Azure AI Language

Key phrase extraction
- Max 5120 char

Question Answering
- Starts with knowledge base
  - Az AI Service → Create language Service → Enable Question Answering → Create AI search → Language Studio → Custom QA project

- Utterance
  - Phrase user enters
- Intent
  - Meaning of utterance
- Entities
  - Add context to intent
  - Learned, list, prebuilt.

Classifioation

Projets → Single vs Multi label

Metrics

- Recall → true posites / total labels
- Precision → true pos / all pos
- F1 → recall / position

Project Limits

- Training: 10–100k files
- Deployments: 10/project
- API:
  - Authoring → 10 POST + 100 GET/minute (train/deploy)
  - Analyze → 20 GET <u>or</u> POST (extract)
- Projects: 1 storge/project, 500 projects/resource, 50 mods/project
- Entities: 200 types, ≤500 characters

Labeling

Consistency, precision, completeness.

Semantic Kernal AI Agents

Components:

- AI Service Connectors
- Memory Connectors
- Functions + Plugins
- Prompt Templates
- Filters

Concepts

- Agents: entities using models, functions, memory to make decisions
- Collab: agent group chats
- Kernal: execution engine
- Tools + Plugins: plugins connect external, tools add function
- History

Termination Strategy

- Proper conclusion of agent task
- Conversation state
  - After termination, state set to False to reuse chat.

AI Agent using Foundry Agent Service

- Simplifies agent creation
- Features:
  - Automatic Tool Calling
  - Securely managed data
  - Out-of-the-box tools
  - Flex model selection
  - Security
  - Custom Storage
- Integration
  - Connect to Foundry Project
  - Reference agent
  - Create thread
  - Add messages, invoke with agent
  - Check thread status
- Tools
  - Code interpretor
  - Knowledge
    - Bing, File search, Azure AI search, MS Fabric
  - Action
    - Code interpreter, custom function, azure function, OpenAPI Spec

Custom Tools

Options:
- Custom Function (extension of Azure Functions?)
- Azure Functions
- OpenAPI tools
- Azure Logic Apps

Completion Quality

- Prompt engineering, model params, training data

Integration

- Azure OpenAI accessed by REST or SDK
- Endpoints: (vary by model)
  - Completion (input, 1+ completions)
  - ChatCompletion (chat input, next response generated)
  - Embeddings (input, return vector)

OpenAI Prompt Engineering

Blah blah...

Adjusting Model Params

- Temperature → variance in sentence structure
- Top-p → variance in chosen words.
- Change only one at a time!!!

- Recency bias → Last infomation in prompt takes priority
- Cues → leading words to shape response

RAG + OpenAI

- Graphics/Images: Response quality depends on text recognition
- Recomended to use Azure AI Studio for search service + index, chunks better.
- Token Limits (1 token ≈ 4 letters)
  - System message unlimited, but truncated by model's token limit. (400-4000)
  - Response limited to 1500 tokens using your own data.

RAG in a prompt flow

- Use index lookup so subsequent tools can use results to augment prompt

Ex:

$$\begin{tikzcd}
\text{inputs} \arrow[r] \arrow[drrr] & \text{modify\_query\_with\_history} \arrow[r] & \overset{\text{(RAG)}}{\text{lookup}} \arrow[r] & \text{generate\_prompt\_context} \arrow[d] & \\
 & & & \text{prompt\_variants} \arrow[d] & \\
 & & & \text{chat\_with\_context} \arrow[r] & \text{output}
\end{tikzcd}$$

Fine Tune with Foundry

- Maximize consistency
- Chat completions
  - Data needs system message, user message, assistant response. JSONl
  - Use weights for tuning. 0=ignore 1=tune
- Foundational vs fine-tuned
  - Filter ~~by~~ catalog fine-tuning tasks
  - Options
    - batch_size
      - # examples used in single forward backward pass.
      - larger for larger sets
      - larger batch, less model parameter updates, but less variance.
    - learning_rate_multiplier
      - smaller rate, less overfitting. Larger for larger sets.
    - n_epochs
      - cycles of training
    - seed
      - job reproducability

Flow

Executable workloads, 3 parts.

1. Inputs, data passed to flow
2. Nodes, tools that perform processing
3. Outputs, data produced by flow.

Flow Types:

- Standard – general LLM app dev
- Chat – conversational apps
- Eval – Performance evaluation

Connections + Runtimes

- Used to connect flow to external data/service/API
- Connection details in Azure Key Vault.
- Runtimes
  - Compute instance providing environment and packages needed for flow.

Variants + Monitoring

Variants

- Versions of tool nods with distinct settings.
- Only available in LLM tool
- Customice approach to tasks

Deploy Flow

- Deploy to online endpoint
- URL + key
- Run flow + output real-time

Monitor

Metrics

- Groundedness (source)
- Relevance (topic)
- Cohereance (logical flow)
- Fluency (grammar)
- Similarity (source vs. output)

Azure AI Foundry SDK (Preview).

- Provids Python + C# libraries
- Allows code to connect to endpoint

Project Connections

- Connected resources defined at hub & project level.
- SDK can view and use connected services

Create a chat client

- Create ChatCompletionsClient object
  - Easily change model using parameters Easily change

```
client = AIProjectClient()
chat = client.inference.get_chat_client()
response = chat.complete()
```

- Azure OpenAI SDK

  Azure AI SDK provides get_azure_openai_client() method for consumption with OpenAI SDK

Prompt Flow to Develop Language Model Apps in AAI Foundry

LLM Life cycle

1. Initialization → Define use case + solution
2. Experimentation → Dev flow, test small set
3. Eval + Refine → Assess flow, test large set
4. Production → Deploy + monitor

Initialization
1. Define objective
2. Collect sample
3. Build prompt
4. Design flow

Experiment
1. Run flow on data
2. Evaluate performance
3. Evaluate + refine
4. Modify flow

Production
1. Optimize flow
2. Deploy
3. Monitor

Images

- Moderate Images: Scan images, severity (safe, low, high)
- Multimodal: Scans using OCR

Choose + Deploy Models from Azure AI Foundry Portal Catalog

Questions

- Solve my use case?
- Select best model?
- Scale for real world?

Deploy model to endpoint

- Api request from client to endpoint, passed to model, return to client
- URI consists of AI hub name, model name, task for model

Optemize Performance

- Apply prompt patters to use prompt engineering.
- Model optimization strategies
  - Retrieval Augmented Generation (RAG)
    - Provided grounding context
  - Fine Tuning
    - Providing model with example prompts + responses
  - Context optimization - maximizing response accuracy
  - Model optimization - maximizing consistency

Monitor Azure AI Services

...

Diagnostic Logging

- Capture rich diagnostics for monitoring usage and troubleshooting
- Resources:
  - Log Analytics (query + visualize)
  - Storage (store LA)

Diagnostic Settings

- Set LA workspace

Deploy AI service Containers

AI Containers

- Microsoft Container Registry

Deployment

1. Container image for AI service downloaded, deploy to container host (docker, AKS, ACI (azure container instance)
2. Clients connect to container host endpoint, receive AI service results
3. Metrics sent to Azure for billing

- AI container images are subdivided by function, not service
- 3 settings to configure:
  - ApiKey
  - Billing
  - Eula

Azure AI Foundry Content Safety

- Set of content moderation features, available as a resource
- Four categories
  - Hate, sexual, self harm, violence
- Text
  - Moderate Text: Scan text, provide severity 0-6
  - Prompt shields: Scan prompts for jailbreak attempts and unsafe prompts
  - Protected Material: Copyrights
  - Groundedness: Ensure grounded response for accuracy