# Part 2: LLM

Aniket Kamat Yuyang Wu Brycen Manners
Soohyun Kim

Wednesday 17th December, 2025

Curvenote

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

EO = pd.read_csv("executive_orders.csv")
```

## 1 Intro

Here, we will fine-tune a small pretrained language model on historical executive order titles to examine whether domain-specific stylistic patterns could be learned. We will compare generated outputs before and after fine-tuning. This experiment is exploratory and qualitative in nature.

This notebook explores whether a small pretrained language model can adapt to the stylistic structure of U.S. executive order titles after lightweight fine-tuning.

```
EO.shape
```

```
(1000, 13)
```

## 2 Check 'title' column (NA values, type)

```
#check if they are all string type
EO["title"].dtype == object
```

```
True
```

```
#no missing values
EO["title"].isna().sum()
```

```
np.int64(0)
```

We verified that the title column contains no missing values and is stored as a string-type variable. This ensures that all executive order titles are valid textual inputs for downstream language-model fine-tuning.

# 3 Check suitability for model

```
titles_df = EO["title"].reset_index(drop=True).to_frame(name="title")
len(titles_df)
```

```
1000
```

```
titles_df["char_len"] = titles_df["title"].str.len()
titles_df["char_len"].describe()
```

```
count    1000.000000
mean       78.167000
std        48.402147
min        16.000000
25%        49.000000
50%        66.000000
75%        96.000000
max       905.000000
Name: char_len, dtype: float64
```

```
sum(titles_df["char_len"]<200)/1000
```

```
0.986
```

```
titles_df_trunc = titles_df[titles_df["char_len"]<200]['title']
titles_df_trunc = titles_df_trunc.to_frame(name='title')
```

Executive order titles are generally short, with a median length of 66 characters, making them well suited for lightweight language-model fine-tuning. Thus, we are going to work with titles with length smaller than 200 for the sake of simplicity of the model, and it covers more than 95% of the total data.

# 4 Using LLM

## 4.1 Before Training

```
import pandas as pd
import torch
from transformers import AutoTokenizer, AutoModelForCausalLM, set_seed
```

```
/home/jovyan/.local/share/envs/finalproj/lib/python3.11/site -packages/tqdm/auto.py:21: Tqdm
  from .autonotebook import tqdm as notebook_tqdm
```

```
model_name = "distilgpt2"
tokenizer = AutoTokenizer.from_pretrained(model_name)
model_base = AutoModelForCausalLM.from_pretrained(model_name)
model_base.eval()

set_seed(259)  # reproducibility
def generate_text(model, prompt, max_new_tokens=20):
    inputs = tokenizer(prompt, return_tensors="pt")
    with torch.no_grad():
        out = model.generate(
            **inputs,
            max_new_tokens=max_new_tokens,
            do_sample=True,
            temperature=0.8,
            top_p=0.95,
            pad_token_id=tokenizer.eos_token_id,
            eos_token_id=tokenizer.eos_token_id
        )
    text = tokenizer.decode(out[0], skip_special_tokens=True)
    return text.replace("\n", " ").strip()

prompts = [
    "Executive Order on ",
    "Executive Order on Protecting ",
    "Establishing the ",
    "Amending Executive Order ",
    "Executive Order on National Security and "
]

before = []
for p in prompts:
    before.append(generate_text(model_base, p))
before
```

```
['Executive Order on ix -8 -9.',
 'Executive Order on Protecting - - - - - - - - - - - - - - - - - - - - - - - - -',
 'Establishing the vernacular is a process that allows us to incorporate the same set of cor
 'Amending Executive Order _____',
 'Executive Order on National Security and Â Trade in Information."']
```

Before fine-tuning, the model often follows the prompt structure but produces placeholders, encoding artifacts, and generic prose, indicating that it has not learned the formal conventions of executive order titles.

## 4.2 After training

```
from datasets import Dataset
```

```
from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from_pretrained("distilgpt2")
tokenizer.pad_token = tokenizer.eos_token

ds = Dataset.from_dict({
    "text": titles_df_trunc["title"].tolist()
})

def tokenize(batch):
    return tokenizer(
        batch["text"],
        truncation=True,
        padding="max_length",
        max_length=64
    )

tokenized = ds.map(tokenize, batched=True, remove_columns=["text"])
tokenized = tokenized.map(lambda x: {"labels": x["input_ids"]})
```

```
Map:   0%|                                                                                | 0/9

Map: 100%|                                     | 986/986 [00:00<00:00, 20106.88 examples/s]
```

```
Map:   0%|                                                                                | 0/9

Map: 100%|                                     | 986/986 [00:00<00:00, 11980.98 examples/s]
```

```
from transformers import Trainer, TrainingArguments, DataCollatorForLanguageModeling

data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

training_args = TrainingArguments(
    output_dir="llm_ckpt",
    num_train_epochs=1,
    per_device_train_batch_size=8,
    logging_steps=50,
    save_steps=200,
    report_to="none"
)

model_ft = AutoModelForCausalLM.from_pretrained(model_name)
```

```
trainer = Trainer(
    model=model_ft,
    args=training_args,
    train_dataset=tokenized,
    data_collator=data_collator
)

trainer.train()
```

```
/home/jovyan/.local/share/envs/finalproj/lib/python3.11/site -packages/torch/utils/data/data
  warnings.warn(warn_msg)
```

```
`loss_type=None` was set in the config but it is unrecognized. Using the default loss: `For
```

[124/124 02:51, Epoch 1/1]

| Step | Training Loss |
|---|---|
| 50 | 3.918400 |
| 100 | 3.544500 |

```
TrainOutput(global_step=124, training_loss=3.6490478515625, metrics={'train_runtime': 173.22
```

```
model_ft.eval()
set_seed(259)

after = []
for p in prompts:
    after.append(generate_text(model_ft, p))
after
```

```
['Executive Order on ixenable Employment, Economic Performance and Support for the American
 'Executive Order on Protecting ills and Jobs From Terrorist, Terrorist, and Terrorist Extre
 'Establishing the étente Agreement on Civil Rights and Equal Opportunity and Equality in th
 'Amending Executive Order _____ of 2018 to Prohibit Executive Order No. 13981, Effective on
 'Executive Order on National Security and ills of the White House Council on the Foreign Re
```

```
results = pd.DataFrame({
    "prompt": prompts,
    "before": before,
    "after": after
})
results.to_csv("outputs/llm_title_outputs.csv", index=False)

for i, row in results.iterrows():
    print(f"=============  {i+1}th prompt: \"{row['prompt']}\"  ==============\n")
    print(f"before -tuning: {row['before']}\n")
    print(f"after -tuning: {row['after']}\n")
```

```
=============  1th prompt: "Executive Order on "  ==============

before -tuning: Executive Order on ix -8 -9.

after -tuning: Executive Order on ixenable Employment, Economic Performance and Support for

=============  2th prompt: "Executive Order on Protecting "  ==============

before -tuning: Executive Order on Protecting - - - - - - - - - - - - - - - - - - - - - - - -

after -tuning: Executive Order on Protecting ills and Jobs From Terrorist, Terrorist, and Te

=============  3th prompt: "Establishing the "  ==============

before -tuning: Establishing the vernacular is a process that allows us to incorporate the s

after -tuning: Establishing the étente Agreement on Civil Rights and Equal Opportunity and E

=============  4th prompt: "Amending Executive Order "  ==============

before -tuning: Amending Executive Order _____

after -tuning: Amending Executive Order _____ of 2018 to Prohibit Executive Order No. 13981,

=============  5th prompt: "Executive Order on National Security and "  ==============

before -tuning: Executive Order on National Security and Â Trade in Information."

after -tuning: Executive Order on National Security and ills of the White House Council on t
```

After one epoch of fine-tuning on historical executive order titles, the model's generations become noticeably more aligned with the formal structure of real EO titles. In particular, the fine-tuned outputs more frequently include administrative phrasing (e.g., "Establishing…", "Amending Executive Order…", references to EO numbers, and effective dates) that was largely absent or inconsistent in baseline generations. While some artifacts remain (occasional repetition and nonsensical fragments), the overall tone and format shift toward the bureaucratic, title-like style of the training corpus. This suggests that even lightweight fine-tuning can adapt a small pretrained language model to domain-specific writing conventions.