Historical Policy Text Classification – Annotated Version

This notebook explains how to extract insights from historical economic policy texts using two NLP techniques:

- Named Entity Recognition (NER) with spaCy
- Zero-Shot Text Classification with HuggingFace Transformers

---

Exercise 1: Named Entity Recognition with `spaCy`

We use spaCy to identify named entities (people, dates, organizations) in a historical policy statement.

```
import spacy

nlp = spacy.load("en_core_web_sm")

doc = nlp("In 1955, President Ruiz Cortines approved higher tariffs on
textile imports.")

for ent in doc.ents:
    print(ent.text, ent.label_)
```

Explanation

- `spacy.load(...)` loads a pre-trained English language model.
- `doc.ents` extracts entities recognized by the model.
- Useful to identify policy actors and time references.

Exercise 2: Zero-Shot Text Classification

We classify sentences without labeled training data using a transformer model (`facebook/bart-large-mnli`).

```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")

texts = [
    "In 1983, the Ministry of Finance lobbied for textile tariff hikes.",
    "In 1991, Taiwan reduced duties on electronics following WTO
recommendations."
]
```

```
candidate_labels = ["political", "economic", "neutral"]

for text in texts:
    result = classifier(text, candidate_labels)
    print(f"{text} → {result['labels'][0]} ({result['scores'][0]:.2f})")
```

Explanation

- We initialize a `pipeline` for zero-shot classification.
- Each sentence is compared with the candidate labels.
- The model returns the most likely label with a confidence score.

Exercise 3: Zero-Shot Classification with HuggingFace Transformers

In this exercise, we explore **zero-shot classification**, a technique that allows models to assign labels to text **without any specific training on those labels**. This is especially powerful for rapid prototyping in NLP when you don't have a labeled dataset.

We use the `facebook/bart-large-mnli` model. It is based on the **Natural Language Inference (NLI)** paradigm, where it scores how likely a candidate label entails the meaning of the input sentence.

🔍 How it works

For each sentence, the model:

1. Converts your candidate label into a hypothesis (e.g., *"This text is about politics."*).
2. Evaluates whether this hypothesis is entailed by the original text (e.g., *"The Ministry of Finance raised tariffs..."*).
3. Assigns a probability score to each candidate label.

This means we can dynamically classify any new sentence without fine-tuning the model on a custom dataset.

🔧 Code

```
from transformers import pipeline
from pprint import pprint

classifier = pipeline("zero-shot-classification")

texts = [
    "In 1983, the Ministry of Finance lobbied for textile tariff hikes.",
    "In 1991, Taiwan reduced duties on electronics following WTO
recommendations."
]

candidate_labels = ["political", "economic", "neutral"]

for text in texts:
```

```
    result = classifier(text, candidate_labels)
    print(f"Text: {text}")
    pprint(result)
```

✅ Output Example

```
Text: In 1983, the Ministry of Finance lobbied for textile tariff hikes.
{'labels': ['economic', 'political', 'neutral'],
 'scores': [0.912, 0.061, 0.027],
 'sequence': 'In 1983, the Ministry of Finance lobbied for textile tariff
hikes.'}

Text: In 1991, Taiwan reduced duties on electronics following WTO
recommendations.
{'labels': ['economic', 'neutral', 'political'],
 'scores': [0.884, 0.084, 0.032],
 'sequence': 'In 1991, Taiwan reduced duties on electronics following WTO
recommendations.'}
```

As you can see, the model confidently classifies both sentences as **economic**, since the actions described relate to tariffs and duties — classic economic policy content.

🧠 Key Takeaways

- You don't need labeled training data to perform classification.
- Label design is flexible — you can change or expand candidate labels at inference time.
- Ideal for quick prototyping in policy, legal, and historical research domains.