



Microsoft Surface + Azure Hackathon

# HelpChat

## Hybrid AI

---





**20+**

Years of Innovation

**0**

Advertising

**100%**

Growth Through  
Client Results



## Your Company's Secret Weapon

---

# Driving Your Competitive Advantage

---

From concept to implementation, our arsenal of comprehensive services ensure your technology projects succeed:



## Solutions Delivery Services

We develop custom software solutions that help organizations engage with their customers and empower their employees.



## Solutions Enablement Services

We provide services to help advance and support your technology initiatives.





CME Group



Gensler

GSK



JPMORGAN CHASE & CO.



PANDORA



TEGNA



verizon<sup>®</sup>



500+

—  
Happy Customers

# Trusted To Deliver Results

---

**Powerful solutions that deliver engaging customer experiences and empower employees.**

- ◆ Solving business challenges with both enduring and emerging technologies.
- ◆ Solutions at the intersection of hardware and software.
- ◆ End-to-end thinking that ensures seamless delivery.

# ABOUT ME

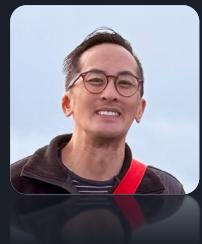
---



> 20 years at Teknikos



Worked on all types of  
Microsoft solutions



Jonathan Khoo  
CTO & VP Engineering



...rative and relative pronouns), definite noun phrase or proper name. Both  
sentential (sentence) and intersentential (discourse) relations have been  
important both for anaphora (a short-distance relation) and coreference  
(on). Cataphoric relations were not encountered in the corpus. Coreference  
occurred only between two discourse entities (NP1 and NP2) that refer to the  
world, i.e. **identity coreference**, as presented in (van Deemter and  
NP2 *corefer* if and only if *Referent(NP1) = Referent(NP2)*).

Using the entire set of ID variations, the top two parent categories for each were compiled into a list and counted by frequency. If there was a three-way tie (which occurred 9 times out of 40 rules), the parents were not separated into three different variation sets, but rather left as-is (e.g., ADJX/ADVX/DM was not split into ADJX/ADVX, ADVX/DM and ADJX/DM). The top five variations are: FKONJ/SIMPX, ADJX/NX, R-SIMPX/SIMPX, ADVX/PX, and ADVX/NX.

Results for the 10% cutoff – ambiguity heuristic follow:

(10%-ambig.)	Precision	Recall
Type	21.2% (7/33)	33.3% (7/21)

**Overview**

(1.2a)  $\left[ \begin{array}{l} \text{content} \\ \text{QSTORE } \{ \text{quant}^* \} \end{array} \right]$  (1.2b)  $\left[ \begin{array}{l} \text{word} \\ \dots \\ \text{NEW-QUANTIFIERS } \{ \text{quant}^* \} \end{array} \right]$

psoa nom-obj quant

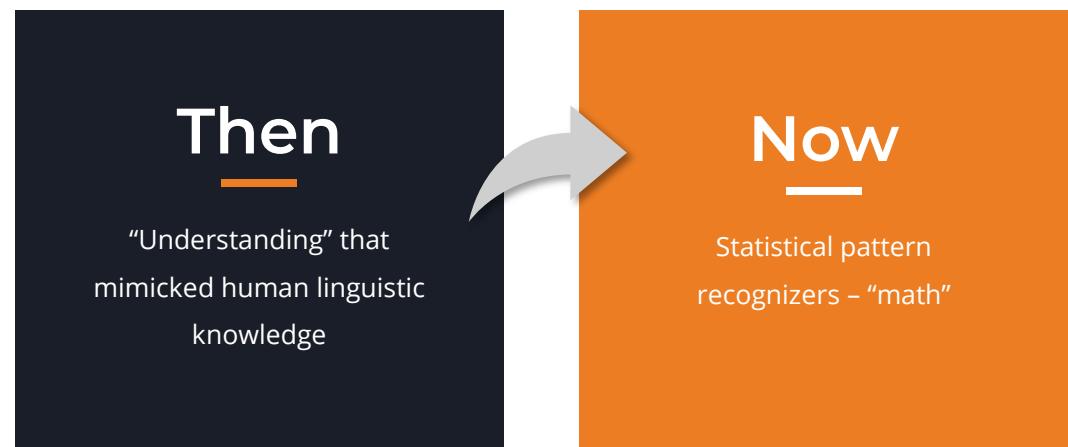
(1.3)  $\text{word} \rightarrow \text{Desc}_1 \vee \text{Desc}_2$

(1.4)  $\text{Desc}_1 = \left[ \begin{array}{l} \text{SS|LOC|CONT} \\ \text{QSTORE } \boxed{1} \end{array} \right] \vee \left[ \begin{array}{l} \text{nom-obj} \vee \text{quant} \\ \text{QSTORE } \boxed{2} \\ \text{QUANTS } \boxed{3} \end{array} \right]$

where  $\boxed{1} = \boxed{5} \uplus \text{union QSTORES of selected arguments}$   
 $\boxed{4} = \text{set of elements of } \boxed{3}$   
 $\boxed{1} = \boxed{2} \uplus \boxed{4}$

# UNLOCKED POSSIBILITIES WITH COMPUTING POWER

Local and in the cloud



← Actual data, papers, and presentations from my 2005-2007 Masters degree classes

# TERMINOLOGY

---



## Transformers

Embedding vectors pass through layers of transformer operations that help the model better understand and work with each word in context



## Tokenization

Breaking text down into smaller units (tokens)

"I love chocolate"

→ ["I", "love", "chocolate"]



## input\_ids

Model's vocabulary-specific ID number for each token

[ "I", "love", "chocolate" ]

→ [ 101, 4567, 9876 ]



## attention\_mask

Tells model which tokens to pay attention to

input\_ids: [ 101, 4567, 9876, 0 ]

attention\_mask: [ 1, 1, 1, 0 ]



## Embedding

Vector representation of token, "learned" during training

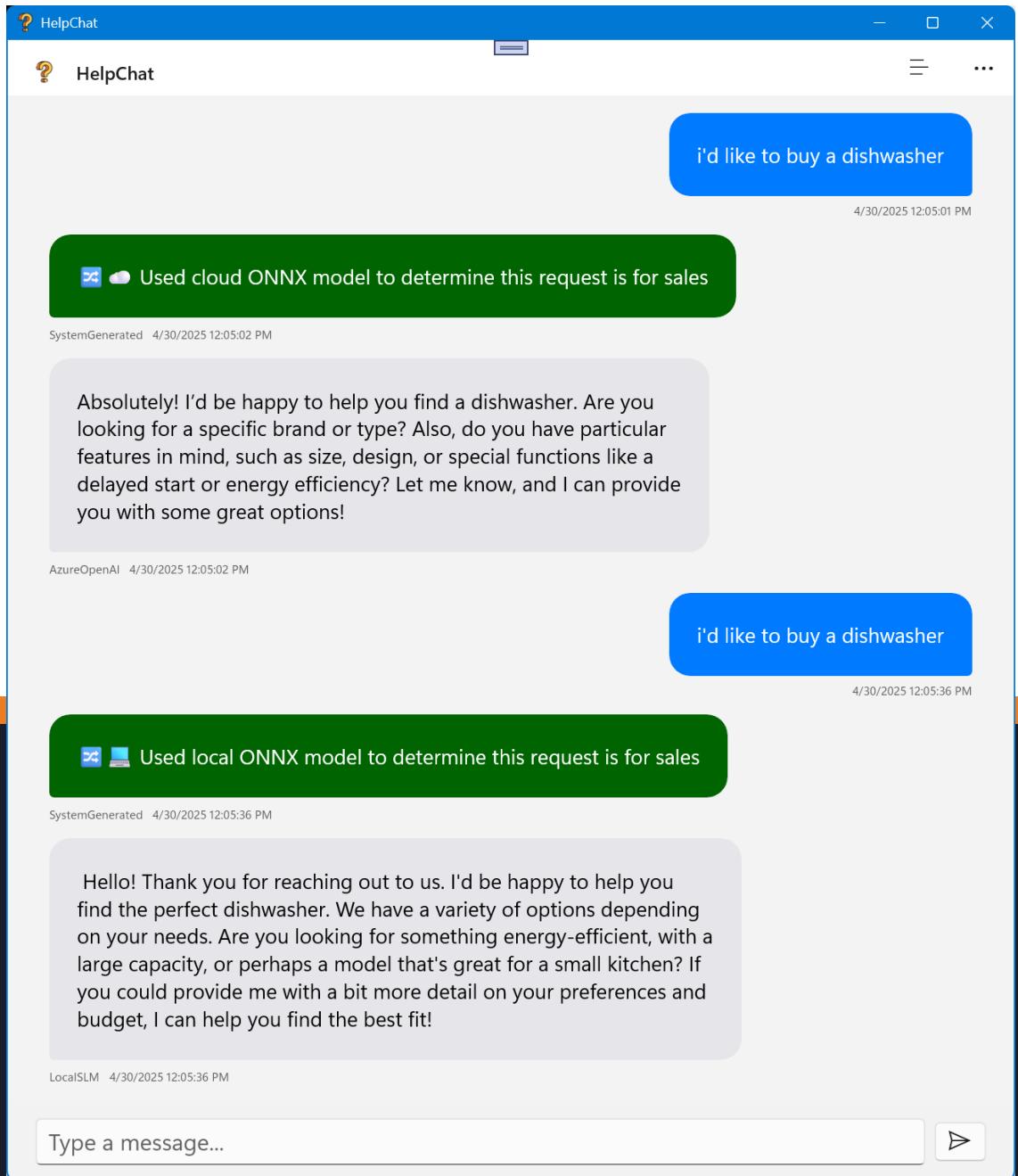
Hundreds, thousands of dimensions encode meaning and context

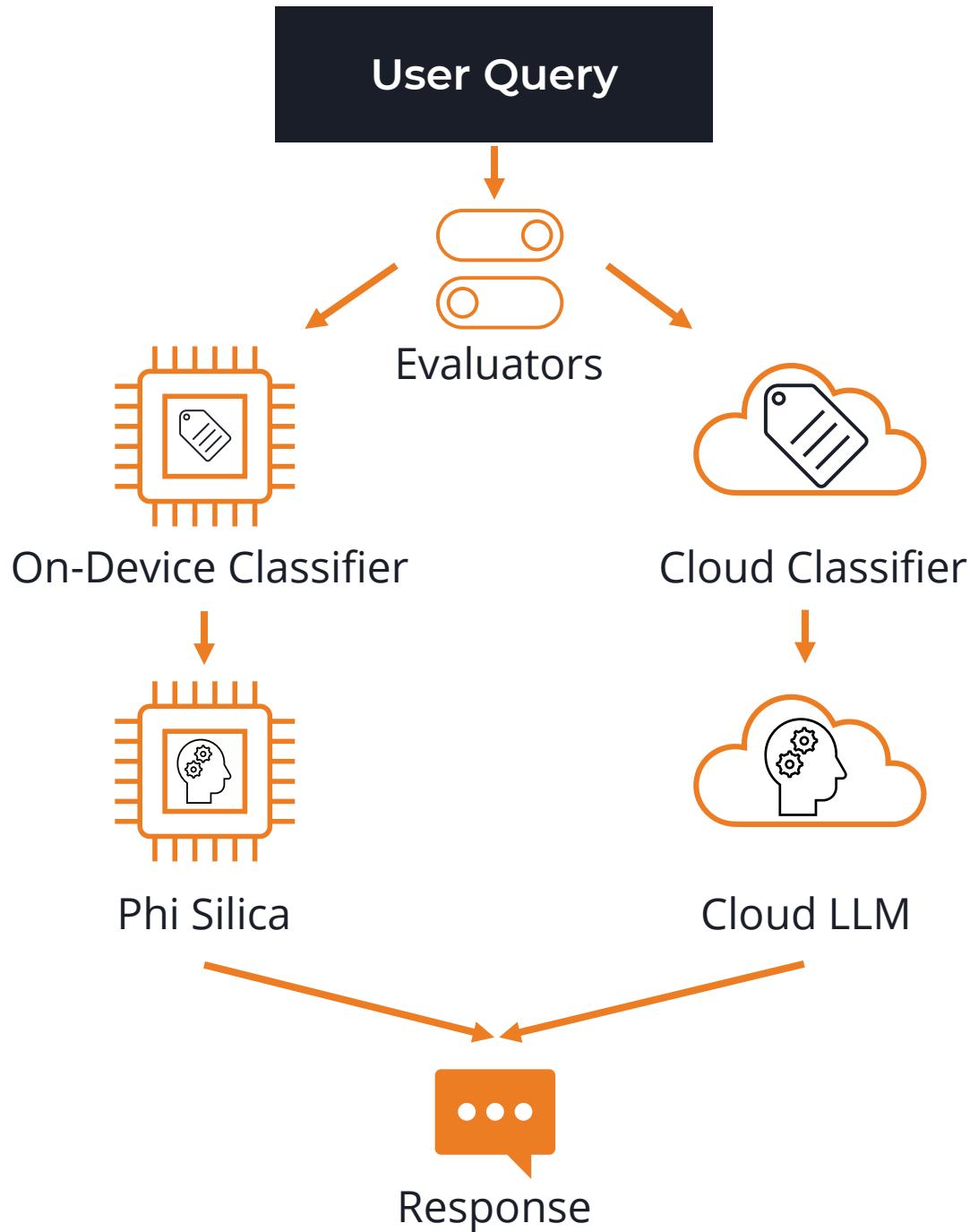
Group of embeddings is a *tensor*

# TODAY'S GOAL

Hybrid AI solution

- Probably not a real use case you'd build,
- BUT, a concrete example of possibilities on-device  
AI opens up





## FLOW & COMPONENTS

---

### Evaluators

---

Go/No-Go

### Classifier

---

Cross-Platform  
ONNX Model

# TO-DO LIST

---



Understand and Work with  
an ONNX Model



Implement Cloud Version



Implement Evaluators



Implement Local Version

# YOU ARE HERE

---



Understand and Work with  
an ONNX Model



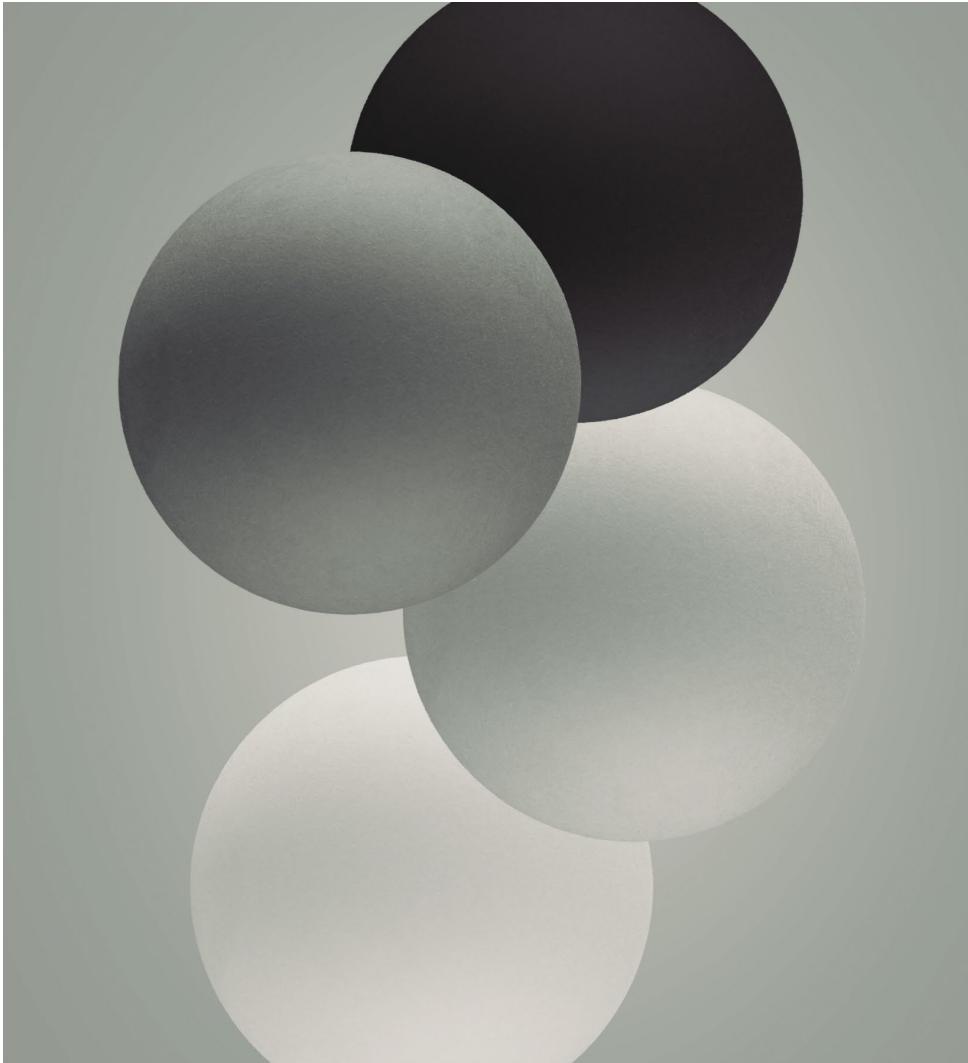
Implement Cloud Version



Implement Evaluators



Implement Local Version



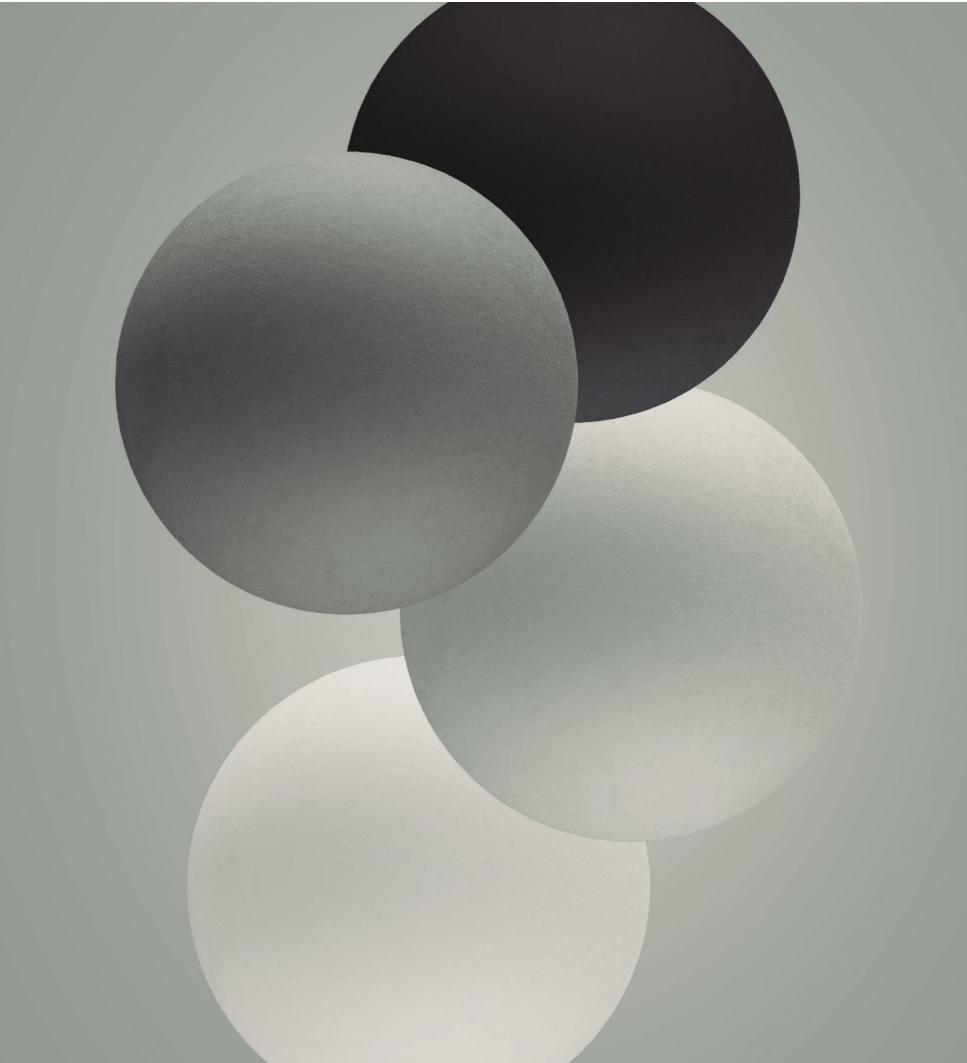
# WHAT IS ONNX?

---

- Open Neural Network Exchange
  - Open format for AI models
    - Solves interoperability problems between frameworks like PyTorch and TensorFlow
  - ONNX Runtime
    - Cross-platform, multiple language support
    - *Execution providers* take advantage of hardware
  - Same model in the cloud and on the device
- 



ONNX



## GET AN ONNX MODEL

---

- Train in Azure Machine Learning
- Get a pretrained model in ONNX Model Zoo or some Hugging Face listings
- Convert an existing model to ONNX
  - Take advantage of existing AI deployments



ONNX

---

# ZERO-SHOT CLASSIFICATION

---

“The prime minister met with international leaders.”

This text is about politics

This text is about weather

This text is about entertainment

This text is about sports

This text is about politics

This text is about science

**Semantic similarity** to measure likelihood of input being in a category

**Zero-shot** means we can give it possible category labels that the model hasn't been trained on

How likely does the input **entail a hypothesis**?

# DeBERTa

- BERT: Bidirectional encoder representations from transformers (2018, Google)
- DeBERTa: Decoding-enhanced BERT with disentangled attention (2020-2021, Microsoft Research)
- Fine-tuned version of DeBERTa:  
<https://huggingface.co/MoritzLaurer/deberta-v3-large-zeroshot-v1>

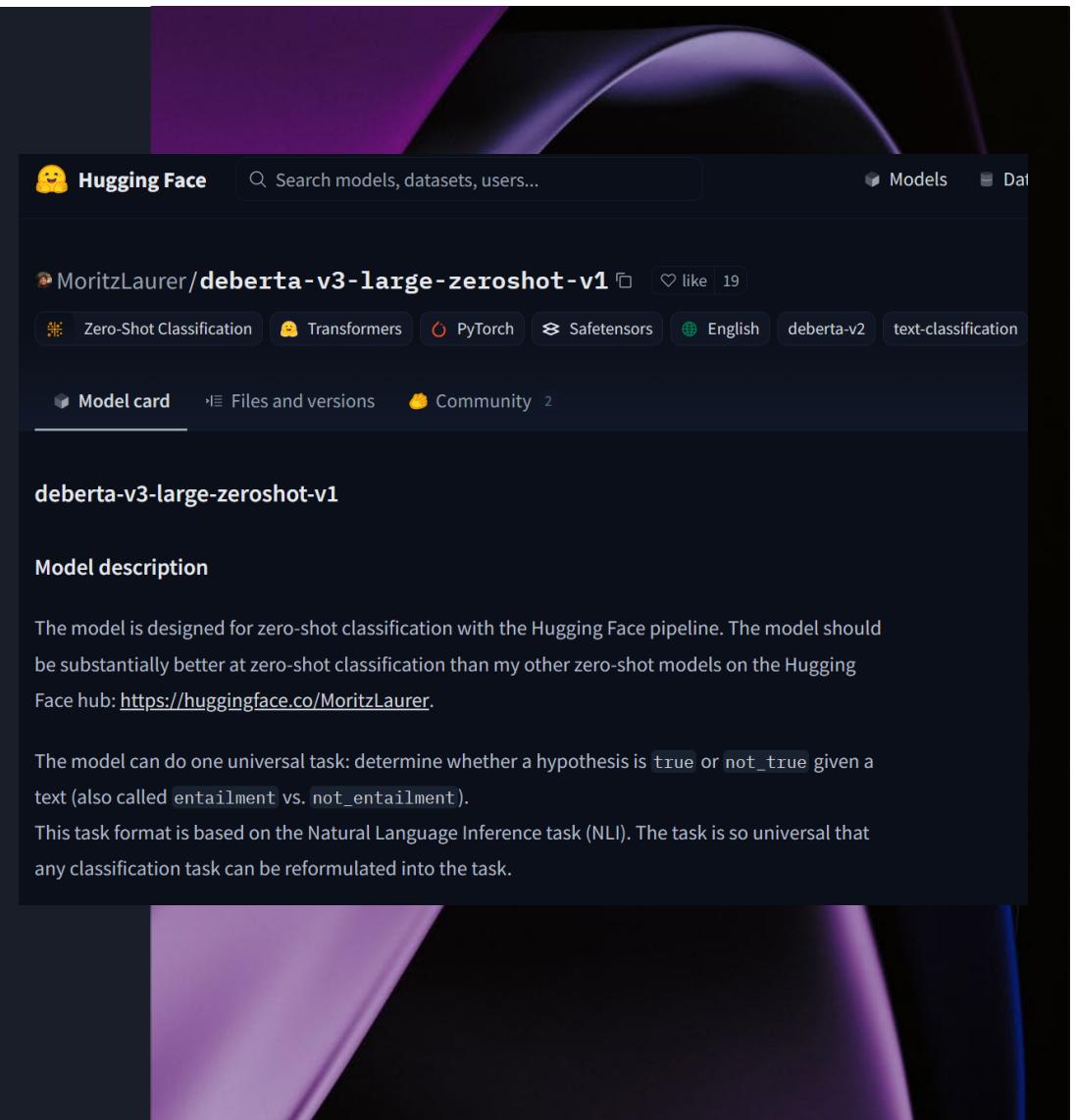
## Less Annotating, More Classifying: Addressing the Data Scarcity Issue of Supervised

### Machine Learning with Deep Transfer Learning and BERT-NLI

Moritz Laurer, Wouter Van Atteveldt, Andreu Casas, Kasper Welbers

Political Analysis, Cambridge University Press, 2023.

<https://doi.org/10.1017/pan.2023.20>



The screenshot shows the Hugging Face model card for `deberta-v3-large-zeroshot-v1`. The card includes the author's profile picture and name (`MoritzLaurer`), the model name, a like count of 19, and a link to the pipeline. Below the card are sections for the Model card, Files and versions, and Community. The Model description section explains the model's purpose for zero-shot classification and its performance compared to other models. The text format section describes the NLP task (Natural Language Inference) and its universality.

**Hugging Face** Search models, datasets, users... Models Data

MoritzLaurer/`deberta-v3-large-zeroshot-v1` like 19

Zero-Shot Classification Transformers PyTorch Safetensors English deberta-v2 text-classification

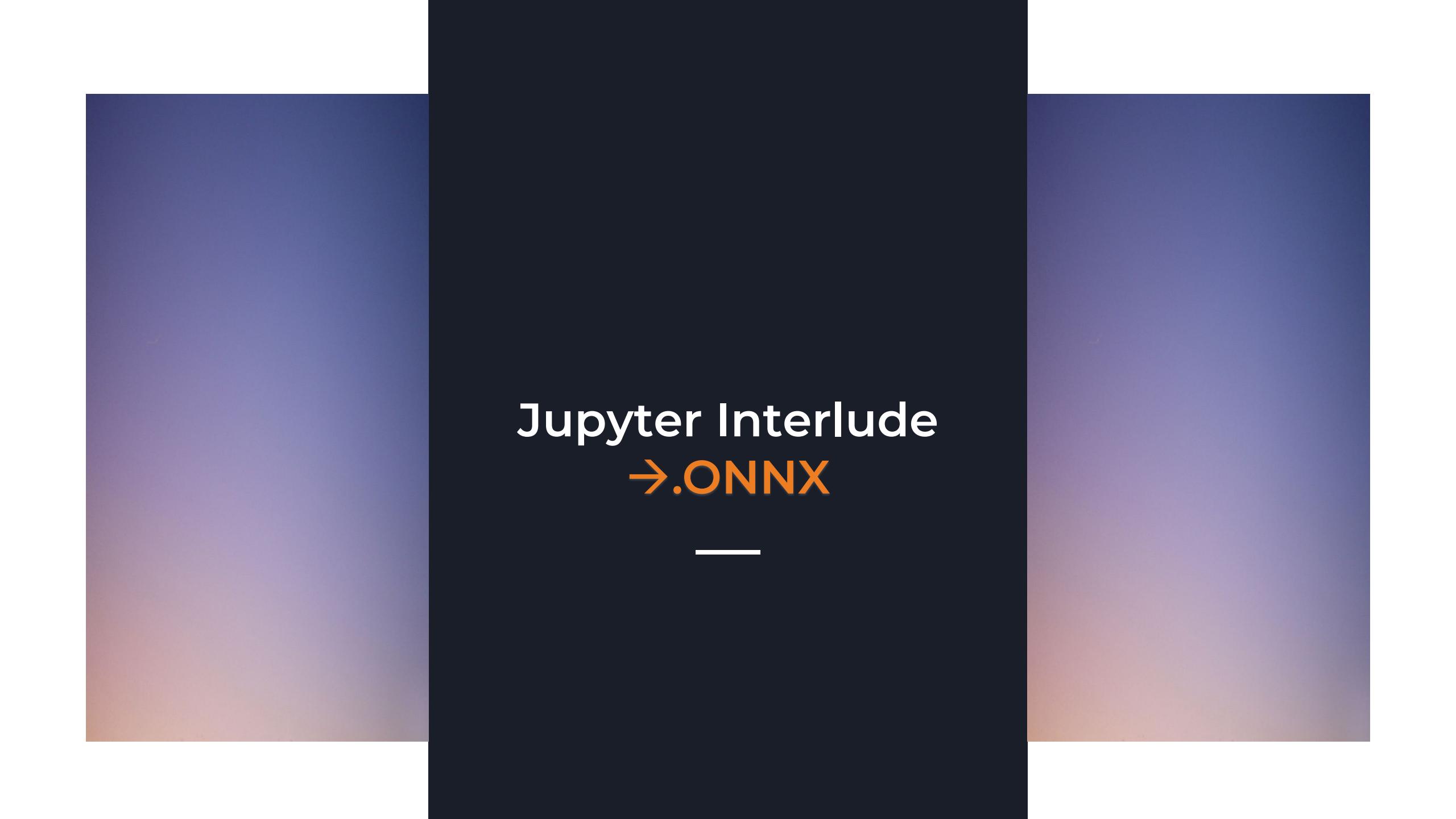
Model card Files and versions Community 2

### deberta-v3-large-zeroshot-v1

#### Model description

The model is designed for zero-shot classification with the Hugging Face pipeline. The model should be substantially better at zero-shot classification than my other zero-shot models on the Hugging Face hub: <https://huggingface.co/MoritzLaurer>.

The model can do one universal task: determine whether a hypothesis is `true` or `not_true` given a text (also called `entailment` vs. `not_entailment`). This task format is based on the Natural Language Inference task (NLI). The task is so universal that any classification task can be reformulated into the task.



# Jupyter Interlude

## →.ONNX

---

# Local Pipeline

```
import json
from transformers import pipeline
classifier = pipeline("zero-shot-classification", model="MoritzLaurer/deberta-v3-large-zeroshot-v1")
sequence_to_classify = "The prime minister met with international leaders"
candidate_labels = ["weather", "entertainment", "sports", "finance", "science", "politics"]
output = classifier(sequence_to_classify, candidate_labels, multi_label=False)
print(json.dumps(output, indent=4))

| ✓ 5.4s

Device set to use cpu
{
    "sequence": "The prime minister met with international leaders",
    "labels": [
        "politics",
        "finance",
        "entertainment",
        "science",
        "sports",
        "weather"
    ],
    "scores": [
        0.998672604560852,
        0.00031738588586449623,
        0.0002589169598650187,
        0.00025301624555140734,
        0.0002507023746147752,
        0.00024740086519159377
    ]
}
```



## Local Pipeline

```
import json
from transformers import pipeline
classifier = pipeline("zero-shot-classification", model="MoritzLaurer/deberta-v3-large-zeroshot-v1")
sequence_to_classify = "How many calories are in your hamburger?"
candidate_labels = [ "sales", "customer support", "technical support", "accounting", "marketing", "shipping and orders" ]
output = classifier(sequence_to_classify, candidate_labels, multi_label=False)
print(json.dumps(output, indent=4))

✓ 4.5s

Device set to use cpu
{
    "sequence": "How many calories are in your hamburger?",
    "labels": [
        "marketing",
        "accounting",
        "sales",
        "customer support",
        "technical support",
        "shipping and orders"
    ],
    "scores": [
        0.634022057056427,
        0.12989985942840576,
        0.06265775114297867,
        0.0603039376437664,
        0.06020446866750717,
        0.052911896258592606
    ]
}
```



## Local ONNX + QNN EP

```
# Use the QNN (aka Qualcomm AI Engine Direct) execution provider
options = onnxruntime.SessionOptions()
session = onnxruntime.InferenceSession(model_path, sess_options=options, providers=["QNNExecutionProvider"], p

# Create hypotheses
template = "This example is {}."
hypotheses = [template.format(label) for label in candidate_labels]

# Tokenize
encoded = tokenizer(
    [text] * len(candidate_labels),
    hypotheses,
    return_tensors="np",
    truncation=True,
    padding=True
)

# Run ONNX model
logits = session.run(
    None,
    {
        "input_ids": encoded["input_ids"],
        "attention_mask": encoded["attention_mask"],
    }
)[0]

# Use entailment scores (index 0) and normalize with softmax
entailment_scores = logits[:, 0]
probs = torch.softmax(torch.from_numpy(entailment_scores), dim=0).numpy()

# Zip labels and scores
results = sorted(zip(candidate_labels, probs), key=lambda x: x[1], reverse=True)

# Display
for label, score in results:
    print(f"{label}: {score:.4f}")

✓ 10.5s
marketing: 0.6341
accounting: 0.1299
sales: 0.0627
customer support: 0.0603
technical support: 0.0602
shipping and orders: 0.0529
```



# Local ONNX + QNN EP



# Azure Interlude

## Deploy ONNX in

---



[ml.azure.com](https://ml.azure.com)

## In Azure ML

```
# Test deployment
import requests
import json

url = "https://onnxworkspace-jfass.centralus.inference.ml.azure.com/score"

data = {
    "text": "How many calories are in your hamburger?",
    "labels": [ "sales", "customer support", "technical support", "accounting", "marketing" ]
}

headers = {'Content-Type':'application/json', 'Accept': 'application/json', 'Authorization': 'Bearer ' + token}

response = requests.post(url, json=data, headers=headers)
parsed = json.loads(response.content)      # parse the JSON string

print(parsed)
```

✓ 1.8s

[["marketing", 0.6340243816375732], ["accounting", 0.12989859282970428], ["sales", 0.06265752762], ["customer support", 0.05123456789012345], ["technical support", 0.048654321000000004], [





# FINAL WORDS ON ONNX

---

Cross-platform, but you must keep some things in mind



## Tweaks may be required

You may need to tweak files for optimum execution on local or cloud: multiple options to adjust. **OLIVE!**

You may see (ideally, none, but) minuscule differences in result numbers.



## Additional files

Cloud endpoint: **score.py** to tell Azure ML how to load and process data with the model (included!)

If dealing with text, may need **vocab.txt** for tokenization (included!)

# YOU ARE HERE

---



Understand and Work with  
an ONNX Model



Implement Cloud Version



Implement Evaluators



Implement Local Version

## BREAK INCOMING!



# CHOOSE YOUR OWN ADVENTURE

---

Depending on your comfort level...



## Build the app with me

Use readymade code snippets as we work through our solution



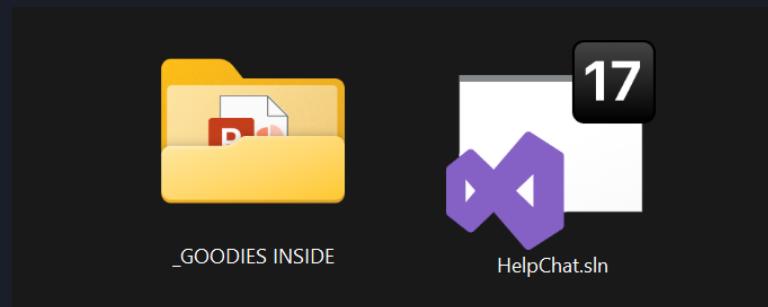
## DIY

Look at the code snippets to get an idea of what we are doing, but try it unsupervised (👉 machine learning joke 😅)

(or, take the offramp to DIY once you see how it works)

# GET READY

- One key folder and one key file
- Visual Studio: A quick familiarization



Files

Start Debugging

BREAK!

Checkpoint: Solution and Code Open

```
Labels.cs ChatService.cs* □ X
HelpChat
1  using HelpChat.Models;
2  using HelpChat.Utils;
3  using Microsoft.Extensions.AI;
4  using Serilog;
5  using System;
6  using System.Collections.Generic;
7  using System.Threading;
8  using System.Threading.Tasks;
9
10 namespace HelpChat.Services;
11
12 public class ChatService
13 {
14     private readonly List<HelpChatMessage> _conversationHistory = new();
15     private readonly IChatClient _chatClient;
16     private int _maxTokens = 800;
17
18     public Microsoft.UI.Dispatching.DispatcherQueue WindowDispatcherQueue { get; init; }
19 }
```

# CLOUD STEPS



By the end of this slide, you should have an app that welcomes the user, gets the classifier result from the ONNX model in Azure ML, and returns a GPT response.



## SHOW A WELCOME MESSAGE

**By the end of this step:**

- Run the application
- See a generated welcome message

## HANDLE SUBMITTED MESSAGES

**By the end of this step:**

- Type a message in the text box and submit it by pressing Enter or the send button

## GET CLOUD ONNX RESULT

**By the end of this step:**

- See a message with the classification of the input
- View logs

## GET GPT RESPONSE

**By the end of this step:**

- Verify we are telling GPT what persona (=label) to use
- See the response by GPT to a message
- Try a couple variations

# YOU ARE HERE

---



Implement Evaluators



Understand and Work with  
an ONNX Model



Implement Cloud Version



Implement Local Version

# WHAT ARE EVALUATORS

---

**Deciders that tell whether to use cloud or local AI**

## IMPLEMENTATIONS MAY VARY

- One evaluator failing is enough to sway decision?
- Or plurality of evaluators with weights attached to each



## PRIVACY

---

- Any data that is sensitive or personal?

## CONNECTIVITY AND LATENCY

---

- Remote-remote workers with a spotty or non-existent connection

## DEVICE RESOURCES

---

- Larger model in cloud for special cases
- Battery level

## COST

---

- Paying subscription user?
- Ways to integrate cost-savings dynamically

# EVALUATOR STEPS

By the end of this slide, you should have an evaluator system set up that knows about the user's connectivity as well as input message contents.



## SET UP INTERFACE & FRAMEWORK

Nothing to see; don't run anything just yet

## IMPLEMENT PRIVACY EVALUATOR

Keep holding your horses

## IMPLEMENT LATENCY EVALUATOR

Almost there (but you can see if it builds, as a treat!)

## PUT IT ALL TOGETHER

By the end of this step:

- We should get nothing (and not error out) if we have no connectivity or have a message with a long number in it
- Verify cloud still works, though!

# YOU ARE HERE

---



Understand and Work with  
an ONNX Model



Implement Cloud Version



Implement Evaluators

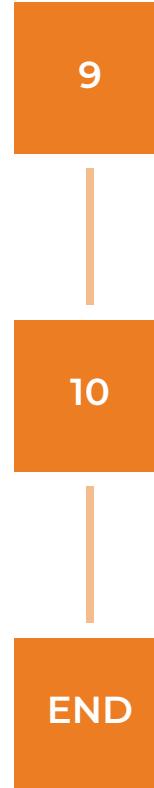


Implement Local Version

# LOCAL STEPS

---

By the end of this slide, you should have a complete application that decides on whether to use cloud or local AI, and generate a response appropriately.



## GET LOCAL ONNX RESULT

**By the end of this step:**

- See a message using local classification of the input

## GET PHI SILICA RESPONSE

**By the end of this step:**

- Get a response from Phi Silica

## YOU NOW HAVE A COMPLETE APP

Congratulations!



## EXTEND/ENHANCE THE SOLUTION

---

If there's time...

### GIVE IT A WHIRL

Try the solution out with various input statements.

#### ● ENGINEER THE PROMPTS

Play around with the prompts a bit and see if you can vary the responses that come back.

#### ● ADD MORE EVALUATORS

Try adding one that uses local AI if it's plugged in, and cloud if it's not

#### ● ADD CHAT HISTORY / RESET BUTTON

Right now it starts anew with each input message.

# RECAP AND NEXT STEPS

---

Today we:

- **Converted** a Hugging Face model to ONNX
- Deployed that model in the **cloud**
- Used that same model on the **device**
- Talked with an Azure OpenAI **GPT** deployment
- Talked with **Phi Silica** running on the NPU
- Implemented **evaluators** that can route AI requests to the cloud or the device, as required

## THINGS THAT MAKE YOU GO “HMM”

- Do you have any AI models in the cloud that you can bring on-device?
- Do you have any AI models you can run on-device?
- What sort of evaluators would you use in your application to decide cloud vs device?

# FINAL BITS



## GITHUB

Complete Code  
This presentation  
Jupyter notebook  
Score.py

<https://bit.ly/hybridailab>



## JONATHAN KHOO

jonk@teknikos.com