```
# Install required libraries (run once in Google Colab)
!pip install -q spacy transformers torch pandas datasets
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-any
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 73.2 MB/s eta 0:00:00
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```
# spaCy for traditional NLP and NER
import spacy
from spacy import displacy

# Hugging Face libraries for transformer-based NER
from transformers import pipeline, AutoTokenizer, AutoModelForTokenClassification

# PyTorch (backend for transformer models)
import torch

# Pandas for tabular data representation
import pandas as pd
```

```
# Load English pretrained spaCy model
nlp = spacy.load("en_core_web_sm")

"""
A pipeline in spaCy is a sequence of components applied to text.
spaCy pipeline includes:
- Tokenizer: splits text into tokens
- Tagger: assigns POS tags
- Parser: analyzes sentence structure
- NER: detects named entities like PERSON, ORG, GPE
"""
```

```
'\nA pipeline in spaCy is a sequence of components applied to text.\nspaCy pipeline includes:\n- Tokenizer: splits text into to
kens\n- Tagger: assigns POS tags\n- Parser: analyzes sentence structure\n- NER: detects named entities like PERSON, ORG, GPE
```

```
# List of real-world sentences
sentences = [
    "Apple announced a new iPhone in California.",
    "Virat Kohli scored a century for India in the World Cup.",
    "The Prime Minister of India met Elon Musk in New York.",
    "Google is investing heavily in artificial intelligence.",
    "The Olympics will be held in Paris in 2024."
]
```

```
# List to store spaCy NER results
spacy_results = []

for sentence in sentences:
    doc = nlp(sentence)
    for ent in doc.ents:
        spacy_results.append([sentence, ent.text, ent.label_])

# Display results
for row in spacy_results:
    print(f"Sentence: {row[0]}")
    print(f"Entity: {row[1]} | Label: {row[2]}")
    print("-" * 50)
```

```
Sentence: Apple announced a new iPhone in California.
Entity: Apple | Label: ORG
--------------------------------------------------
Sentence: Apple announced a new iPhone in California.
Entity: iPhone | Label: ORG
--------------------------------------------------
Sentence: Apple announced a new iPhone in California.
Entity: California | Label: GPE
```

```
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: Virat Kohli | Label: PERSON
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: a century | Label: DATE
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: India | Label: GPE
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: the World Cup | Label: EVENT
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: India | Label: GPE
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: Elon Musk | Label: PERSON
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: New York | Label: GPE
--------------------------------------------------
Sentence: Google is investing heavily in artificial intelligence.
Entity: Google | Label: ORG
--------------------------------------------------
Sentence: The Olympics will be held in Paris in 2024.
Entity: Olympics | Label: EVENT
--------------------------------------------------
Sentence: The Olympics will be held in Paris in 2024.
Entity: Paris | Label: GPE
--------------------------------------------------
Sentence: The Olympics will be held in Paris in 2024.
Entity: 2024 | Label: DATE
--------------------------------------------------
```

```
# Create DataFrame
spacy_df = pd.DataFrame(
    spacy_results,
    columns=["Sentence", "Entity", "Label"]
)

spacy_df
```

	Sentence	Entity	Label
0	Apple announced a new iPhone in California.	Apple	ORG
1	Apple announced a new iPhone in California.	iPhone	ORG
2	Apple announced a new iPhone in California.	California	GPE
3	Virat Kohli scored a century for India in the ...	Virat Kohli	PERSON
4	Virat Kohli scored a century for India in the ...	a century	DATE
5	Virat Kohli scored a century for India in the ...	India	GPE
6	Virat Kohli scored a century for India in the ...	the World Cup	EVENT
7	The Prime Minister of India met Elon Musk in N...	India	GPE
8	The Prime Minister of India met Elon Musk in N...	Elon Musk	PERSON
9	The Prime Minister of India met Elon Musk in N...	New York	GPE
10	Google is investing heavily in artificial inte...	Google	ORG
11	The Olympics will be held in Paris in 2024.	Olympics	EVENT
12	The Olympics will be held in Paris in 2024.	Paris	GPE
13	The Olympics will be held in Paris in 2024.	2024	DATE

```
# Visualize NER for the first sentence
doc = nlp(sentences[0])
displacy.render(doc, style="ent", jupyter=True)

"""
Different colors represent different entity types.
Labels like ORG, GPE, PERSON indicate entity categories.
"""
```

```
Apple ORG announced a new iPhone ORG in California GPE .
'\nDifferent colors represent different entity types.\nLabels like ORG, GPE, PERSON indicate entity categories.\n'
```

```
# Load tokenizer and model (BERT fine-tuned on CoNLL-2003)
tokenizer = AutoTokenizer.from_pretrained("dbmdz/bert-large-cased-finetuned-conll03-english")
model = AutoModelForTokenClassification.from_pretrained("dbmdz/bert-large-cased-finetuned-conll03-english")

# Create NER pipeline
ner_pipeline = pipeline("ner", model=model, tokenizer=tokenizer, aggregation_strategy="simple")

"""
Token classification assigns a label to each token.
Transformer models understand context using attention mechanisms,
allowing better handling of ambiguous words.
"""
```

```
/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/settings/tokens), set it
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(
Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to enable higher rate limits and faster d
WARNING:huggingface_hub.utils._http:Warning: You are sending unauthenticated requests to the HF Hub. Please set a HF_TOKEN to en
config.json: 100%  998/998 [00:00<00:00, 27.6kB/s]
tokenizer_config.json: 100%  60.0/60.0 [00:00<00:00, 1.43kB/s]
vocab.txt: 213k/? [00:00<00:00, 4.37MB/s]
model.safetensors: 100%  1.33G/1.33G [00:17<00:00, 243MB/s]
Loading weights: 100%  391/391 [00:00<00:00, 533.99it/s, Materializing param=classifier.weight]
BertForTokenClassification LOAD REPORT from: dbmdz/bert-large-cased-finetuned-conll03-english
Key                       | Status     |  |
--------------------------+------------+--+-
bert.pooler.dense.weight | UNEXPECTED |  |
bert.pooler.dense.bias   | UNEXPECTED |  |

Notes:
- UNEXPECTED    :can be ignored when loading from different task/architecture; not ok if you expect identical arch.
'\nToken classification assigns a label to each token.\nTransformer models understand context using attention mechanisms,\nallo
wing better handling of ambiguous words.\n'
```

```
# List to store transformer NER results
hf_results = []

for sentence in sentences:
    entities = ner_pipeline(sentence)
    for ent in entities:
        hf_results.append([
            sentence,
            ent["word"],
            ent["entity_group"],
            round(ent["score"], 4)
        ])

# Print results
for row in hf_results:
    print(f"Sentence: {row[0]}")
    print(f"Entity: {row[1]} | Label: {row[2]} | Confidence: {row[3]}")
    print("-" * 50)
```

```
Sentence: Apple announced a new iPhone in California.
Entity: Apple | Label: ORG | Confidence: 0.9976999759674072
--------------------------------------------------
Sentence: Apple announced a new iPhone in California.
Entity: iPhone | Label: MISC | Confidence: 0.9952999949455261
--------------------------------------------------
Sentence: Apple announced a new iPhone in California.
Entity: California | Label: LOC | Confidence: 0.9997000098228455
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: Virat Kohli | Label: PER | Confidence: 0.9973999857902527
--------------------------------------------------
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: India | Label: LOC | Confidence: 0.9995999932289124
--------------------------------------------------
```

```
Sentence: Virat Kohli scored a century for India in the World Cup.
Entity: World Cup | Label: MISC | Confidence: 0.9952999949455261
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: India | Label: LOC | Confidence: 0.9994000196456909
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: Elon Musk | Label: PER | Confidence: 0.9961000084877014
--------------------------------------------------
Sentence: The Prime Minister of India met Elon Musk in New York.
Entity: New York | Label: LOC | Confidence: 0.9994999766349792
--------------------------------------------------
Sentence: Google is investing heavily in artificial intelligence.
Entity: Google | Label: ORG | Confidence: 0.9991000294685364
--------------------------------------------------
Sentence: The Olympics will be held in Paris in 2024.
Entity: Olympics | Label: MISC | Confidence: 0.9918000102043152
--------------------------------------------------
Sentence: The Olympics will be held in Paris in 2024.
Entity: Paris | Label: LOC | Confidence: 0.9993000030517578
--------------------------------------------------
```

```
# Create DataFrame for transformer output
hf_df = pd.DataFrame(
    hf_results,
    columns=["Sentence", "Entity", "Label", "Confidence Score"]
)

hf_df
```

	Sentence	Entity	Label	Confidence Score
0	Apple announced a new iPhone in California.	Apple	ORG	0.9977
1	Apple announced a new iPhone in California.	iPhone	MISC	0.9953
2	Apple announced a new iPhone in California.	California	LOC	0.9997
3	Virat Kohli scored a century for India in the ...	Virat Kohli	PER	0.9974
4	Virat Kohli scored a century for India in the ...	India	LOC	0.9996
5	Virat Kohli scored a century for India in the ...	World Cup	MISC	0.9953
6	The Prime Minister of India met Elon Musk in N...	India	LOC	0.9994
7	The Prime Minister of India met Elon Musk in N...	Elon Musk	PER	0.9961
8	The Prime Minister of India met Elon Musk in N...	New York	LOC	0.9995
9	Google is investing heavily in artificial inte...	Google	ORG	0.9991
10	The Olympics will be held in Paris in 2024.	Olympics	MISC	0.9918
11	The Olympics will be held in Paris in 2024.	Paris	LOC	0.9993

```
comparison_data = {
    "Feature": [
        "Model Type",
        "Speed",
        "Accuracy (Qualitative)",
        "Context Handling",
        "Confidence Score",
        "GPU Requirement"
    ],
    "spaCy": [
        "Statistical + Rules",
        "Fast",
        "Moderate",
        "Limited",
        "No",
        "No"
    ],
    "Hugging Face": [
        "Transformer-based",
        "Slower",
        "High",
        "Excellent",
        "Yes",
        "Yes"
    ]
```

```
}

comparison_df = pd.DataFrame(comparison_data)
comparison_df
```

	Feature	spaCy	Hugging Face
0	Model Type	Statistical + Rules	Transformer-based
1	Speed	Fast	Slower
2	Accuracy (Qualitative)	Moderate	High
3	Context Handling	Limited	Excellent
4	Confidence Score	No	Yes
5	GPU Requirement	No	Yes