Microsoft Surface + Azure Hackathon

HelpChat
Hybrid AI

TEKNIKOS

TEKNIKOS

20+

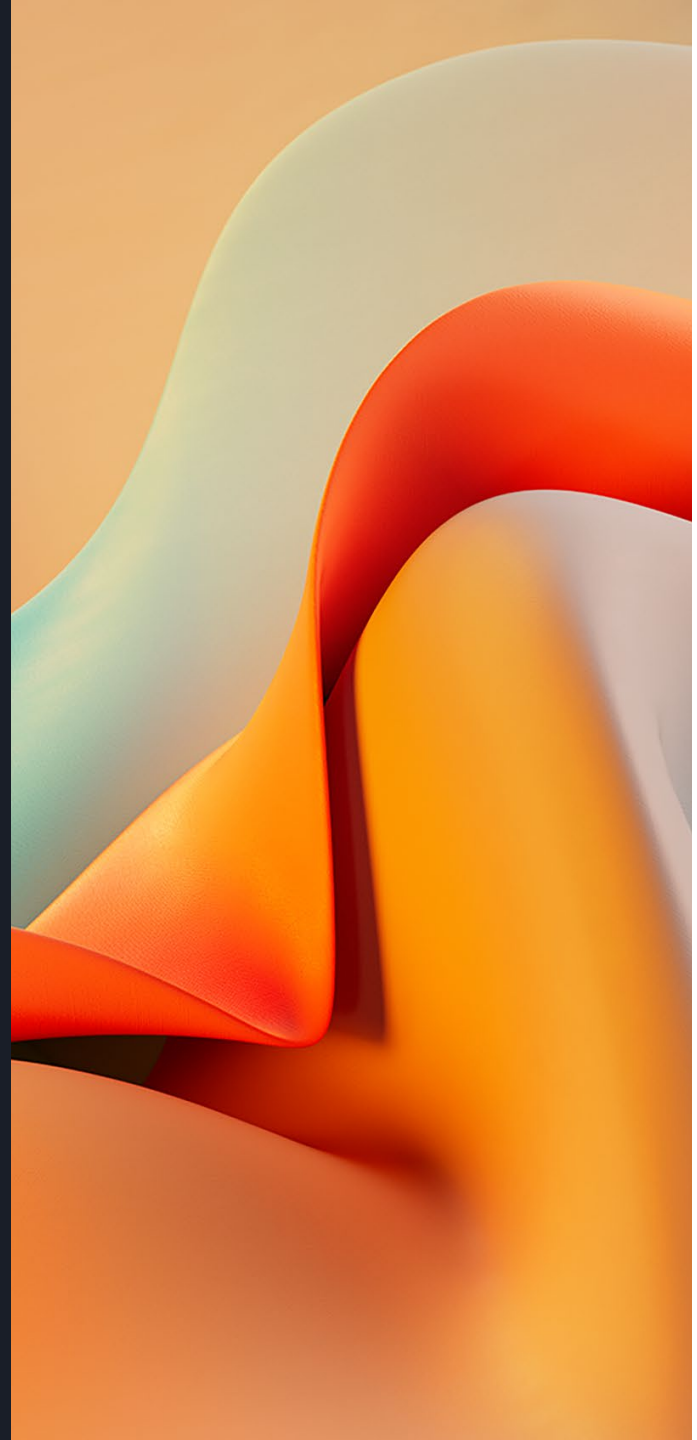
Years of Innovation

0

Advertising

100%

Growth Through
Client Results



Your Company's Secret Weapon

Driving Your Competitive Advantage

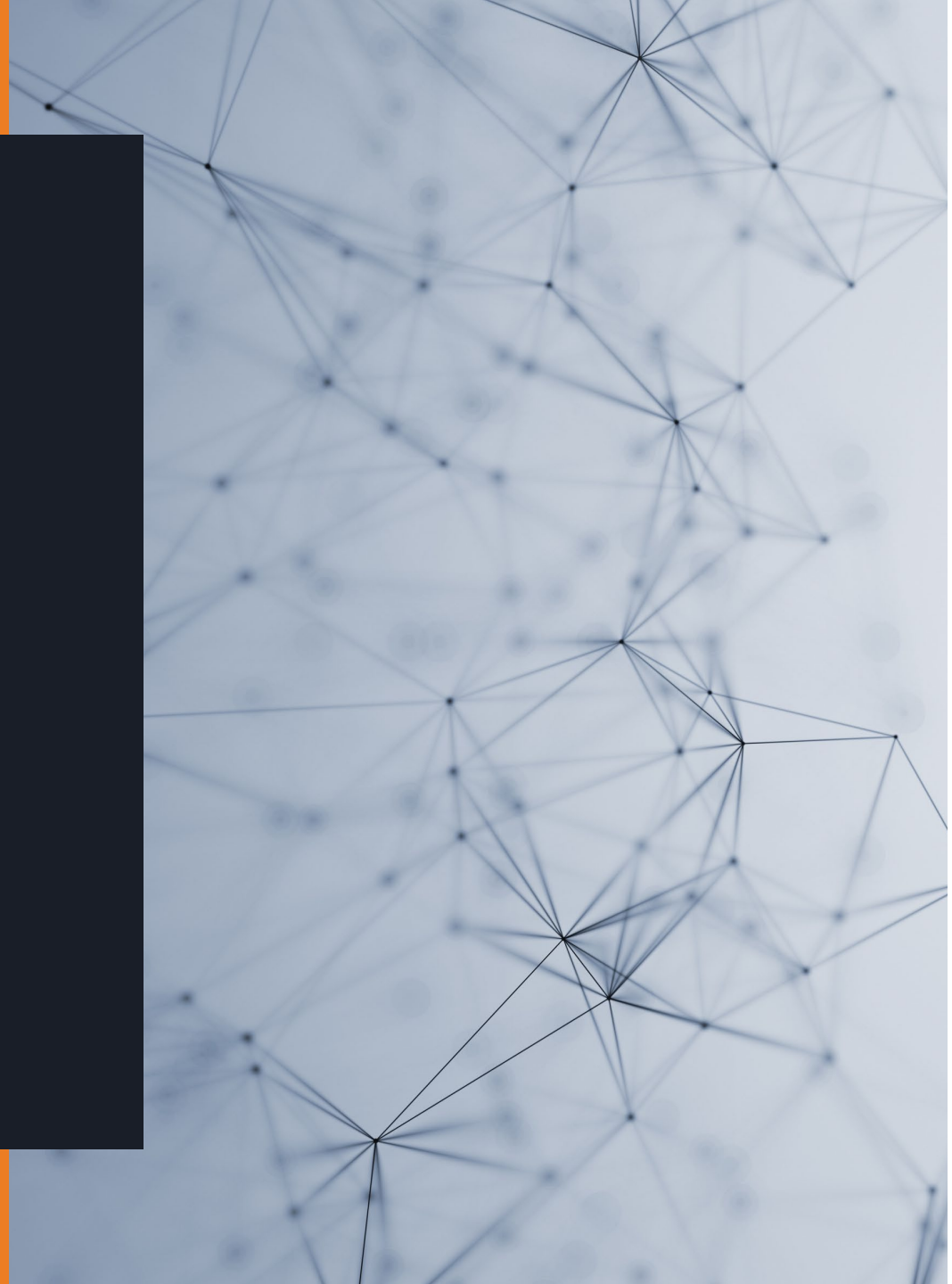
From concept to implementation, our arsenal of comprehensive services ensure your technology projects succeed:

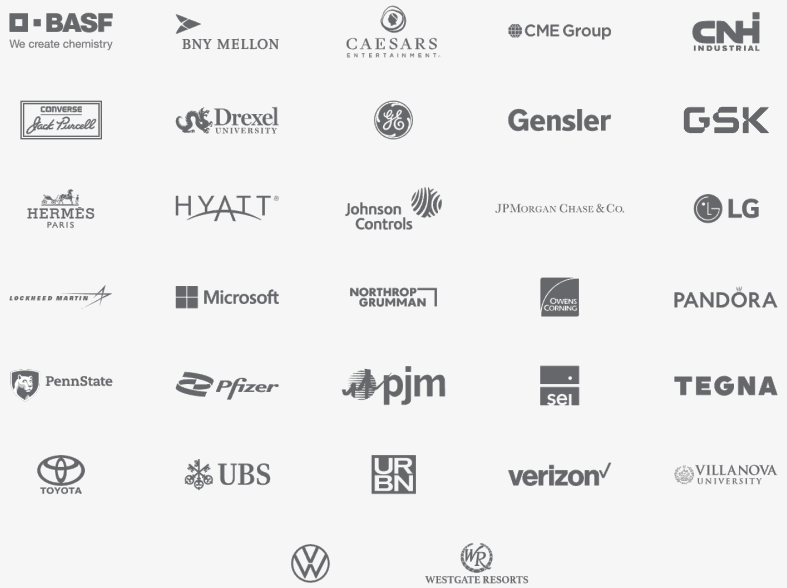
Solutions Delivery Services

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.





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.

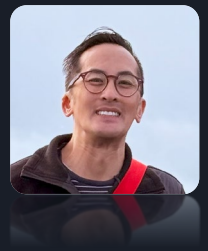
500+

Happy Customers

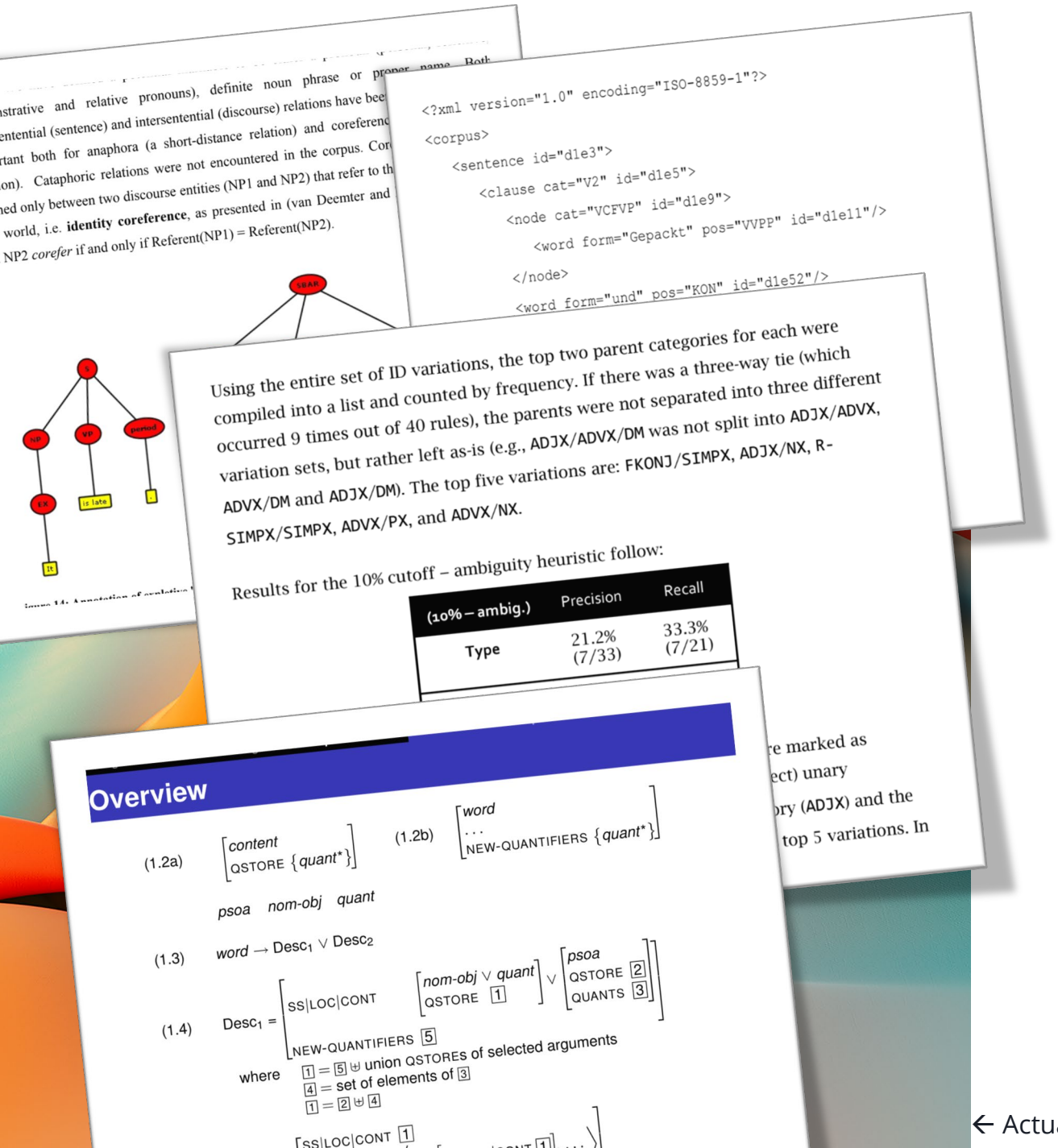
ABOUT ME

> 20 years at Teknikos

Worked on all types of Microsoft solutions



Jonathan Khoo
CTO & VP Engineering



UNLOCKED POSSIBILITIES WITH COMPUTING POWER

Local and in the cloud

Then

"Understanding" that mimicked human linguistic knowledge

Now

Statistical pattern recognizers – "math"

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

TERMINOLOGY

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]

Transformers

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

Embedding

Vector representation of token, “learned” during training

Hundreds, thousands of dimensions encode meaning and context

Group of embeddings is a *tensor*

attention_mask

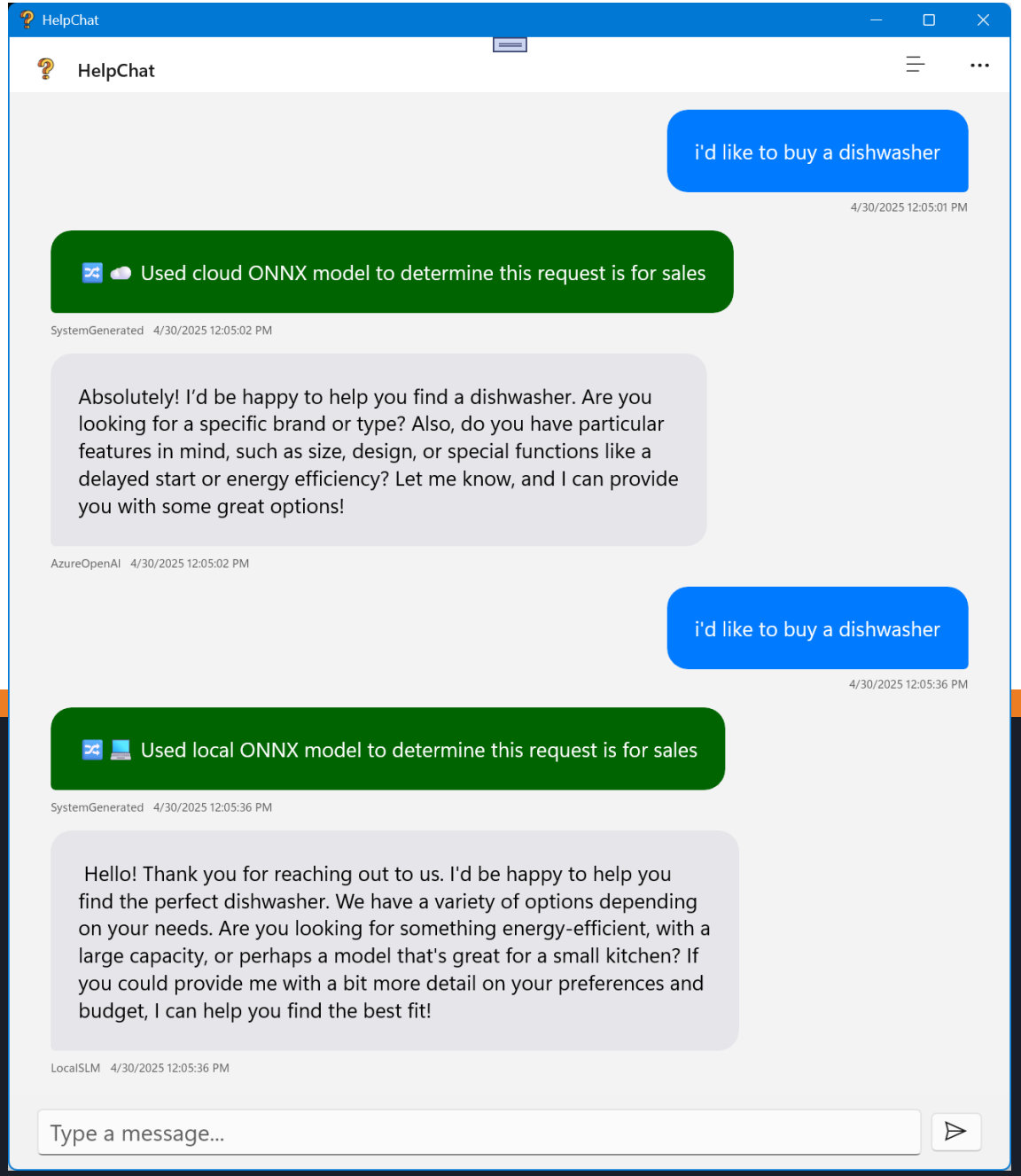
Tells model which tokens to pay attention to

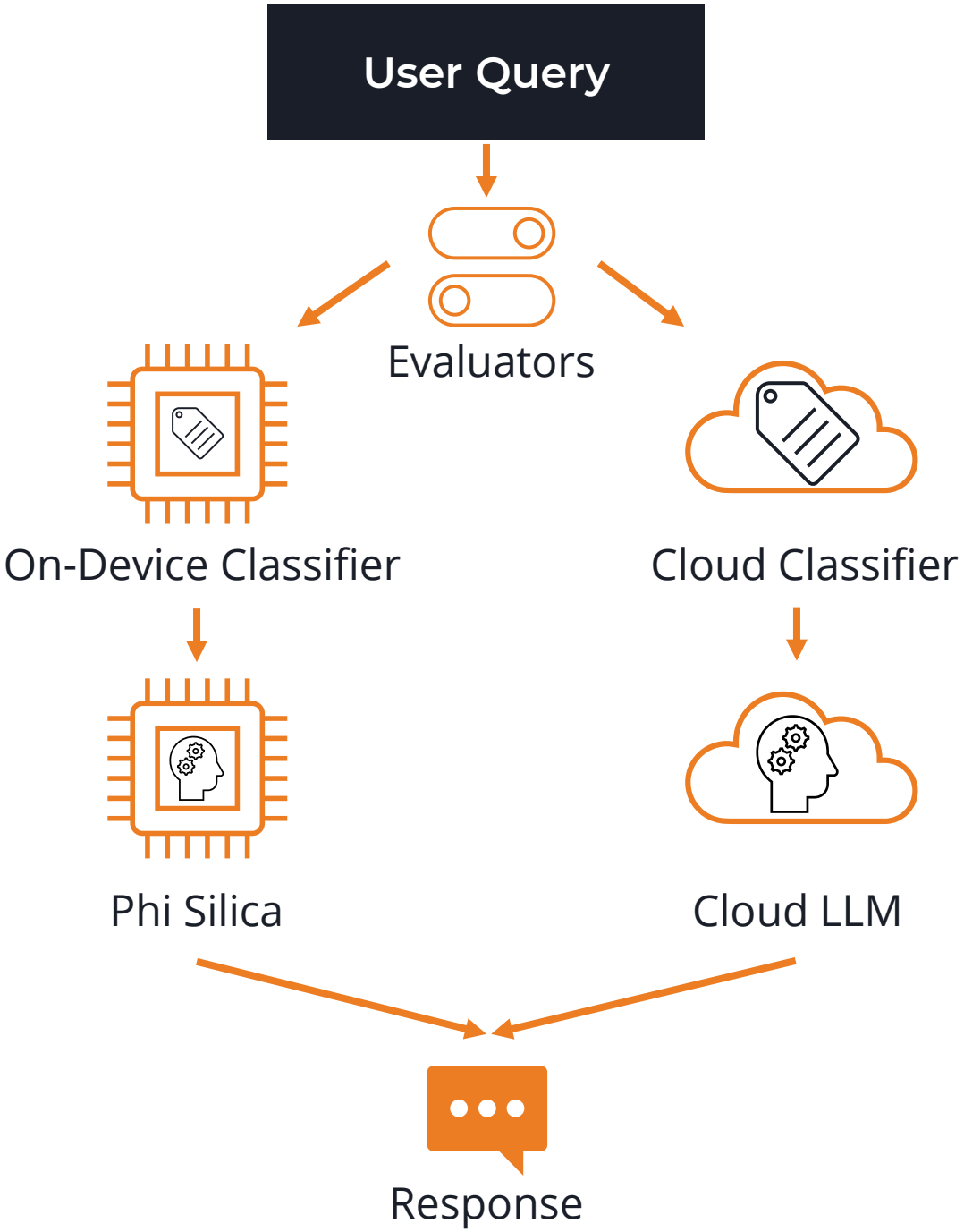
```
input_ids:      [101, 4567, 9876, 0]
attention_mask: [1,   1,    1,    0]
```

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

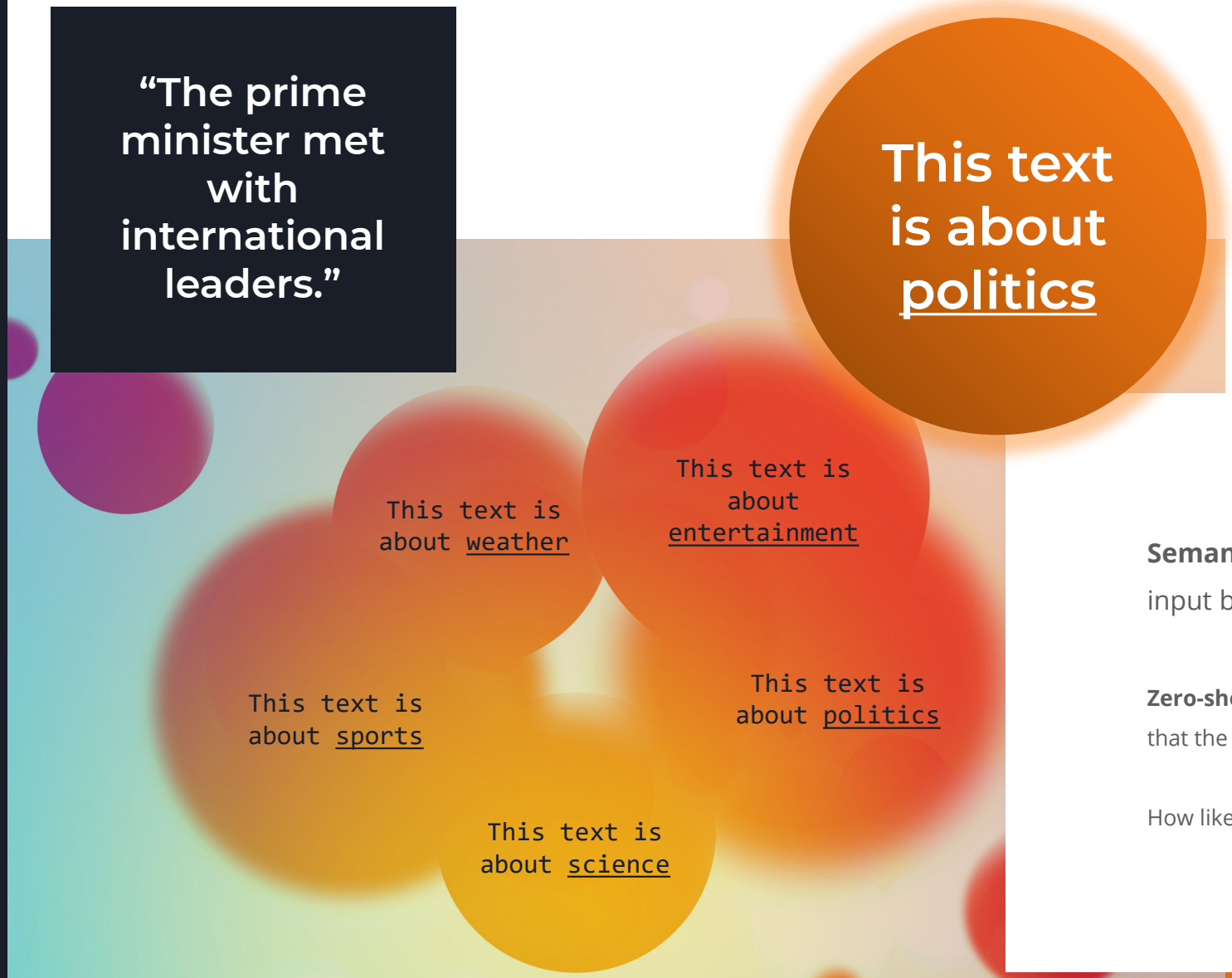
ONNX

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



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

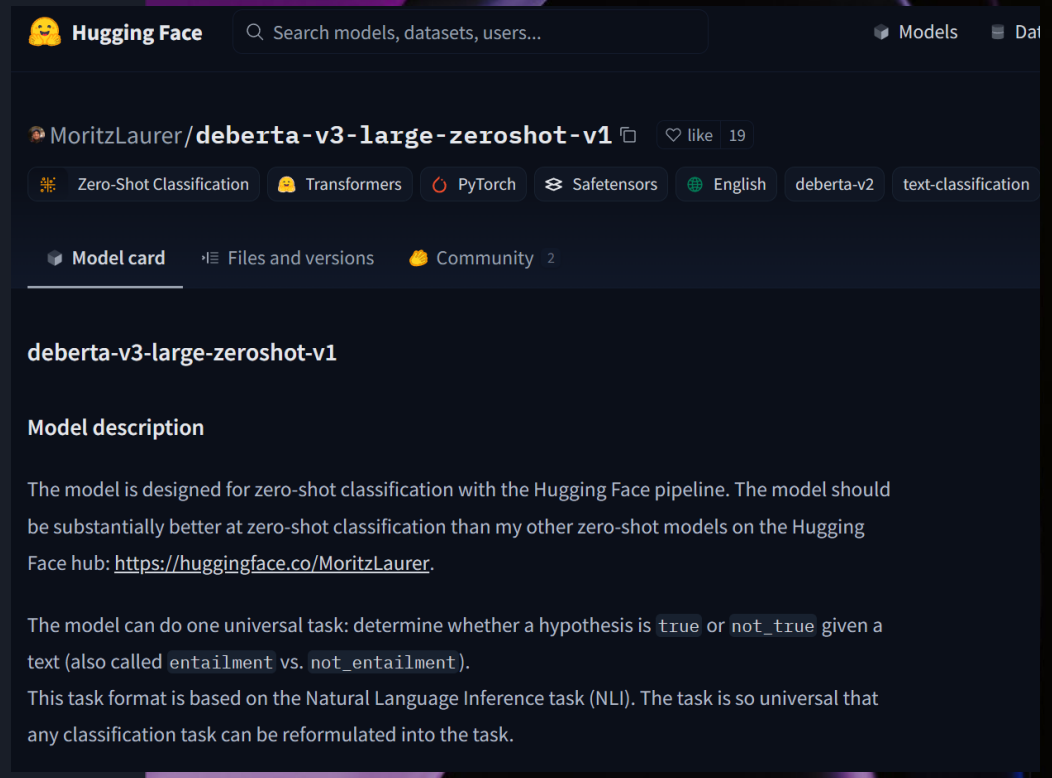
How likely does the input **entail a hypothesis**?

ONNX

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



Jupyter Interlude
→.ONNX

Local Pipeline

```python
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

```python
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
    ]
}
```

ONNX

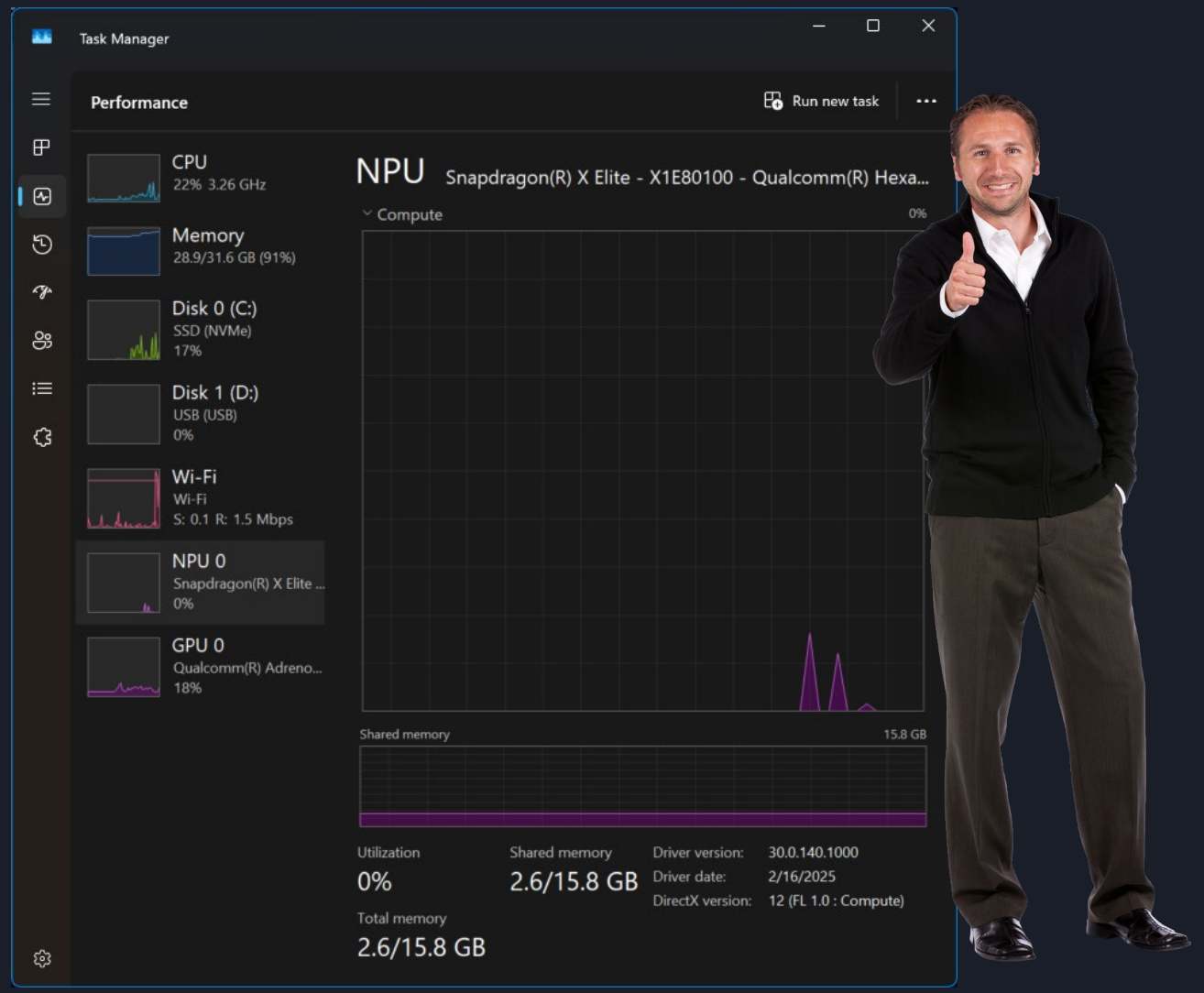
Local ONNX + QNN EP

```python
# 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 ☁️

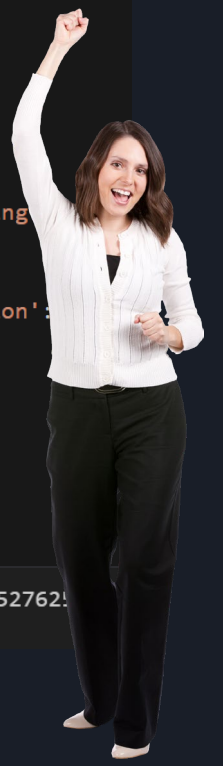
↗️ ml.azure.com

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':

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", 0.062657527625
```



ONNX

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.

You may see (ideally, none, but) miniscule 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 👇

BREAK INCOMING!

Implement Cloud Version

Understand and Work with an ONNX Model

Implement Evaluators

Implement Local Version

CHOOSE YOUR OWN ADVENTURE

Depending on your comfort level...

Build the app with me

Use readymade code snippets as we work through our solution

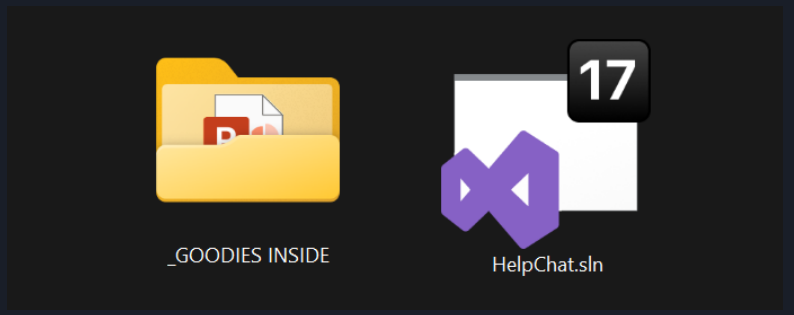
DIY

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



Start Debugging

Files

BREAK!

Checkpoint: Solution and Code Open

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.

1

SHOW A WELCOME MESSAGE

By the end of this step:

- Run the application
- See a generated welcome message

2

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

3

GET CLOUD ONNX RESULT

By the end of this step:

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

4

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.

5

SET UP INTERFACE & FRAMEWORK

Nothing to see; don't run anything just yet

6

IMPLEMENT PRIVACY EVALUATOR

Keep holding your horses

7

IMPLEMENT LATENCY EVALUATOR

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

8

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.

9

GET LOCAL ONNX RESULT

By the end of this step:

- See a message using local classification of the input

10

GET PHI SILICA RESPONSE

By the end of this step:

- Get a response from Phi Silica

END

YOU NOW HAVE A COMPLETE APP

Congratulations!

GIVE IT A WHIRL

Try the solution out with various input statements.

EXTEND/ENHANCE THE SOLUTION

If there's time...

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

TEKNIKOS

GITHUB

Complete Code
This presentation
Jupyter notebook
Score.py

https://bit.ly/hybridailab

JONATHAN KHOO

jonk@teknikos.com