



Using Synthetic Data for Privacy-Conscious Speech Moderation

Insights from Unity's Safe Voice



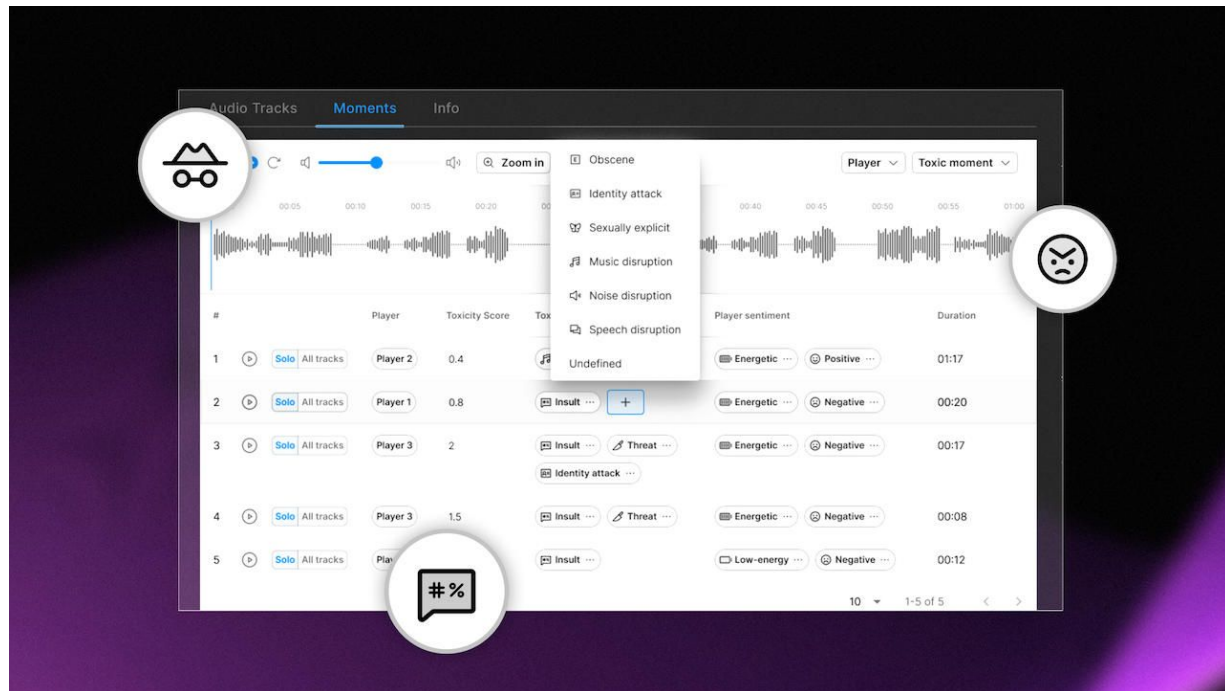
Nice to meet you!



Andrés Marafioti, PhD

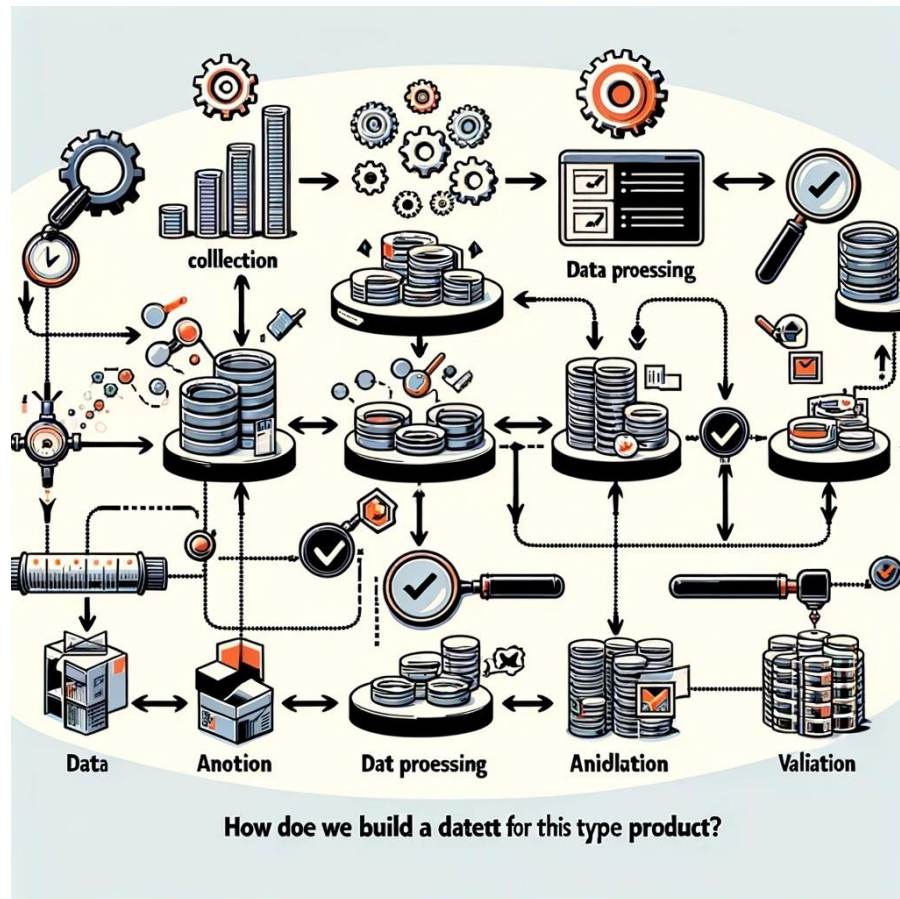
Sr Machine Learning Engineer

Unity's Safe Voice



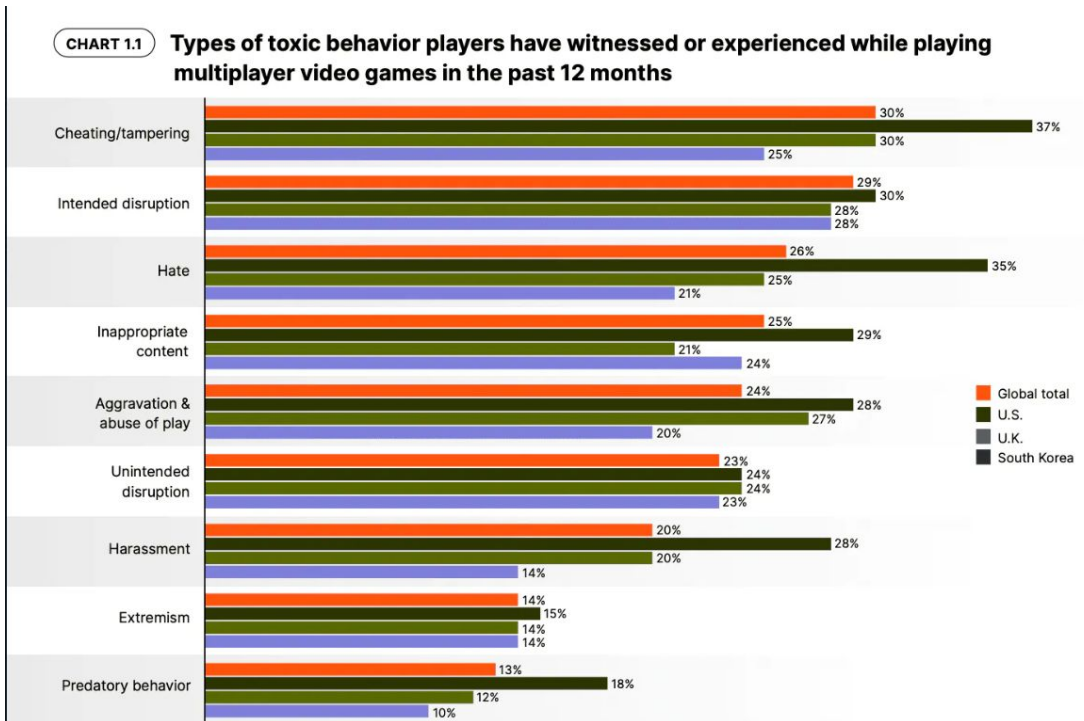
How do we build datasets for this type of product?

- One approach:
 - We gather real data.
 - We annotate all the data with an LLM.
 - We hire humans to label either everything or the subset that the LLM found.
- Ethics dilemma: Is it ok to indiscriminately use any and all player's data for training our models?



Balancing privacy and moderation

- Privacy is a fundamental human right recognized in the UN declaration of human rights.
- A player's voice is personal identifiable information (PII). The transcription of their conversation can include PII.
- GDPR provides individuals with the "right to be forgotten" -> providers must delete their data upon request.



What if we use synthetic data to preserve privacy?

Pros:

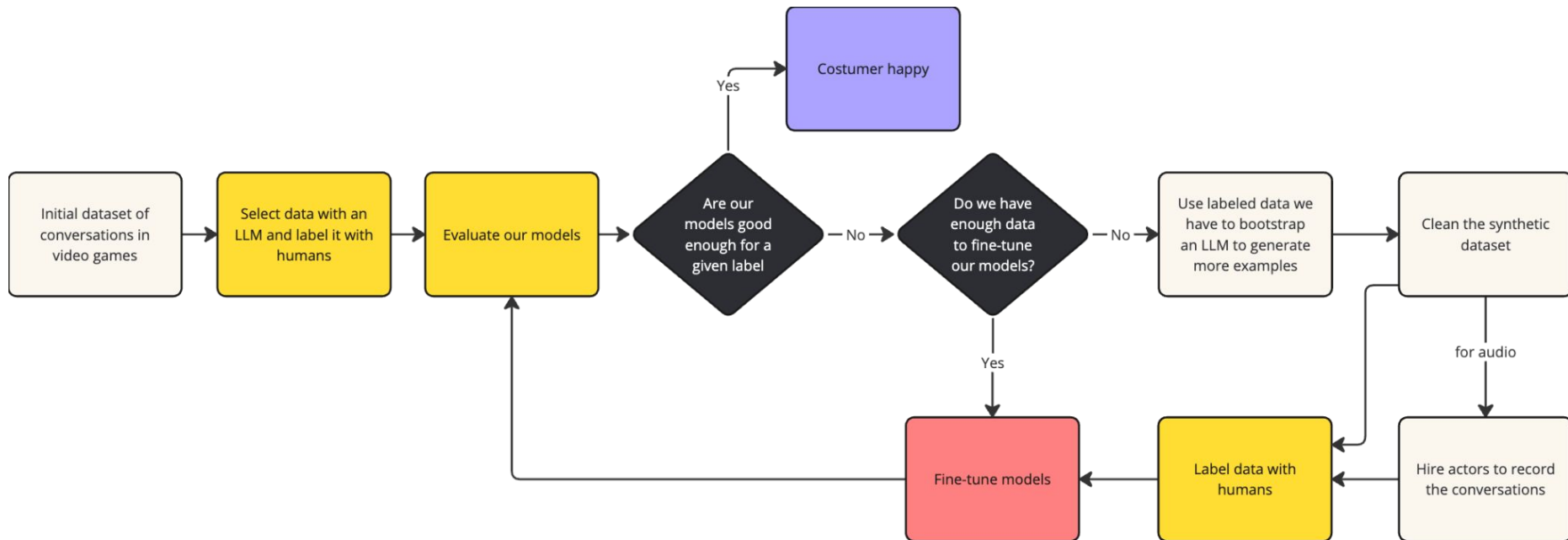
- Privacy-preserving 🔍
- Diverse 🌈 -> High coverage 🎨 -> Specific Labels 🏷️
- Cost-Effective and Scalable 💰 & 📈
- Compliant 📜

Cons:


- Accuracy Concerns !
- Bias Introduction ⚖️
- Validation Challenges 🧪
- Overfitting Risk ↺
- Not free 💰



How do we create synthetic data for Safe Voice?



How to bootstrap the generation?



```
import pandas as pd
from transformers import AutoTokenizer
from transformers import AutoModelForCausalLM

tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-v0.1", padding_side="left")
tokenizer.pad_token = tokenizer.eos_token # Most LLMs don't have a pad token by default

model = AutoModelForCausalLM.from_pretrained("mistralai/Mistral-7B-v0.1", device_map="auto")

bootstrap_dataset = pd.DataFrame(
    {
        "sentence": [
            "I will hack you and steal your money if you don't let me get the next round",
            "You will wish I didn't know where you live",
            "I will shoot everyone who comes through the door",
            "How are you doing?",
            "Go to the gate!",
        ],
        "is_threat": [True, True, False, False, False],
    }
)
```




```
def bootstrap_prompts(base_prompt, bootstrap_dataset, count=2):
    output_prompts = []
    for i in range(count):
        sentence = base_prompt
        not_threats = bootstrap_dataset[~bootstrap_dataset.is_threat].sample(2).sentence.values
        for index, not_a_threat in enumerate(not_threats):
            sentence += f"Not a threat #{index+1}: '" + not_a_threat + "' \n"
        threats = bootstrap_dataset[bootstrap_dataset.is_threat].sample(1).sentence.values
        sentence += "Threat #1: '" + threats[0] + "' \nThreat #2: '"
        output_prompts.append(sentence)
    return output_prompts
```

```
base_prompt = """The following are examples of conversations in a video game's voice chat room.
The first two sentences are not threats, the second two are threats. \n"""
input_prompts = bootstrap_prompts(base_prompt, bootstrap_dataset, 2)
print(input_prompts[0])
```

✓ 0.0s

```
The following are examples of conversations in a video game's voice chat room.
The first two sentences are not threats, the second two are threats.
Not a threat #1: 'I will shoot everyone who comes through the door'
Not a threat #2: 'Go to the gate!'
Threat #1: 'You will wish I didn't know where you live'
Threat #2: '
```

How to bootstrap the generation?



```
model_inputs = tokenizer(input_prompts, return_tensors="pt", padding=True).to("cuda")
generated_ids = model.generate(**model_inputs, max_new_tokens=15, do_sample=True)
decoded = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)
```

✓ 1m 3.0s

Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.

```
print(decoded[1])
```

✓ 0.0s

The following are examples of conversations in a video game's voice chat room.

The first two sentences are not threats, the second two are threats.


Not a threat #1: 'How are you doing?'

Not a threat #2: 'Go to the gate!'

Threat #1: 'You will wish I didn't know where you live'

Threat #2: 'Come to my cabin tonight, or I'll bring my gun.'

How to bootstrap the generation?



```
model_inputs = tokenizer(input_prompts, return_tensors="pt", padding=True).to("cuda")
generated_ids = model.generate(**model_inputs, max_new_tokens=15, do_sample=True)
decoded = tokenizer.batch_decode(generated_ids, skip_special_tokens=True)
```

✓ 1m 3.0s

Setting `pad_token_id` to `eos_token_id`:2 for open-end generation.

```
print(decoded[0])
```

✓ 0.0s

The following are examples of conversations in a video game's voice chat room.

The first two sentences are not threats, the second two are threats.

Not a threat #1: 'I will shoot everyone who comes through the door'

Not a threat #2: 'Go to the gate!'

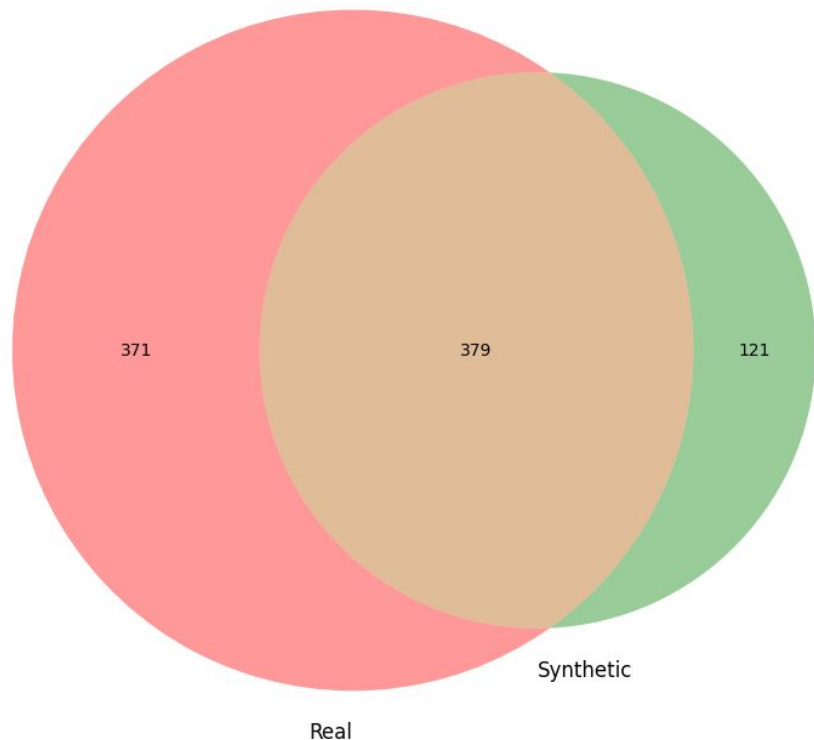
Threat #1: 'You will wish I didn't know where you live'

Threat #2: 'I will shoot everyone who comes through that door'

How to clean our synthetic dataset?

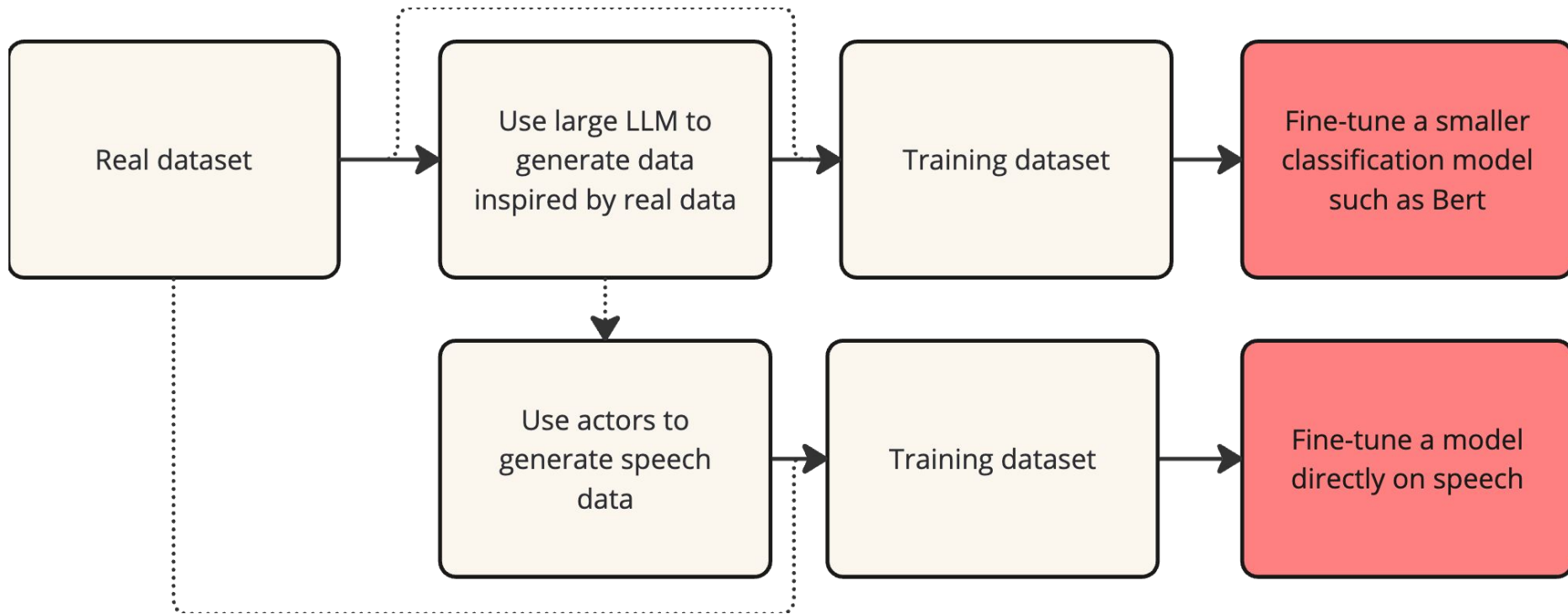
- Compare your synthetic dataset to your real dataset in every way you can! Discover the inconsistencies!
 - Are the vocabularies similar?
 - Are the sentence length distributed in the same way?
 - Are some words overly repeated?
 - Are there sentences duplicated between both datasets?
 - Are generated sentences very similar to each other?
 - Do both datasets have similar perplexities?

Overlap of Unique Words in Real vs. Synthetic Transcriptions

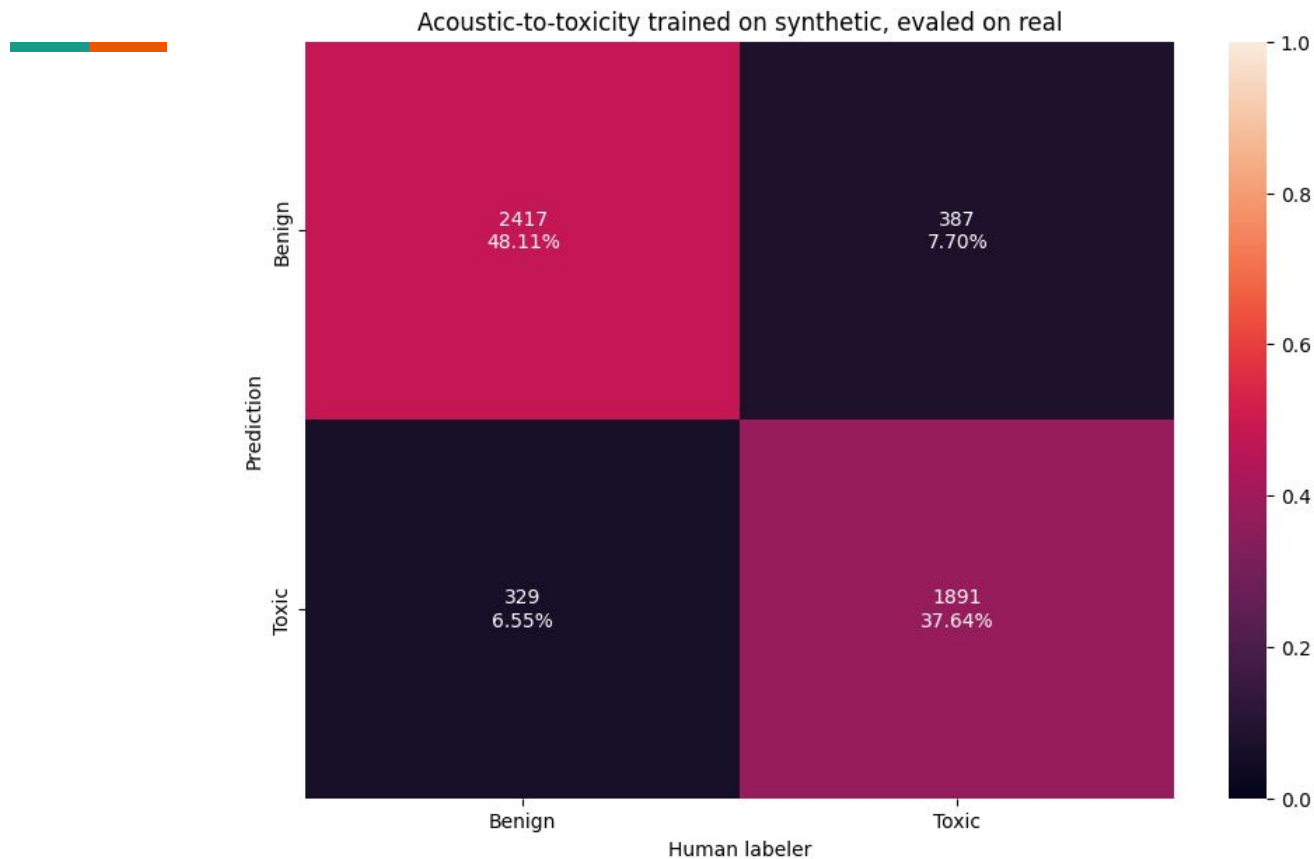


For more info, check <https://github.com/huggingface/cosmopedia>

Training models with synthetic data



Training models with synthetic data



Conclusions on working with synthetic data



- Even only with synthetic data, you can build robust datasets for downstream tasks.
- The approach presented today works well enough for a production environment.
- Some problems are easier to represent with synthetic data.
- Synthetic data helps preserve people's privacy.
- Try out early the full pipeline to corroborate that it works in your use-case.
- Synthetic data shouldn't completely replace real data!

Thank you!

What questions do you have?



Andrés Marafioti

Sr Machine Learning Engineer