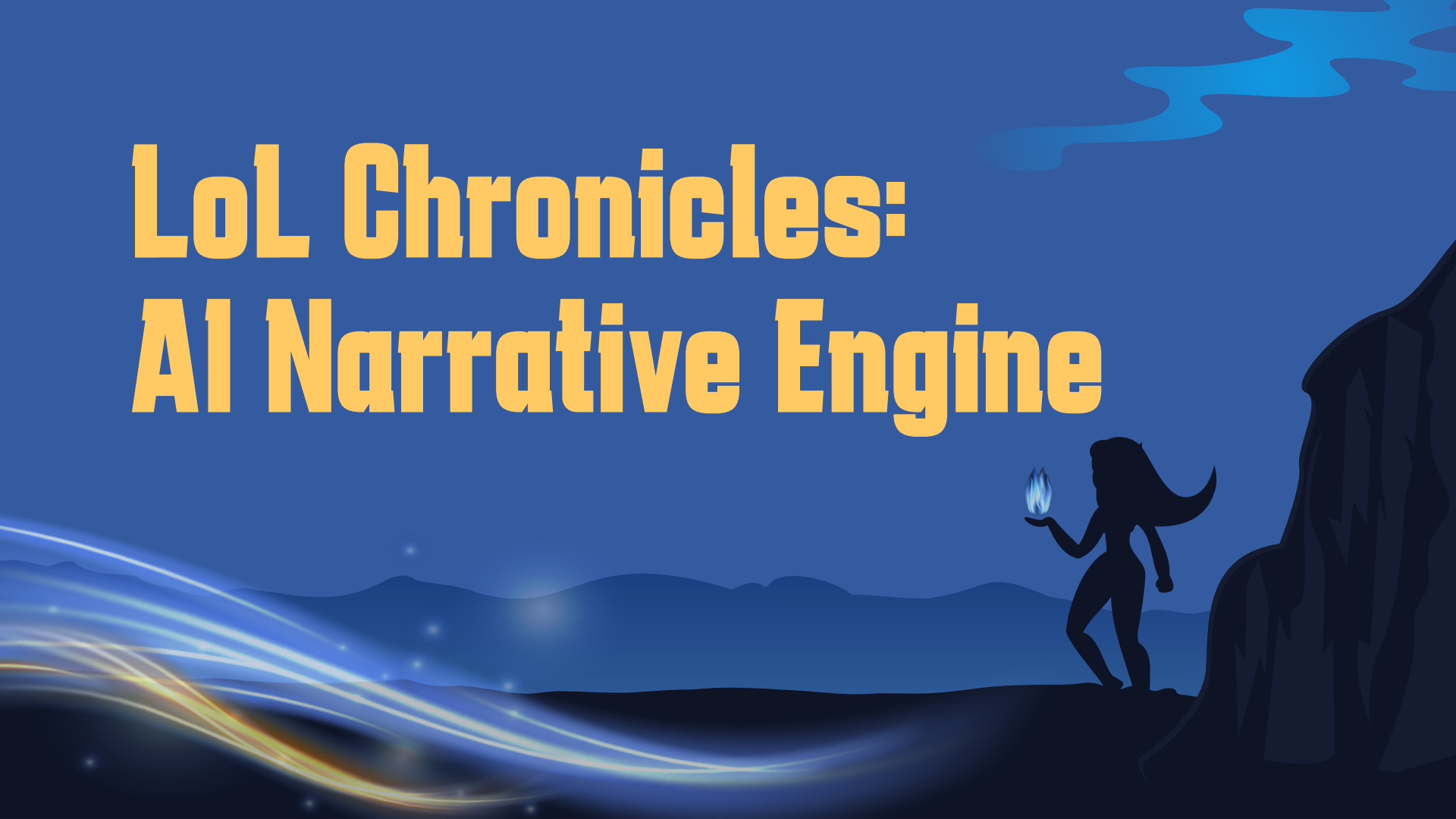
LoL Chronicles: AI Narrative Engine

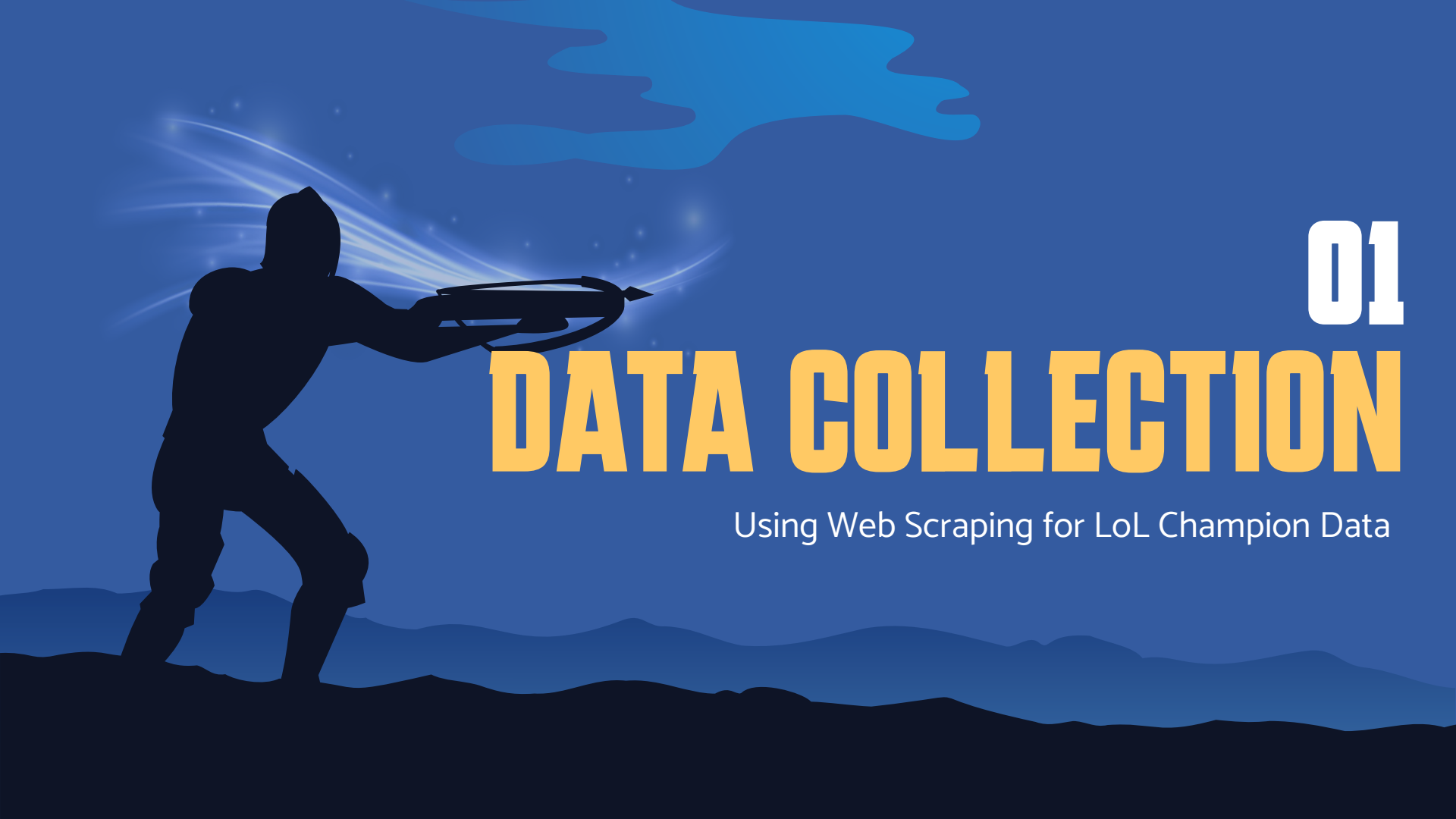


Our Objective

Problem Statement: Creating consistent, engaging content for 170+ champions is time-consuming and requires deep lore knowledge

Core Objectives:

1. Develop a specialized language model for League of Legends narrative generation
2. Create contextually appropriate content consistent with character personalities
3. Build a tool for efficient lore-consistent content creation



01
DATA COLLECTION

Using Web Scraping for LoL Champion Data

Approach

ESSENTIAL DATA

Name, region, role, race, quote, short bio, related champions, biography, story

WHY SCRAPING?

Structured data not readily available, wanted more accurate information directly from the official lore

TOOLS

Python and Selenium

Challenges and Solutions

Challenge A

Dynamic Content Loading

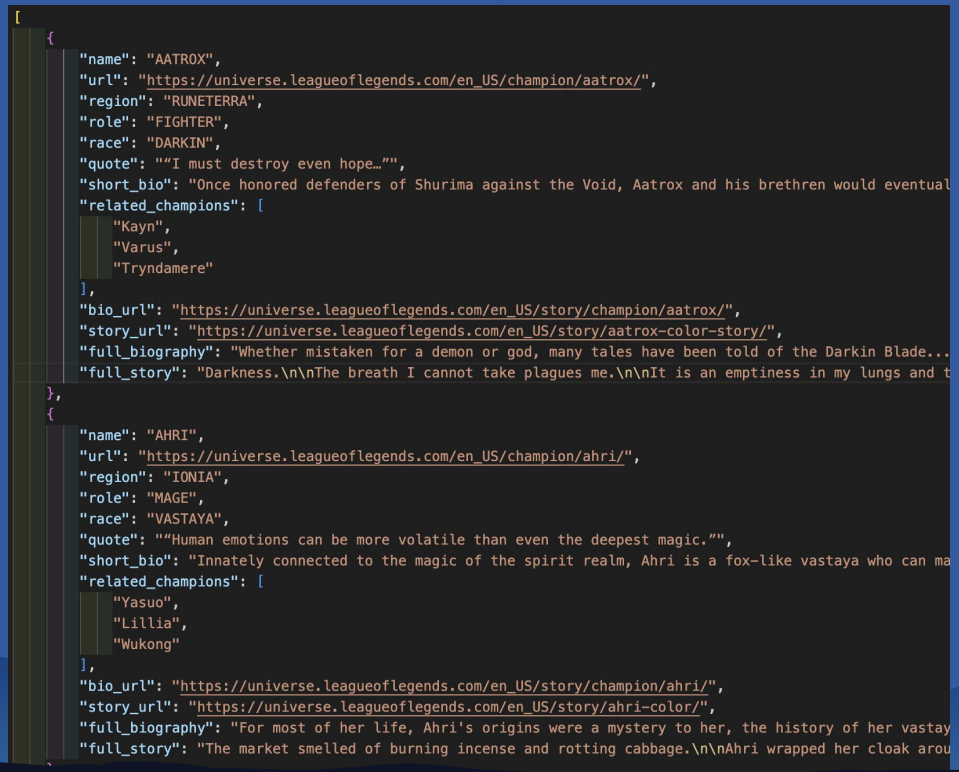
Solution: Selenium for browser automation that is JavaScript-based

Challenge B

Inconsistent HTML

Multiple CSS selector strategies with graceful fallbacks

170 Champions Successfully Scraped!



02

Fine-tuning

Training a Specialized Model for LoL Narratives

Fine-tuning Steps

01

Model Selection

Select the Base Model

02

Data Tokenization

Tokenize the Scraped Data

03

Model Fine-tuning

Select Hyperparameters and Fine-tune in the Cloud

Model Selection: GPT2-XL

GP2-XL is our base model for fine-tuning:

- **Parameters:** 1.5B
- **Source:** Hugging Face

Rational for model selection:

1. Open-weight & permissive MIT license
2. Decoder-only architecture for story generation
3. Trainable locally

Tokenization

Goal:

- Transform rows of fine-tuning text data (1 story per row) into rows of **Byte-Pair Encoding** and **Attention Mask**

```
{
    'text':   'Ahri whispered an ancient spell...'
}
```



```
{
    'input_ids':      tensor([502, 345, ..., 50256, 50256]),
    'attention_mask': tensor([  1,   1, ...,     0,     0])
}
```

Tokenization

- Loads the original tokenizer
- Use end-of-sentence token (ID 50256) for padding

- Tokenization consideration
  - Truncate at 1024 tokens
  - Pad shorter inputs to 1024 tokens

- Tokenize

```
tokenizer = AutoTokenizer.from_pretrained(
    "models/gpt2-xl"
)
tokenizer.pad_token = tokenizer.eos_token

ds = Dataset.from_dict({"text": texts})

# GPT-2's max context length is 1024
context_length = 1024

def tokenize(batch):
    return tokenizer(
        batch["text"],
        truncation=True,
        max_length=context_length,
        padding = "max_length"
    )

ds_tok = ds.map(
    tokenize, batched=True, remove_columns=["text"]
)
ds_tok.set_format(
    type="torch", columns=["input_ids", "attention_mask"]
)
ds_tok
```

Fine-tuning

Goal:

- Change model weights to generate better stories for League of Legends

Hyperparameters for Best Performance

- per_device_train_batch_size = 1
- gradient_accumulation_steps = 8
- Learning_rate = 5e-5
- Epochs = 3

Fine-tuning

1. Forward Pass

- Embedding (B, T, D)
- 25 attention heads/block
  - Causal masking
- 48 blocks
- Logits layer

2. Loss Calculation

$$\mathcal{L}_{\text{batch}} = \frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_i$$

$$= -\frac{1}{B}\sum_{i=1}^{B}\log\left(\frac{\exp(z_{i,c_i})}{\sum_{k=1}^{|V|}\exp(z_{i,k})}\right)$$

3. Backward Pass

- Loss gradient
- Chain rule all the way back
- Store gradients

Every 8 Cycles

Optimizer Step: weights updated

Final Loss: 2.45

Fine-tuning - Cloud Setup

Modal

- Create an Modal app
- Set up an image for the container

- Specify GPU type,
- Set up volumes (mount data & save model artifacts)

- **Deploy - up to 10 concurrent H100s**

```python
fine_tune_app = modal.App("gpt2-xl-ft")
ft_image = modal.Image.debian_slim().pip_install(
    "transformers", "datasets", "accelerate",
    "bitsandbytes", "peft", "torch"
)
GPU = "H100"
vol = modal.Volume.from_name("data")
model_vol = modal.Volume.from_name(
    "gpt2_ft", create_if_missing=True
)

@fine_tune_app.function(
    image=ft_image,
    gpu=GPU,
    timeout = 3600,
    volumes={"/data": vol, "/checkpoints": model_vol}
)
def train_model():
```

03

Evaluation

Using LLM-as-a-Judge for Evaluation

Model Evaluation: LLM as a Judge

Gemini 2.0 Flash

01	02	03	04
Coherence	**Relevance**	**Narrative**	**Originality**
Logical Flow	Stay on Topic	Engaging & Well Written	Novelty & Creativity

Story Generation Prompt

```
# 5 prompts
prompts = [
    "Ashe, the Frost Archer, stalks her prey in the icy forests of Freljord. Continue the story:",
    "Zed, the Master of Shadows, prepares his ultimate strike at dusk. Continue the tale:",
    "Soraka, the Starchild, descends to heal the wounded on the Fields of Justice. Continue:",
    "Dr. Mundo, the Madman of Zaun, rampages through the chem-labs. Continue:",
    "Jinx, the Loose Cannon, plots her next explosive prank in Piltover. Continue:"
]
```

Base Story

How it Began: The Night of Nightmares is a three-part short-story adventure for 5e D&D, written by Jonathan Tweet.

There are no monsters or PCs in this episode—the plot kicks off in a tavern,

where the party has been hired by a wealthy merchant to retrieve a shipment of goods.

As they explore the dark warehouse, the ship's cargo proves far more sinister than anyone expected.

Result:

The baseline model drifts into tabletop-RPG promo text instead of League lore.

Fine-Tuned Story

Zed the Illuminator

Trundle’s apprentice, Zed the Illuminator, was destined to master the dark arts—yet in a desperate quest to save the dying world of Shurima, he unleashed an arcane plague that turned its denizens into bloodthirsty monsters.

For centuries, Zed studied and honed the ancient magic of shadow.

His master’s instructions were clear: draw no attention, invite no gaze of the living.

Such was the decree of the elders’ great teacher, Master Trundle.

Result:

Emphasize lore consistency: Shurima setting, Trundle reference, shadow-magic tone – clear improvement over baseline.

Evaluation Generation Prompt

```
instruction = (
    "You are a story evaluator. For the given input prompt and its generated "
    "story, rate each of the following on a scale of 1 (worst) to 5 (best) "
    "and return ONLY a JSON object with keys:\n"
    '  "coherence": …,          # logical flow\n'
    '  "relevance": …,          # stays on topic\n'
    '  "narrative_quality": …, # engaging & well-written\n'
    '  "originality": …         # novelty / creativity\n\n'
    f"Input Prompt:\n{prompt}\n\nGenerated Story:\n{story}"
)
```

Metric	Base GPT-2	Fine-tuned	Δ
Coherence	2.4	3.7	+1.3
Relevance	2.0	4.1	**+2.1**
Narrative Quality	2.6	3.8	+1.2
Originality	2.1	3.5	+1.4

Limitations and Next Steps

Limitations:

- Context window constraints
- Evolving lore
- LLM as a Judge evaluation blind spots

Next Steps

- Fine-tune larger models
- Real-time lore data pipeline
- Creator-friendly tools

THANKS!

GitHub: Link

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**