```
!pip install transformers pandas seaborn matplotlib tqdm torch --quiet
```

```
import torch
import transformers
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from tqdm import tqdm

%matplotlib inline
%config InlineBackend.figure_format = 'retina'
```

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("gpt2")
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/s
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 368B/s]
config.json: 100%  665/665 [00:00<00:00, 10.6kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 6.98MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 4.85MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 19.2MB/s]
```

```
import torch
import transformers
```

```
model_names = {
    "gpt2-small": "gpt2",
    "gpt2-medium": "gpt2-medium",
    "pythia-70m": "EleutherAI/pythia-70m"
}

models = {}
tokenizers = {}

for name, hf_name in model_names.items():
    print(f"Loading model: {name}")
    tokenizers[name] = transformers.AutoTokenizer.from_pretrained(hf_name)
    models[name] = transformers.AutoModelForCausalLM.from_pretrained(hf_name)
    models[name].eval()
```

```
Loading model: gpt2-small
Loading model: gpt2-medium
```

```
tokenizer_config.json: 100%  26.0/26.0 [00:00<00:00, 513B/s]
config.json: 100%  718/718 [00:00<00:00, 61.7kB/s]
vocab.json: 100%  1.04M/1.04M [00:00<00:00, 8.10MB/s]
merges.txt: 100%  456k/456k [00:00<00:00, 31.6MB/s]
tokenizer.json: 100%  1.36M/1.36M [00:00<00:00, 47.1MB/s]
model.safetensors: 100%  1.52G/1.52G [00:22<00:00, 171MB/s]
generation_config.json: 100%  124/124 [00:00<00:00, 3.73kB/s]
```

```
Loading model: pythia-70m
```

```
tokenizer_config.json: 100%  396/396 [00:00<00:00, 9.05kB/s]
tokenizer.json:  2.11M/? [00:00<00:00, 23.5MB/s]
special_tokens_map.json: 100%  99.0/99.0 [00:00<00:00, 1.44kB/s]
config.json: 100%  567/567 [00:00<00:00, 9.66kB/s]
model.safetensors: 100%  166M/166M [00:02<00:00, 123MB/s]
```

```
# sentence, final_word, condition, item_number
stimuli = [
    # Item 1
    ("The boy ate the", "sandwich", "Expected", 1),
    ("The boy ate the", "rock", "Anomalous", 1),

    # Item 2
    ("The mechanic repaired the", "engine", "Expected", 2),
    ("The mechanic repaired the", "cloud", "Anomalous", 2),

    # Item 3
    ("The child drank the", "juice", "Expected", 3),
    ("The child drank the", "sand", "Anomalous", 3),

    # Item 4
    ("The professor read the", "book", "Expected", 4),
    ("The professor read the", "apple", "Anomalous", 4),

    # Item 5
    ("The driver parked the", "car", "Expected", 5),
    ("The driver parked the", "moon", "Anomalous", 5),

    # Item 6
    ("The farmer milked the", "cow", "Expected", 6),
    ("The farmer milked the", "chair", "Anomalous", 6),

    # Item 7
    ("The man kicked the", "ball", "Expected", 7),
    ("The man kicked the", "cloud", "Anomalous", 7),

    # Item 8
    ("The nurse washed the", "patient", "Expected", 8),
    ("The nurse washed the", "idea", "Anomalous", 8),

    # Item 9
    ("The chef chopped the", "onion", "Expected", 9),
    ("The chef chopped the", "shadow", "Anomalous", 9),

    # Item 10
    ("The tourist photographed the", "statue", "Expected", 10),
    ("The tourist photographed the", "sound", "Anomalous", 10),

    # Item 11
    ("The woman folded the", "clothes", "Expected", 11),
```

```
    ("The woman folded the", "water", "Anomalous", 11),

    # Item 12
    ("The scientist measured the", "temperature", "Expected", 12),
    ("The scientist measured the", "happiness", "Anomalous", 12),
]
```

```
### Create DataFrame
df_stimuli = pd.DataFrame(stimuli, columns=['Sentence', 'Word', 'Condition', 'Item'])
df_stimuli.head(3)
```

	Sentence	Word	Condition	Item
0	The boy ate the	sandwich	Expected	1
1	The boy ate the	rock	Anomalous	1
2	The mechanic repaired the	engine	Expected	2

```
import numpy as np

def surprisal(prob):
    """Return surprisal = -log(prob)."""
    if prob <= 0:
        return np.inf
    return -np.log(prob)
```

```
def next_seq_prob(model, tokenizer, prefix, target_word):
    """
    Returns P(target_word | prefix) for GPT-style causal LMs.
    """

    # tokenize prefix
    inputs = tokenizer(prefix, return_tensors="pt")

    # forward pass
    with torch.no_grad():
        outputs = model(**inputs)

    # logits for next token (final position)
    next_token_logits = outputs.logits[0, -1]

    # convert to probabilities
    probs = torch.softmax(next_token_logits, dim=0)

    # tokenize target word
    target_id = tokenizer.encode(target_word, add_special_tokens=False)[0]

    return float(probs[target_id])
```

```
all_results = []

for model_name in models.keys():
    print(f"\nRunning model: {model_name}")
    model = models[model_name]
    tokenizer = tokenizers[model_name]

    for index, row in tqdm(df_stimuli.iterrows(), total=df_stimuli.shape[0]):
        prob = next_seq_prob(model, tokenizer, row['Sentence'], row['Word'])

        all_results.append({
            'Model': model_name,
            'Sentence': row['Sentence'],
            'Word': row['Word'],
            'Condition': row['Condition'],
            'Item': row['Item'],
```

```
            'Probability': prob,
            'Surprisal': surprisal(prob),
        })
```

```
Running model: gpt2-small
100%|██████████| 24/24 [00:04<00:00,  5.50it/s]

Running model: gpt2-medium
100%|██████████| 24/24 [00:18<00:00,  1.28it/s]

Running model: pythia-70m
100%|██████████| 24/24 [00:01<00:00, 21.44it/s]
```

```
### Create DataFrame
df_all = pd.DataFrame(all_results)
df_all.head(10)
```

	Model	Sentence	Word	Condition	Item	Probability	Surprisal
0	gpt2-small	The boy ate the	sandwich	Expected	1	7.359986e-08	16.424623
1	gpt2-small	The boy ate the	rock	Anomalous	1	4.877549e-08	16.836038
2	gpt2-small	The mechanic repaired the	engine	Expected	2	2.701130e-07	15.124425
3	gpt2-small	The mechanic repaired the	cloud	Anomalous	2	4.968908e-10	21.422651
4	gpt2-small	The child drank the	juice	Expected	3	4.292898e-07	14.661134
5	gpt2-small	The child drank the	sand	Anomalous	3	5.192500e-09	19.076051
6	gpt2-small	The professor read the	book	Expected	4	2.576080e-06	12.869242
7	gpt2-small	The professor read the	apple	Anomalous	4	1.296419e-10	22.766245
8	gpt2-small	The driver parked the	car	Expected	5	6.414365e-06	11.956971
9	gpt2-small	The driver parked the	moon	Anomalous	5	9.186772e-09	18.505501

```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
sns.barplot(
    data=df_all,
    x="Condition",
    y="Surprisal",
    hue="Model",
    ci=68
)
plt.title("Surprisal by Condition across Models")
plt.ylabel("Surprisal (-log P)")
plt.show()
```

```
/tmp/ipython-input-3180124756.py:5: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=('ci', 68)` for the same effect.

  sns.barplot(
```