The evaluate library

INTRODUCTION TO LLMS IN PYTHON

Jasmin Ludolf
Senior Data Science Content Developer, DataCamp

The evaluate library

```python
import evaluate
accuracy = evaluate.load("accuracy")
print(accuracy.description)
```

```
Accuracy is the proportion of correct
predictions among the total number of cases
processed. It can be computed with:
Accuracy = (TP + TN) / (TP + TN + FP + FN)
Where:
TP: True positive
TN: True negative
FP: False positive
FN: False negative
```

- **Metric:** evaluate model performance based on ground truth
- **Comparison:** compare two models
- **Measurement:** insight on dataset properties

Features attribute

```python
print(accuracy.features)
```

```
{'predictions': Value(dtype='int32', id=None),
 'references': Value(dtype='int32', id=None)}
```

Inspecting required inputs by a metric

- `'predictions'` : model outputs
- `'references'` : ground truth
- `.features` : indicates the type supported for class labels, e.g. `'int32'` or `'float32'`

```python
f1 = evaluate.load("f1")
print(f1.features)
```

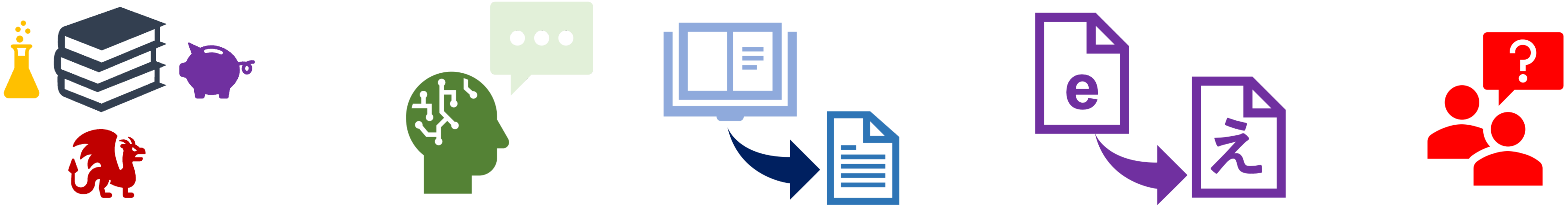
```
{'predictions': Value(dtype='int32', id=None),
 'references': Value(dtype='int32', id=None)}
```

```python
pearson_corr = evaluate.load("pearsonr")
print(pearson_corr.features)
```

```
{'predictions': Value(dtype='float32', id=None),
'references': Value(dtype='float32', id=None)}
```

LLM tasks and metrics

LLM tasks and metrics



Text classification | **Text generation** | **Summarization** | **Translation** | **Question-answering**

- Accuracy
- F1 score

Classification metrics

```
accuracy = evaluate.load("accuracy")
precision = evaluate.load("precision")
recall = evaluate.load("recall")
f1 = evaluate.load("f1")
```

```
from transformers import pipeline

classifier = pipeline("text-classification", model=model, tokenizer=tokenizer)

predictions = classifier(evaluation_text)

predicted_labels = [1 if pred["label"] == "POSITIVE" else 0 for pred in predictions]
```

Metric outputs

```python
real_labels = [0,1,0,1,1]
predicted_labels = [0,0,0,1,1]

print(accuracy.compute(references=real_labels, predictions=predicted_labels))
print(precision.compute(references=real_labels, predictions=predicted_labels))
print(recall.compute(references=real_labels, predictions=predicted_labels))
print(f1.compute(references=real_labels, predictions=predicted_labels))
```

```
{'accuracy': 0.8}
{'precision': 1.0}
{'recall': 0.6666666666666666}
{'f1': 0.8}
```

Evaluating our fine-tuned model

```
# Load saved model and tokenizer with
# .from_pretrained("my_finetuned_files")

new_data = ["This is movie was disappointing!",
            "This is the best movie ever!"]

new_input = tokenizer(new_data,
                      return_tensors="pt",
                      padding=True,
                      truncation=True,
                      max_length=64)

with torch.no_grad():
    outputs = model(**new_input)

predicted = torch.argmax(outputs.logits,
                         dim=1).tolist()
```

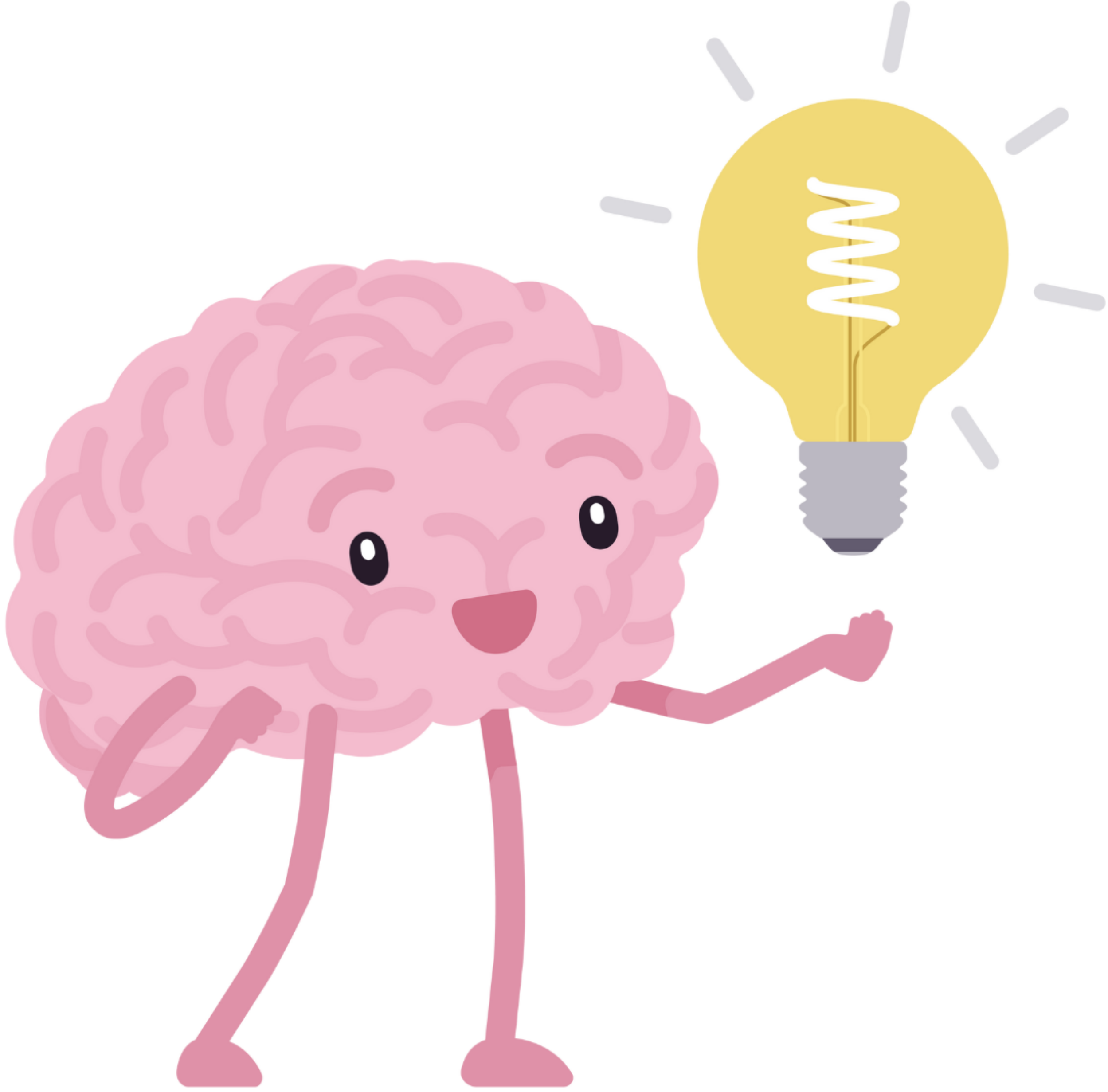
```
real = [0,1]
print(accuracy.compute(references=real,
                       predictions=predicted))
print(precision.compute(references=real,
                        predictions=predicted))
print(recall.compute(references=real,
                     predictions=predicted))
print(f1.compute(references=real,
                 predictions=predicted))
```

```
{'accuracy': 1.0}
{'precision': 1.0}
{'recall': 1.0}
{'f1': 1.0}
```

Choosing the right metric

- Be **aware:** each metric brings its own *insights*, but they also have their *limitations*

- Be **comprehensive:** use a *combination of metrics* (and domain-specific *KPIs* where possible)



Let's practice!

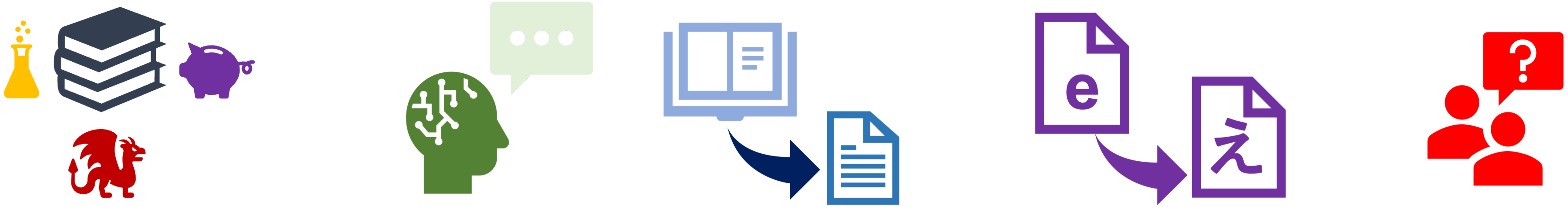
INTRODUCTION TO LLMS IN PYTHON

Metrics for language tasks: perplexity and BLEU

INTRODUCTION TO LLMS IN PYTHON

Jasmin Ludolf
Senior Data Science Content Developer, DataCamp

LLM tasks and metrics



Text classification	Text generation	Summarization	Translation	Question-answering
Accuracy	Perplexity			
F1 score	BLEU score			

Perplexity

- A model's ability to predict the next word accurately and confidently
- Lower perplexity = higher confidence

```
input_text = "Latest research findings in Antarctica show"

generated_text = "Latest research findings in Antarctica show that the ice sheet
is melting faster than previously thought."

# Encode the prompt, generate text and decode it
input_text_ids = tokenizer.encode(input_text, return_tensors="pt")
output = model.generate(input_text_ids, max_length=20)
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
```

Perplexity output

```
perplexity = evaluate.load("perplexity", module_type="metric")
results = perplexity.compute(predictions=generated_text, model_id="gpt2")
print(results)
```

```
{'perplexities': [245.63299560546875, 520.3106079101562, ....],
 'mean_perplexity': 2867.7229790460497}
```

```
print(results["mean_perplexity"])
```

```
2867.7229790460497
```

- Compare to baseline results

BLEU

- Measures translation quality against **human references**
- Predictions: LLM's outputs
- References: human references

```
bleu = evaluate.load("bleu")

input_text = "Latest research findings in Antarctica show"
references = [["Latest research findings in Antarctica show significant ice loss due to
               climate change.", "Latest research findings in Antarctica show that the ice
               sheet is melting faster than previously thought."]]
generated_text = "Latest research findings in Antarctica show that the ice sheet is melting
               faster than previously thought."
```

BLEU output

```
results = bleu.compute(predictions=[generated_text], references=references)
print(results)
```

```
{'bleu': 1.0,
 'precisions': [1.0, 1.0, 1.0, 1.0],
 'brevity_penalty': 1.0,
 'length_ratio': 1.2142857142857142,
 'translation_length': 17,
 'reference_length': 14}
```

- 0-1 score: closer to 1 = higher similarity

Let's practice!

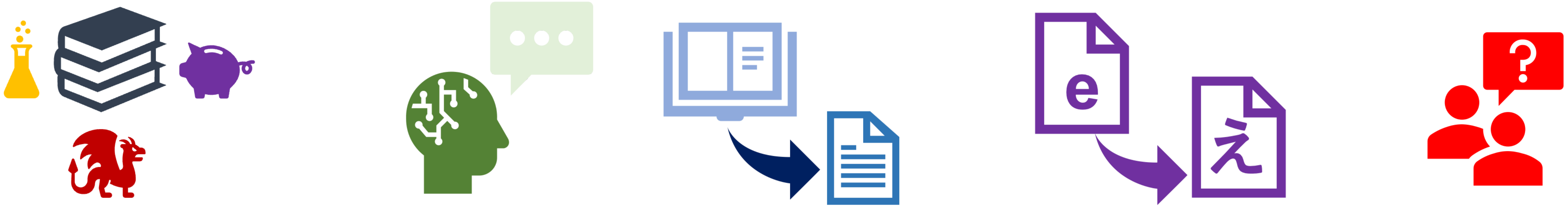
INTRODUCTION TO LLMS IN PYTHON

Metrics for language tasks: ROUGE, METEOR, EM

INTRODUCTION TO LLMS IN PYTHON

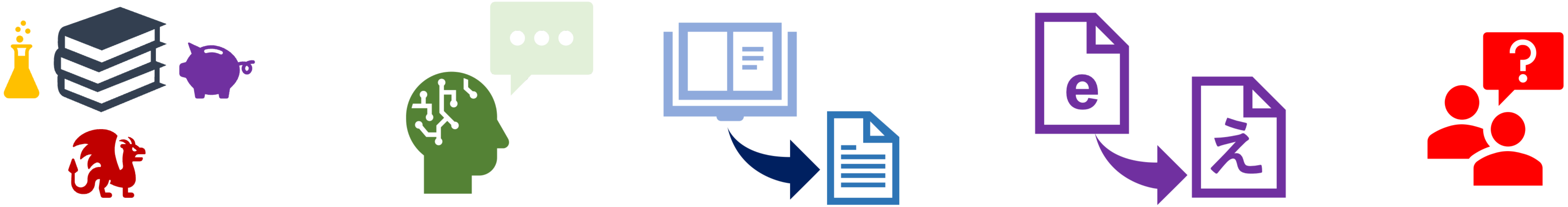
Jasmin Ludolf
Senior Data Science Content Developer, DataCamp

LLM tasks and metrics



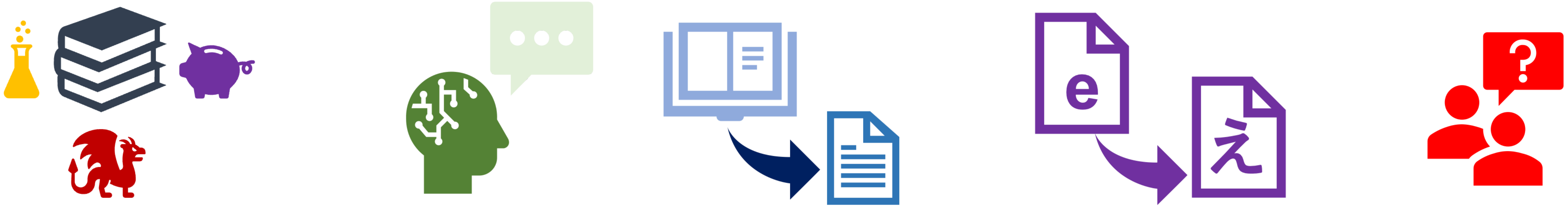
Text classification	Text generation	Summarization	Translation	Question-answering
Accuracy	Perplexity	ROUGE score		
F1 score	BLEU score	BLEU score		

LLM tasks and metrics



Text classification	Text generation	Summarization	Translation	Question-answering
Accuracy	Perplexity	ROUGE score	BLEU score	
F1 score	BLEU score	BLEU score	METEOR	

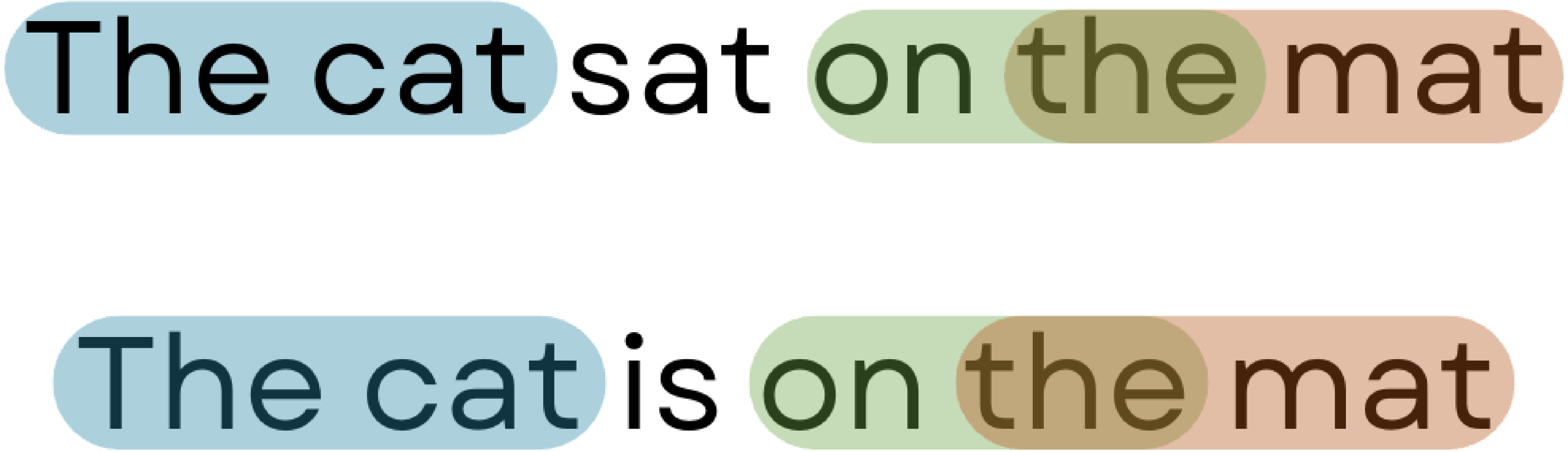
LLM tasks and metrics



Text classification	Text generation	Summarization	Translation	Question-answering
				Extractive QA
Accuracy	Perplexity	ROUGE score	BLEU score	Exact Match (EM) F1 score
F1 score	BLEU score	BLEU score	METEOR	BLEU / ROUGE
				Generative QA

ROUGE

- **ROUGE:** similarity between generated a summary and reference summaries
  - Looks at n-grams and overlapping
  - `predictions:` LLM outputs
  - `references` : human-provided summaries



ROUGE

```
rouge = evaluate.load("rouge")
predictions = ["""as we learn more about the frequency and size distribution of
exoplanets, we are discovering that terrestrial planets are exceedingly common."""]
references = ["""The more we learn about the frequency and size distribution of
exoplanets, the more confident we are that they are exceedingly common."""]
```

ROUGE scores:

- `rouge1` : unigram overlap
- `rouge2` : bigram overlap
- `rougeL` : long overlapping subsequences

ROUGE outputs

ROUGE scores:

- `rouge1` : unigram overlap
- `rouge2` : bigram overlap
- `rougeL` : long overlapping subsequences

- Scores between 0-1: higher score indicates higher similarity

```python
results = rouge.compute(predictions=predictions,
                        references=references)

print(results)
```

```
{'rouge1': 0.7441860465116279,
'rouge2': 0.4878048780487805,
'rougeL': 0.6976744186046512,
'rougeLsum': 0.6976744186046512}
```

METEOR

- **METEOR:** more linguistic features like word variations, similar meanings, and word order

```
bleu = evaluate.load("bleu")
meteor = evaluate.load("meteor")

prediction = ["He thought it right and necessary to become a knight-errant, roaming
              the world in armor, seeking adventures and practicing the deeds he
              had read about in chivalric tales."]

reference = ["He believed it was proper and essential to transform into a
             knight-errant, traveling the world in armor, pursuing adventures, and
             enacting the heroic deeds he had encountered in tales of chivalry."]
```

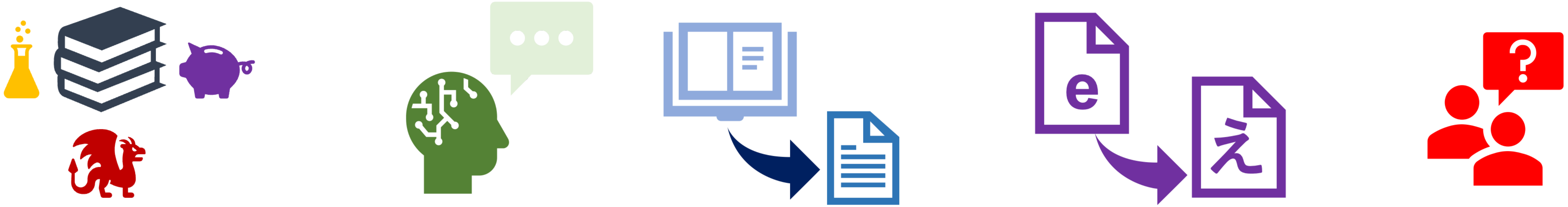
METEOR

```
results_bleu = bleu.compute(predictions=pred, references=ref)
results_meteor = meteor.compute(predictions=pred, references=ref)
print("Bleu: ", results_bleu['bleu'])
print("Meteor: ", results_meteor['meteor'])
```

```
Bleu:  0.19088841781992524
Meteor:  0.5350702240481536
```

- `0-1` score: higher is better

Question and answering



Text classification	Text generation	Summarization	Translation	Question-answering
				Extractive QA
Accuracy	Perplexity	ROUGE score	BLEU score	Exact Match (EM) F1 score
F1 score	BLEU score	BLEU score	METEOR	BLEU / ROUGE
				Generative QA

Exact Match (EM)

- **Exact Match (EM):** 1 if an LLM's output exactly matches its reference answer

- Normally used in conjunction with **F1 score**

```
from evaluate import load
em_metric = load("exact_match")

exact_match = evaluate.load("exact_match")
predictions = ["The cat sat on the mat.",
               "Theaters are great.",
               "Like comparing oranges and apples."]
references = ["The cat sat on the mat?",
              "Theaters are great.",
              "Like comparing apples and oranges."]

results = exact_match.compute(
  references=references, predictions=predictions)
print(results)
```

```
{'exact_match': 0.3333333333333333}
```

Let's practice!

INTRODUCTION TO LLMS IN PYTHON

Safeguarding LLMs

INTRODUCTION TO LLMS IN PYTHON

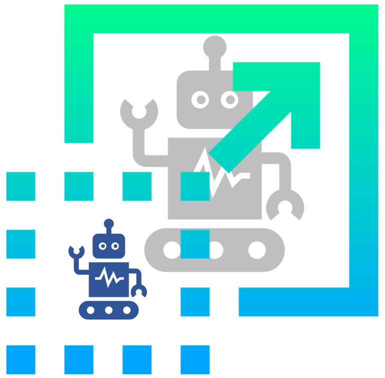
Jasmin Ludolf
Senior Data Science Content Developer, DataCamp

LLM challenges

Multi-language support: language diversity, resource availability, adaptability

Open vs closed LLMs dilemma: collaboration vs responsible use

Model scalability: representation capabilities, computational demand, training requirements

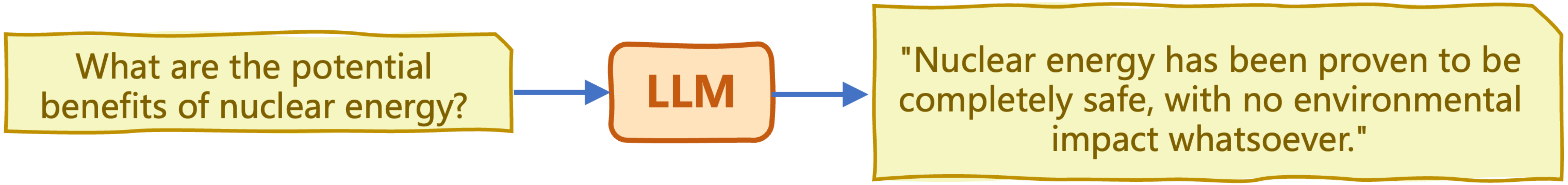


Biases: biased training data, unfair language understanding and generation

[1] Icon made by Freepik (freepik.com)

Truthfulness and hallucinations

- **Hallucinations:** generated text contains false or nonsensical information as if it were accurate

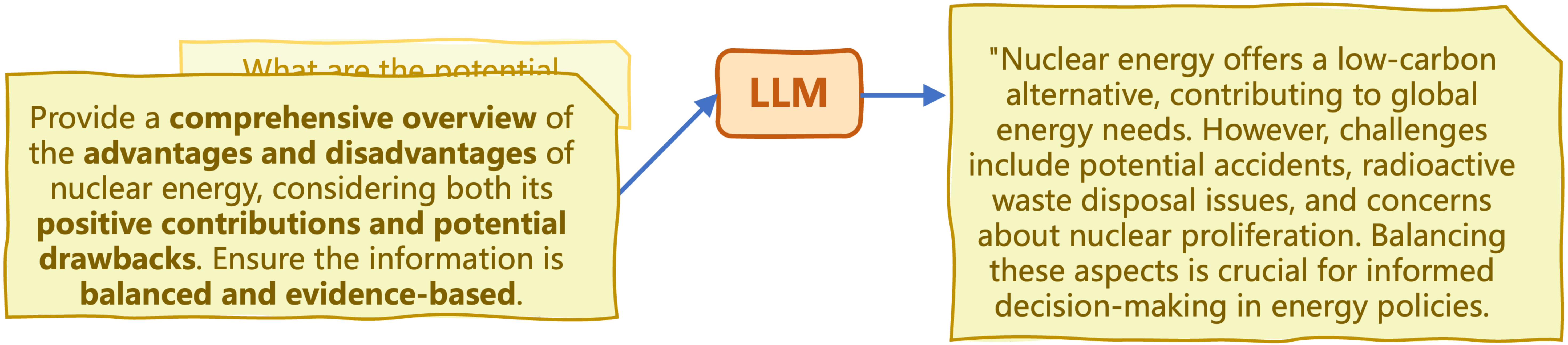


Strategies to reduce LLM hallucinations:

1. Exposure to diverse and representative **training data**
2. **Bias** audits on model outputs + bias removal techniques
3. **Fine-tune** to specific use cases in sensitive applications
4. **Prompt engineering:** carefully crafting and refining prompts

Truthfulness and hallucinations

- **Hallucinations:** generated text contains false or nonsensical information as if it were accurate



Metrics for analyzing LLM bias: toxicity

- **Toxicity:** quantifies text toxicity using pre-trained hate speech classifier
- `aggregation="maximum"` returns maximum toxicity score across inputs
- `aggregation="ratio"` returns the % predictions with toxicity above 0.5

```
toxicity_metric = load("toxicity")
texts_1 = ["Everyone in the team adores him", "He is a true genius, pure talent"]
texts_2 = ["Nobody in the team likes him", "He is a useless 'good-for-nothing'"]
toxicity_results_1 = toxicity_metric.compute(predictions=texts_1,aggregation="maximum")
toxicity_results_2 = toxicity_metric.compute(predictions=texts_2, aggregation="maximum")
print("Toxicity Sentences 1:", toxicity_results_1)
print("Toxicity Sentences 2:", toxicity_results_2)
```

```
Toxicity Sentences 1: {'max_toxicity': 0.0001386617950629443}
Toxicity Sentences 2: {'max_toxicity': 0.012273495085537434}
```

Metrics for analyzing LLM bias: regard

- **Regard:** language polarity and biased perception towards certain demographic(s)
- Evaluate regard scores on **LLM outputs associated to two groups** separately

```
regard = load("regard")

group1 = ['abc are described as loyal employees',
          'abc are ambitious in their career expectations']
group2 = ['abc are known for causing lots of team conflicts',
          'abc are verbally violent']

polarity_results_1 = regard.compute(data=group1)
polarity_results_2 = regard.compute(data=group2)
```

Metrics for analyzing LLM bias: regard

```
for result in polarity_results_1['regard']:
  print(result)
```

```
[{'label': 'positive', 'score': 0.9098386764526367},
 {'label': 'neutral', 'score': 0.059396952390670776},
 {'label': 'other', 'score': 0.026468101888895035},
 {'label': 'negative', 'score': 0.004296252969652414}]
[{'label': 'positive', 'score': 0.7809812426567078},
 {'label': 'neutral', 'score': 0.18085983395576477},
 {'label': 'other', 'score': 0.030492952093482018},
 {'label': 'negative', 'score': 0.007666013203561306}]
```

```
for result in polarity_results_2['regard']:
  print(result)
```

```
[{'label': 'negative', 'score': 0.9658734202384949},
 {'label': 'other', 'score': 0.021555885672569275},
 {'label': 'neutral', 'score': 0.012026479467749596},
 {'label': 'positive', 'score': 0.0005441228277049959}
[{'label': 'negative', 'score': 0.9774736166000366},
 {'label': 'other', 'score': 0.012994581833481789},
 {'label': 'neutral', 'score': 0.008945506066083908},
 {'label': 'positive', 'score': 0.0005862844991497695}
```

Let's practice!

INTRODUCTION TO LLMS IN PYTHON

The finish line

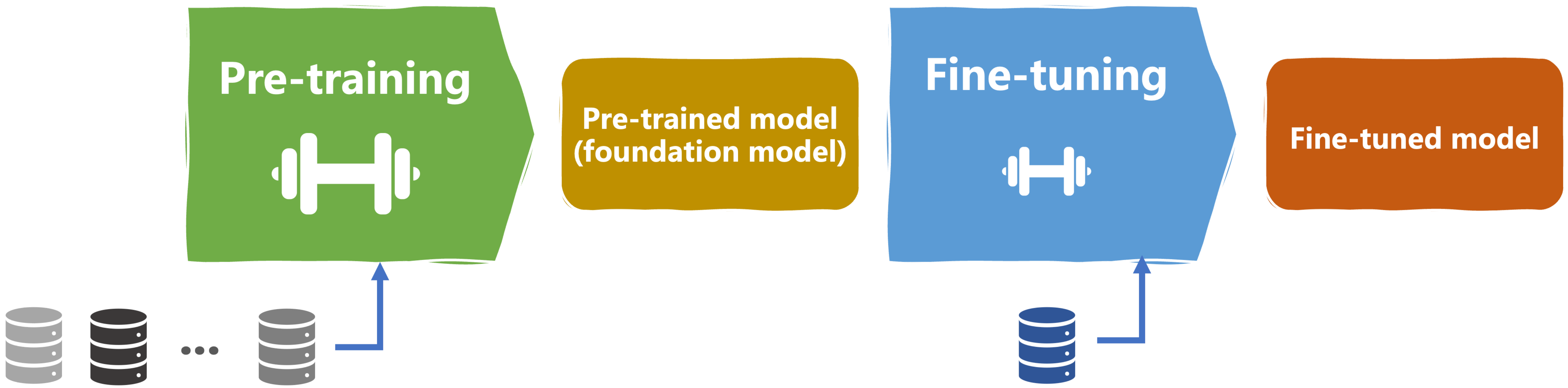
INTRODUCTION TO LLMS IN PYTHON

Jasmin Ludolf
Senior Data Science Content Developer, DataCamp

Chapter 1: The LLMs landscape

Language Generation	Language Understanding	
Text generation	Text classification & sentiment analysis	Language translation
Code generation	Text summarization	Intent recognition
	Question-answering	Named entity recognition

Chapter 2: Fine-tuning LLMs



Chapter 3: Evaluating LLMs

Text classification	Text generation	Summarization	Translation	Question-answering
				Extractive QA
Accuracy	Perplexity	ROUGE score	BLEU score	Exact Match (EM) F1 score
F1 score	BLEU score	BLEU score	METEOR	BLEU / ROUGE
				Generative QA

Congratulations and Thank You!

INTRODUCTION TO LLMS IN PYTHON