$

$$\text{Total Cost} = (S \times Q \times I) \times R_{WT} \times [(W_{rub} + W_{sub}) \times C_{\text{in}} + (W_{eval} \times C_{\text{out}})]$$

$S \times Q \times I$: The total number of API calls (Students $\times$ Questions $\times$ Iterations)

R_{WT} : The Ratio converting Words-to-Tokens

$(W_{\text{rub}} + W_{\text{sub}})$: The total word count of the input (rubric + submission)

C_{in} : The Cost per input token

W_{eval} : The word count of the AI's evaluation output

C_{out} : The Cost per output token