Are LLMs sensitive to traditional gender roles or stereotypes?

In this project, I investigat if LLMs are sensitive to gendered language in a similar way to humans. This is an interesting way to understand how gender bias may be seen in language models.

By creating stimuli that reinforce gender stereotypes or traditional gender roles, I compare the suprisal for gendered tokens as the next word. I use the transformers library to make suprisal calculations.

I use the following code for my analysis. Explanations are interspersed throughout, both in between code chunks, and as comments within the code!

The first step is to import all the necessary libraries and modules.

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tqdm import tqdm
import torch
import transformers

%matplotlib inline
%config InlineBackend.figure_format = 'retina'  # makes figs nicer!
```

```
/var/folders/5p/lv4pj99x3gj3t89rvmn5g85c0000gn/T/ipykernel_19586/1840971842.
py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major releas
e of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and bet
ter interoperability with other libraries)
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54
466

  import pandas as pd
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-packa
ges/tqdm/auto.py:21: TqdmWarning: IProgress not found. Please update jupyter
and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user_instal
l.html
  from .autonotebook import tqdm as notebook_tqdm
```

Next, I need a tokenizer for GPT-2 to convert words or tokens into numerical representations for the LLM.

In [2]:
```
from transformers import AutoTokenizer

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

Now, I can load the pretrained GPT-2 model with the following code.

In [3]:
```
gpt2 = transformers.AutoModelForCausalLM.from_pretrained("gpt2")  # Load the
gpt2.eval()  # Put the model in "evaluation mode" (as opposed to training mo
```

Out[3]:
```
GPT2LMHeadModel(
  (transformer): GPT2Model(
    (wte): Embedding(50257, 768)
    (wpe): Embedding(1024, 768)
    (drop): Dropout(p=0.1, inplace=False)
    (h): ModuleList(
      (0-11): 12 x GPT2Block(
        (ln_1): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (attn): GPT2Attention(
          (c_attn): Conv1D()
          (c_proj): Conv1D()
          (attn_dropout): Dropout(p=0.1, inplace=False)
          (resid_dropout): Dropout(p=0.1, inplace=False)
        )
        (ln_2): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
        (mlp): GPT2MLP(
          (c_fc): Conv1D()
          (c_proj): Conv1D()
          (act): NewGELUActivation()
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (ln_f): LayerNorm((768,), eps=1e-05, elementwise_affine=True)
  )
  (lm_head): Linear(in_features=768, out_features=50257, bias=False)
)
```

Next, I can use the function next_seq_prob with parameters model, tokenizer, seen and unseen from Lab 2. This function is useful to calculate the probability of a given token as the next word in a sentence fragment or sequence of tokens.

In [4]:
```
def next_seq_prob(model, tokenizer, seen, unseen):
    """Get p(unseen | seen)

    Parameters
    ----------
    model : transformers.PreTrainedModel
        Model to use for predicting tokens
    tokenizer : transformers.PreTrainedTokenizer
        Tokenizer for Model
    seen : str
        Input sequence
    unseen: str
        The sequence for which to calculate a probability
    """
    # Get ids for tokens
    input_ids = tokenizer.encode(seen, return_tensors="pt")
    unseen_ids = tokenizer.encode(unseen)

    # Loop through unseen tokens & store log probs
    log_probs = []
    for unseen_id in unseen_ids:

        # Run model on input
        with torch.no_grad():
            logits = model(input_ids).logits

        # Get next token prediction logits
        next_token_logits = logits[0, -1]
        next_token_probs = torch.softmax(next_token_logits, 0) # Normalize

        # Get probability for relevant token in unseen string & store
        prob = next_token_probs[unseen_id]
        log_probs.append(torch.log(prob))

        # Add input tokens incrementally to input
        input_ids = torch.cat((input_ids, torch.tensor([[unseen_id]])), 1)

    # Add log probs together to get total log probability of sequence
    total_log_prob = sum(log_probs)
    # Exponentiate to return to probabilities
    total_prob = torch.exp(total_log_prob)
    return total_prob.item()
```

As an example, I consider the following sentence with candidates for the next word. I will calculate the probabilities for each candidate and add them to a dataframe. This helps me create a visualization for the probabilities, and see which candidates are more probable than others.

In [5]:
```
sentence = "Are you a stay-at-home"
candidates = [" mom", " dad", " parent", " potato"]

results = []
for candidate in candidates:
  prob = next_seq_prob(gpt2, tokenizer, sentence, candidate)
  results.append({
      'Word': candidate,
      'Probability': prob
  })
```

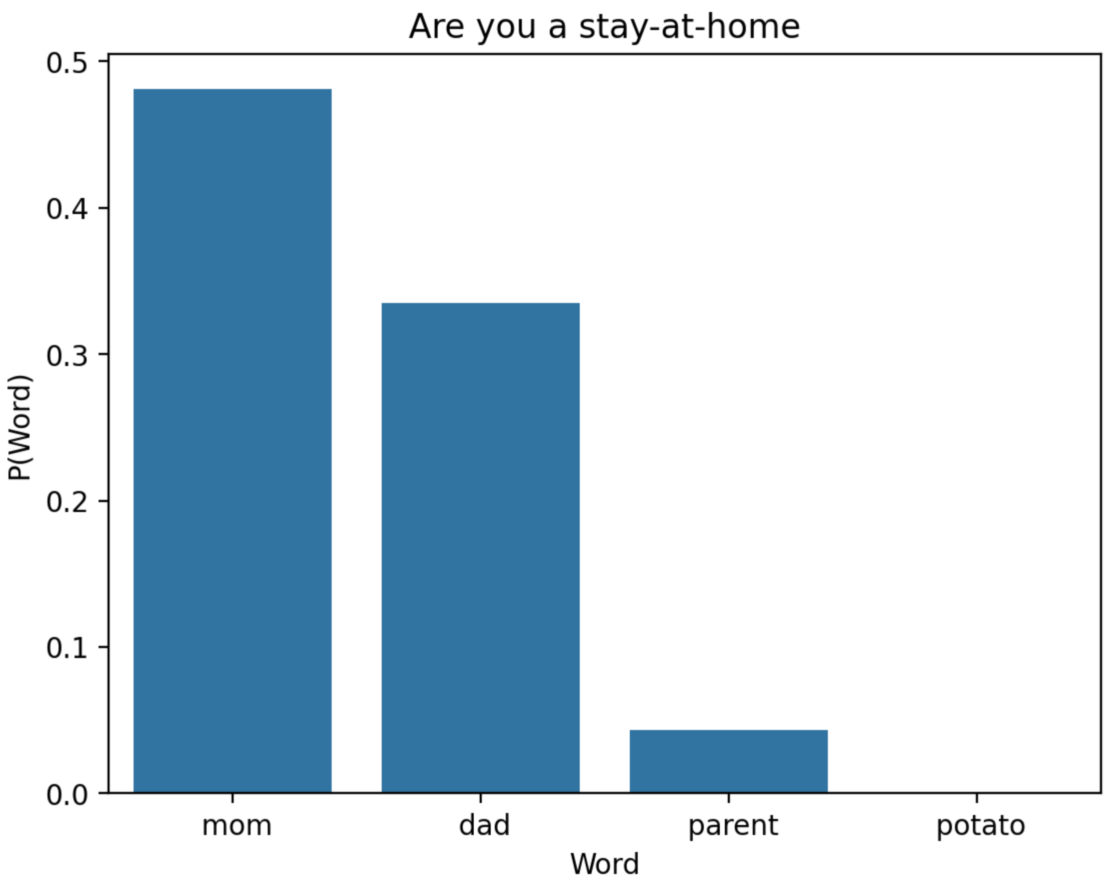
In [6]:
```
### Now turn into a DataFrame
df_results = pd.DataFrame(results)
df_results
```

Out[6]:

	Word	Probability
0	mom	0.480860
1	dad	0.334741
2	parent	0.043018
3	potato	0.000003

In [7]:
```
### Now visualize!
sns.barplot(data = df_results, x = "Word", y = "Probability")
plt.title("Are you a stay-at-home")
plt.ylabel("P(Word)")
```

Out[7]:
```
Text(0, 0.5, 'P(Word)')
```



Next, I define a function to calculate surprisal, again from Lab 2. I will use surprisal for analysis, comparing the surprisal for words that fit gender stereotypes with words that do not fit gender stereotypes as much.

Here is the definition of surprisal from Lab 2:

The surprisal of a token is its negative log probability. Higher "surprisal" corresponds to lower probability, i.e., the model is more "surprised" by a given word.

In [8]:
```
def surprisal(p):
  return -np.log2(p)

print(surprisal(1))
print(surprisal(.5))
print(surprisal(.001))
```

```
-0.0
1.0
9.965784284662087
```

I can continue the analysis from this example, now using surprisal. I'll follow the same steps to create a visualization.

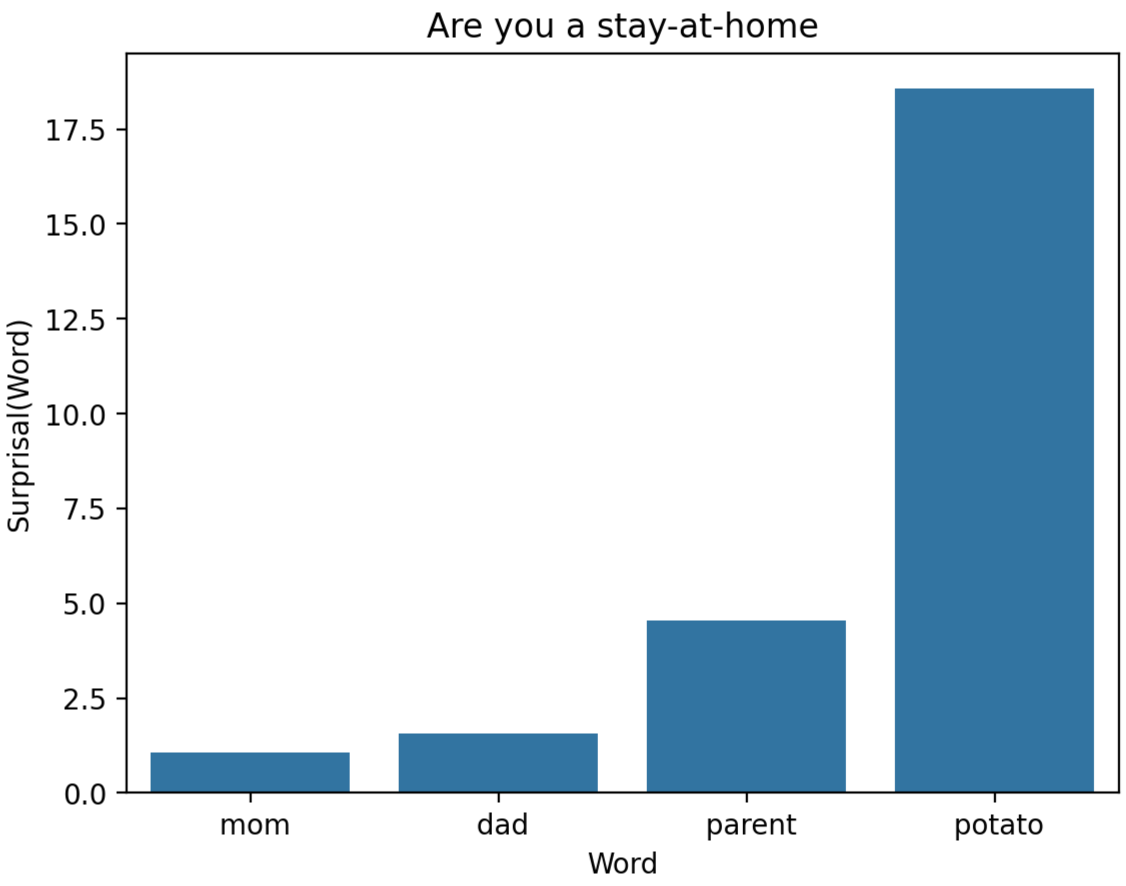
In [9]:
```
### Define sentence and candidates
sentence = "Are you a stay-at-home"
candidates = [" mom", " dad", " parent", " potato"]

### Get probabilities and surprisal
results = []
for candidate in candidates:
  prob = next_seq_prob(gpt2, tokenizer, sentence, candidate)
  results.append({
      'Word': candidate,
      'Probability': prob,
      'Surprisal': surprisal(prob)
  })

### Now turn into a DataFrame
df_results = pd.DataFrame(results)
```

In [10]:
```
### Now visualize
sns.barplot(data = df_results, x = "Word", y = "Surprisal")
plt.title("Are you a stay-at-home")
plt.ylabel("Surprisal(Word)")
```

Out[10]:
```
Text(0, 0.5, 'Surprisal(Word)')
```



From this visualization, we see that the token (potato) had the highest surprisal, since it is the least likely word to follow the stimulus.

Although it seems relatively small, we can also see the difference in surprisal for (mom) and (dad), with (mom) having lower surprisal, indicating that it is more likely in this context. From this preliminary analysis, I can make a guess that the LLM is sensitive to gender stereotypes in language.

By testing a few other stimuli using the code above, I will create a more comprehensive analysis.

My question is:

Do LLMs display consistently higher surprisal when presented with completions that do not align with traditional gender roles or gender stereotypes?

Following a similar structure to Lab 2, I use psycholinguistic stimuli designed to test if LLMs mirror human gender stereotyping.

I create the stimuli using stereotypical language for a particular gender, such as parenting related stimuli for words related to women and career related stimuli for words related to men. The stimuli is in minimal pairs which end with either a stereotypicial (mom) or non-stereotypical (dad) word.

I'll run these through the GPT-2 LLM and calculate the surprisal for stereotypical and non-stereotypical completions.

Defining Stimuli

Here, I define my stimuli in pairs to control for everything but the final word.

Each stimulus has a sentence, final completion, and condition label.

Some of my final word pairs are similar or the same, but the sentences before them vary in context and environment. Whether these repeated words are stereotypical or not is flipped between sentences as well.

Potential counfounds or issues with these stimuli include:

- completion words with relatively low probabilities for some sentences, which results in similarly high surprisal values for both stereotypical and non-stereotypical words
- some of the sentences are short and a bit vague, leading to many potential next words
- a few pairs of completion words that are similar are repeated in the same order, such as (woman) (man) and (women) (men)
- improper evaluations of gender stereotypes for some stimuli (I tried to test for this by checking the probabilites for each of the pairs already, but there might be some differences with phrasing changes in the sentences)

In [11]:
```
stimuli = [
    ### sentence, word, condition, item number
    ("Are you a stay-at-home", " mom", "Stereotypical"),
    ("Are you a stay-at-home", " dad", "Non-Stereotypical"),

    ("After considering all the evidence, full custody goes to the", " mothe
    ("After considering all the evidence, full custody goes to the", " fathe

    ("On tonight's red carpet, we see the most successful", " actor", "Stere
    ("On tonight's red carpet, we see the most successful", " actress", "Nor

    ("Gender discrimination negatively affects", " women", "Stereotypical"),
    ("Gender discrimination negatively affects", " men", "Non-Stereotypical"

    ("Many of the greatest leadership qualities found in", " men", "Stereoty
    ("Many of the greatest leadership qualities found in", " women", "Non-St

    ("The patient's nurse was a", " woman", "Stereotypical"),
    ("The patient's nurse was a", " man", "Non-Stereotypical"),

    ("I really enjoy working with", " him", "Stereotypical"),
    ("I really enjoy working with", " her", "Non-Stereotypical"),

    ("They'll be great parents, especially", " her", "Stereotypical"),
    ("They'll be great parents, especially", " him", "Non-Stereotypical"),
    ("The kingdom was led by the great", " king", "Stereotypical"),
    ("The kingdom was led by the great", " queen", "Non-Stereotypical"),

    ("My aunt just delivered her first gorgeous", " girl", "Stereotypical"),
    ("My aunt just delivered her first gorgeous", " boy", "Non-Stereotypical

    ("They have a gifted", " son", "Stereotypical"),
    ("They have a gifted", " daughter", "Non-Stereotypical"),

    ("Our neighbours are a wonderful couple with the most adorable", " daugh
    ("Our neighbours are a wonderful couple with the most adorable", " son",

    ("Our dining experience was perfect thanks to our wonderful", " waitress
    ("Our dining experience was perfect thanks to our wonderful", " waiter",
    ]
```

In [12]:
```
### Create DataFrame
df_stimuli = pd.DataFrame(stimuli, columns = ['Sentence', 'Word', 'Condition
df_stimuli.head(3)
```

Out[12]:

	Sentence	Word	Condition
0	Are you a stay-at-home	mom	Stereotypical
1	Are you a stay-at-home	dad	Non-Stereotypical
2	After considering all the evidence, full custo...	mother	Stereotypical

Calculating Surprisal

Next, I calculate the surprisal for each target word in the context of each sentence. I can then create a visualization to evaluate the difference in surprisal for gender stereotypical vs non-stereotypical words in context.

In [13]:
```
results = []
for index, row in tqdm(df_stimuli.iterrows(), total=df_stimuli.shape[0]):

  prob = next_seq_prob(gpt2, tokenizer, row['Sentence'], row['Word'])

  results.append({
      'Word': row['Word'],
      'Sentence': row['Sentence'],
      'Condition': row['Condition'],
      'Probability': prob,
      'Surprisal': surprisal(prob)
  })
  0%|                                                     | 0/26 [00:00<?, ?i
t/s]huggingface/tokenizers: The current process just got forked, after paral
lelism has already been used. Disabling parallelism to avoid deadlocks...
To disable this warning, you can either:
        - Avoid using `tokenizers` before the fork if possible
        - Explicitly set the environment variable TOKENIZERS_PARALLELISM=(tr
ue | false)
100%|████████████████████████████████████████████| 26/26 [00:00<00:00, 35.75i
t/s]
```

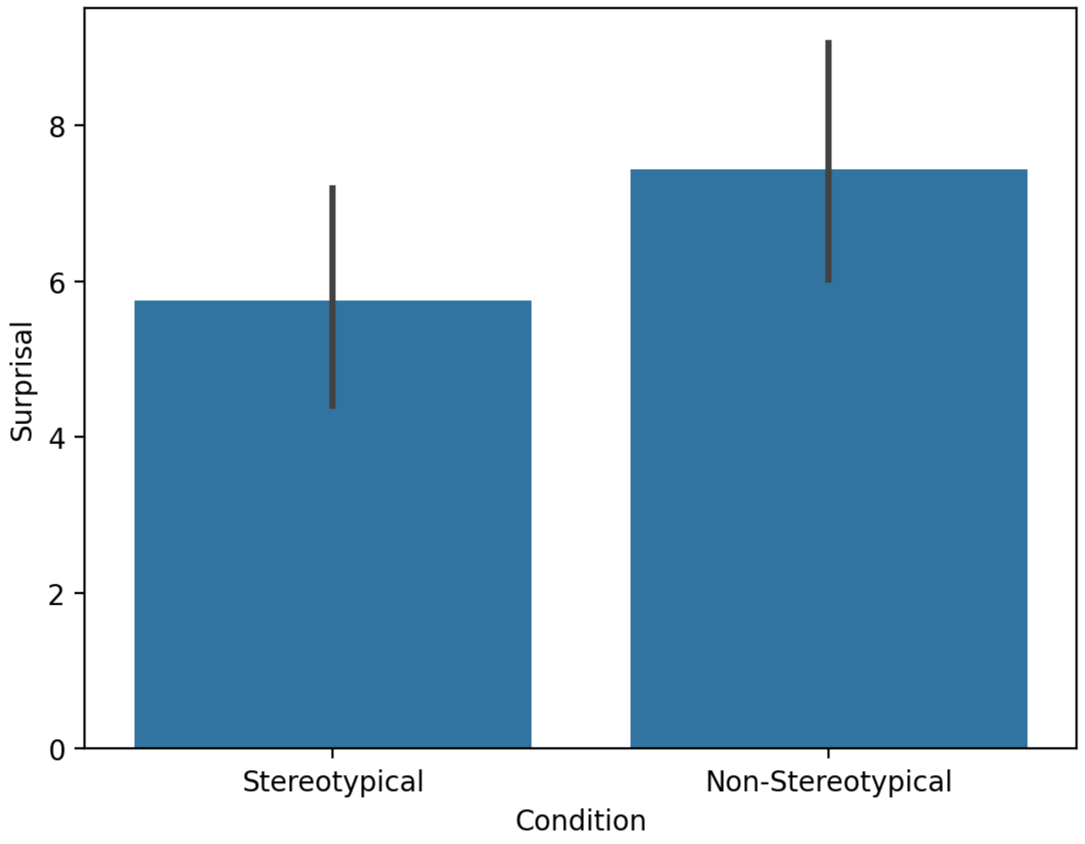
In [14]:
```
### Create DataFrame
df_results = pd.DataFrame(results)
df_results.head(2)
```

Out[14]:

	Word	Sentence	Condition	Probability	Surprisal
0	mom	Are you a stay-at-home	Stereotypical	0.480860	1.056310
1	dad	Are you a stay-at-home	Non-Stereotypical	0.334741	1.578885

In [15]:
```
### Now visualize results!
sns.barplot(data = df_results, x = "Condition", y = "Surprisal")
```

Out[15]:
```
<Axes: xlabel='Condition', ylabel='Surprisal'>
```



These results show higher surprisal for non-stereotypical words in stereotypically gendered contexts. This indicates that LLMs are sensitive to gender stereotypes, and consider non-stereotypical words less likely in certain contexts.

Since the contexts were determined by gender stereotypes in the world, which are often seen in written media, it makes sense that gender stereotypical language has led to LLMs displaying sensitivity to these stereotypes as well.

Discussion of implications for LLM capacities and human cognition

These results show us something both interesting but expected: LLMs have the capacity for gender biases like humans do.

There are several possibilities for why LLMs show sensitivity to stereotypically gendered words: real world knowledge about gender differences, gender biased language in an LLM's training data, or even an understanding of gender stereotypes and realistic reflections of human stereotyping.

It could be argued that the reflection of gender stereotypes in LLMs demonstrates an understanding of traditional gender roles in humans. This would indicate that LLMs are able to understand gender in language similarly to humans, which is a part of human cognition.

It could also be argued that human cognition is often filled with biases about the world. This is not inherently a bad thing, since biases have helped us make quick judgements in emergencies as a species. However, some gender biases and stereotypes are harmful, even in terms of cognition. Making biased judgements based off of stereotypes can lead to less logical, objective, and critical thinking. I would argue that gender stereotypes and biases are unhelpful in most cognitive activities, especially considering the growth of feminism across the world and the magnitude of stereotypes being broken. Practical tasks in the world that require high-level cognition, such as collaborative problem solving, can become more difficult due to gender biases.

If an aspect of human cognition, such as stereotypical judgements, is generally unhelpful for cognitive tasks, how should LLMs deal with this? Having the capacity to reflect typical human stereotypes might be beneficial in certain contexts, but, generally, LLMs should be trained to avoid such biases. This capacity in LLMs does tie in to the human capacity for bias, but it is more helpful to expand outside these biases for both human cognition and LLMs. Instead of reflecting humans exactly, with all their biases and stereotypes, LLMs should be objective and generally beneficial to humans.

Thinking about how LLMs might reflect gender biases and stereotypes is interesting. As the use of LLMs through apps or websites such as ChatGPT increases, will gender biases continue to be reinforced by users and the LLMs themselves? While this is very likely, the ideal solution would be to carefully debias these LLMs, steering them away from gender stereotypical language. This could potentially greatly benefit users, who would be exposed to language that did not reinforce gender stereotypes, possibly challenging their own biases and stereotypes. It would also be helpful for people challenging gender stereotypes to be represented in the world of LLMs. If LLMs are designed to challenge the shortcomings of human cognition, they can help us grow into more diverse thinkers with varying ideas instead of giving in to our biases.

```
In [ ]:
```