Lab 6: Thực hành chuyên sâu về Phân tích cú pháp phụ thuộc (Dependency Parsing)

Phần 1: Giới thiệu và Cài đặt

Trước tiên, chúng ta cần cài đặt thư viện `spaCy` và mô hình ngôn ngữ tiếng Anh cần thiết.

```
!pip install -U spacy
```

```
Requirement already satisfied: spacy in
/usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in
/usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in
/usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in
/usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in
/usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (0.20.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (4.67.1)
Requirement already satisfied: numpy>=1.19.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in
/usr/local/lib/python3.12/dist-packages (from spacy) (2.12.3)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in
```

```
/usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!
=1.8.1,<3.0.0,>=1.7.4->spacy) (0.7.0)
Requirement already satisfied: pydantic-core==2.41.4 in
/usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!
=1.8.1,<3.0.0,>=1.7.4->spacy) (2.41.4)
Requirement already satisfied: typing-extensions>=4.14.1 in
/usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!
=1.8.1,<3.0.0,>=1.7.4->spacy) (4.15.0)
Requirement already satisfied: typing-inspection>=0.4.2 in
/usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!
=1.8.1,<3.0.0,>=1.7.4->spacy) (0.4.2)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0-
>spacy) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0-
>spacy) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0-
>spacy) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0-
>spacy) (2025.11.12)
Requirement already satisfied: blis<1.4.0,>=1.3.0 in
/usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4-
>spacy) (1.3.3)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in
/usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4-
>spacy) (0.1.5)
Requirement already satisfied: click>=8.0.0 in
/usr/local/lib/python3.12/dist-packages (from typer-
slim<1.0.0,>=0.3.0->spacy) (8.3.1)
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in
/usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2-
>spacy) (0.23.0)
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in
/usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2-
>spacy) (7.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
Requirement already satisfied: wrapt in
/usr/local/lib/python3.12/dist-packages (from smart-
open<8.0.0,>=5.2.1->weasel<0.5.0,>=0.4.2->spacy) (2.0.1)

!python -m spacy download en_core_web_md
```

```
Collecting en-core-web-md==3.8.0
  Downloading
https://github.com/explosion/spacy-models/releases/download/en_core_we
b_md-3.8.0/en_core_web_md-3.8.0-py3-none-any.whl (33.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 33.5/33.5 MB 36.5 MB/s eta
0:00:00
d
Successfully installed en-core-web-md-3.8.0
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_md')
⚠ 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.
```

Phần 2: Phân tích câu và Trực quan hóa

Sử dụng `displaCy` để trực quan hóa cây phụ thuộc, giúp chúng ta dễ dàng hiểu cấu trúc câu.

2.1. Tải mô hình và phân tích câu

```
import spacy
from spacy import displacy

nlp = spacy.load("en_core_web_md")

text = "The quick brown fox jumps over the lazy dog."

text

{"type":"string"}

doc = nlp(text)

doc

The quick brown fox jumps over the lazy dog.

print("Token | Phân tích Cú pháp | Head | Quan hệ")
print("-" * 50)
for token in doc:
    print(f"{token.text:<5} | {token.pos_:<15} | {token.head.text:<4}
| {token.dep_}")

Token | Phân tích Cú pháp | Head | Quan hệ
--------------------------------------------------
The   | DET             | fox  | det
```

```
quick | ADJ              | fox   | amod
brown | ADJ              | fox   | amod
fox   | NOUN             | jumps | nsubj
jumps | VERB             | jumps | ROOT
over  | ADP              | jumps | prep
the   | DET              | dog   | det
lazy  | ADJ              | dog   | amod
dog   | NOUN             | over  | pobj
.     | PUNCT            | jumps | punct
```

2.2. Trực quan hóa cây phụ thuộc

Trong môi trường Colab, chúng ta dùng `displacy.render` thay vì `displacy.serve` để hiển thị trực tiếp kết quả.

```
# Trực quan hóa cây phụ thuộc trực tiếp trong notebook
# style="dep" là để hiển thị dependency parsing
displacy.render(doc, style="dep", jupyter=True, options={"compact":
True, "distance": 100})
```

```
<IPython.core.display.HTML object>
```

Trả lời câu hỏi:

1, Từ nào là gốc (`ROOT`) của câu?

- Từ `jumps` là gốc (`ROOT`) của câu. (Đây là động từ chính của câu.)

2, `jumps` có những từ phụ thuộc (`dependent`) nào? Các quan hệ đó là gì?

- `fox` (quan hệ: `nsubj` - chủ ngữ)
- `over` (quan hệ: `prep` - giới từ)
- `.` (quan hệ: `punct` - dấu câu)

3, `fox` là head của những từ nào?

- `fox` là head của các từ: **The** (quan hệ: `det`), **quick** (quan hệ: `amod`), **brown** (quan hệ: `amod`).

Phần 3: Truy cập các thành phần trong cây phụ thuộc

Chúng ta sẽ sử dụng các thuộc tính của Token để duyệt qua cây phụ thuộc.

```python
# Lấy một câu khác để phân tích
text = "Apple is looking at buying U.K. startup for $1 billion"
doc = nlp(text)

# In ra thông tin của từng token
print(f"{'TEXT':<12} | {'DEP':<10} | {'HEAD TEXT':<12} | {'HEAD
POS':<8} | {'CHILDREN'}")
print("-" * 70)

for token in doc:
    # Trích xuất các thuộc tính
    children = [child.text for child in token.children]
    print(f"{token.text:<12} | {token.dep_:<10} |
{token.head.text:<12} | {token.head.pos_:<8} | {children}")
```

```
TEXT         | DEP        | HEAD TEXT    | HEAD POS | CHILDREN
----------------------------------------------------------------------
Apple        | nsubj      | looking      | VERB     | []
is           | aux        | looking      | VERB     | []
looking      | ROOT       | looking      | VERB     | ['Apple', 'is',
'at']
at           | prep       | looking      | VERB     | ['buying']
buying       | pcomp      | at           | ADP      | ['startup']
U.K.         | compound   | startup      | NOUN     | []
startup      | dobj       | buying       | VERB     | ['U.K.', 'for']
for          | prep       | startup      | NOUN     | ['billion']
$            | quantmod   | billion      | NUM      | []
1            | compound   | billion      | NUM      | []
billion      | pobj       | for          | ADP      | ['$', '1']
```

Phần 4: Duyệt cây phụ thuộc để trích xuất thông tin

4.1. Bài toán: Tìm chủ ngữ và tân ngữ của một động từ

```python
text = "The cat chased the mouse and the dog watched them."
doc = nlp(text)

print("--- Trích xuất (Chủ ngữ, Động từ, Tân ngữ) ---")
for token in doc:
    # Chỉ tìm các động từ
    if token.pos_ == "VERB":
        verb = token.text
        subject = ""
        obj = ""

        # Tìm chủ ngữ (nsubj) và tân ngữ (dobj) trong các con của
```

```
động từ
        for child in token.children:
            if child.dep_ == "nsubj":
                subject = child.text
            if child.dep_ == "dobj":
                obj = child.text

        if subject and obj:
            print(f"Found Triplet: ({subject}, {verb}, {obj})")
```

```
--- Trích xuất (Chủ ngữ, Động từ, Tân ngữ) ---
Found Triplet: (cat, chased, mouse)
Found Triplet: (dog, watched, them)
```

4.2. Bài toán: Tìm các tính từ bổ nghĩa cho một danh từ

```
text = "The big, fluffy white cat is sleeping on the warm mat."
doc = nlp(text)

print("\n--- Trích xuất Danh từ và Tính từ bổ nghĩa ---")
for token in doc:
    # Chỉ tìm các danh từ
    if token.pos_ == "NOUN":
        adjectives = []
        # Tìm các tính từ bổ nghĩa (amod) trong các con của danh từ
        for child in token.children:
            if child.dep_ == "amod":
                adjectives.append(child.text)

        if adjectives:
            print(f"Danh từ '{token.text}' được bổ nghĩa bởi các tính
từ: {adjectives}")


--- Trích xuất Danh từ và Tính từ bổ nghĩa ---
Danh từ 'cat' được bổ nghĩa bởi các tính từ: ['big', 'fluffy',
'white']
Danh từ 'mat' được bổ nghĩa bởi các tính từ: ['warm']
```

Phần 5: Bài tập tự luyện

Bài 1: Tìm động từ chính của câu

Động từ chính của câu thường có quan hệ ROOT.

```
def find_main_verb(doc):
    """Tìm và trả về Token là động từ chính (ROOT) của câu."""
```

```
    for token in doc:
        # Quan hệ phụ thuộc (dep_) là 'ROOT'
        if token.dep_ == "ROOT":
            return token
    return None

text1 = "The international conference will take place in Paris."
doc1 = nlp(text1)

doc1
```

```
The international conference will take place in Paris.
```

```
main_verb1 = find_main_verb(doc1)

print(f"Câu 1: '{text1}' -> Động từ chính: {main_verb1.text if
main_verb1 else 'Không tìm thấy'}")
```

```
Câu 1: 'The international conference will take place in Paris.' ->
Động từ chính: take
```

```
text2 = "She quickly wrote a detailed report for the manager."
doc2 = nlp(text2)

doc2
```

```
She quickly wrote a detailed report for the manager.
```

```
main_verb2 = find_main_verb(doc2)

print(f"Câu 2: '{text2}' -> Động từ chính: {main_verb2.text if
main_verb2 else 'Không tìm thấy'}")
```

```
Câu 2: 'She quickly wrote a detailed report for the manager.' -> Động
từ chính: wrote
```

Bài 2: Trích xuất các cụm danh từ (Noun Chunks)

Sử dụng thuộc tính `token.children` và các quan hệ phụ thuộc như `det`, `amod`, `compound` để xây dựng cụm danh từ quanh một `NOUN`.

```
def extract_simple_noun_chunks(doc):
    noun_chunks = []

    for token in doc:
        if token.pos_ == "NOUN":
            chunk_words = []

            # 1. Thêm các từ bổ nghĩa trực tiếp (children)
            for child in token.children:
                # Các quan hệ thường gặp: det (mạo từ), amod (tính
```

```
từ), compound (danh từ ghép)
                if child.dep_ in ("det", "amod", "compound",
"nummod"):
                    chunk_words.append(child)

            # 2. Thêm Danh từ chính
            chunk_words.append(token)

            # Sắ´p xê´p lại theo vị trí xuâ´t hiện trong câu (token.i
là index)
            chunk_words.sort(key=lambda t: t.i)

            # Kê´t hợp các từ thành cụm danh từ
            noun_chunk = " ".join([t.text for t in chunk_words])
            noun_chunks.append(noun_chunk)

    return noun_chunks
```

Kiểm tra

```
text = "The detailed quarterly report, written by the experienced
analyst, was accepted."
doc = nlp(text)

doc
```

```
The detailed quarterly report, written by the experienced analyst, was
accepted.
```

Sử dụng tính năng có sẵn của spaCy để so sánh

```
print("--- So sánh với .noun_chunks của spaCy ---")
print(f"spaCy Noun Chunks: {[chunk.text for chunk in
doc.noun_chunks]}")
```

```
--- So sánh với .noun_chunks của spaCy ---
spaCy Noun Chunks: ['The detailed quarterly report', 'the experienced
analyst']
```

Sử dụng hàm tự xây dựng

```
print("\n--- Hàm tự xây dựng (Simple Noun Chunks) ---")
print(f"Custom Noun Chunks: {extract_simple_noun_chunks(doc)}")
```

```

--- Hàm tự xây dựng (Simple Noun Chunks) ---
Custom Noun Chunks: ['The detailed quarterly report', 'the experienced
analyst']
```

Bài 3: Tìm đường đi ngắn nhất trong cây

Duyệt ngược từ một token bất kỳ lên đến gốc (`ROOT`) bằng cách sử dụng thuộc tính `token.head`.

```
def get_path_to_root(token):
    """
    Tìm và trả về danh sách các Token trên đường đi từ Token hiện tại lên đến gốc (ROOT).
    """
    path = [token]
    current = token

    # Lặp lại cho đến khi head của token là chính nó (chỉ xảy ra với ROOT)
    # hoặc token.head.dep_ == 'ROOT'
    while current.dep_ != "ROOT":
        # Duyệt ngược lên head
        current = current.head
        path.append(current)

        # Guard: Tránh vòng lặp vô hạn (chỉ phòng trường hợp bất thường)
        if len(path) > len(token.doc):
            break

    return path
```

Kiểm tra

```
text = "The quick brown fox jumps over the lazy dog."
doc = nlp(text)

doc
```

```
The quick brown fox jumps over the lazy dog.
```

```
# Chọn một token ngẫu nhiên, ví dụ: 'lazy'
target_token = doc[6] # 'lazy'

target_token
```

```
the
```

```
path = get_path_to_root(target_token)

path
```

```
[the, dog, over, jumps]
```

```
print(f"Đường đi từ '{target_token.text}' đến ROOT:")
```

```
Đường đi từ 'the' đến ROOT:
```

In đường đi với quan hệ phụ thuộc để dễ hình dung

```
path_str = " -> ".join([f"[{t.text} ({t.dep_})]" for t in path])
print(path_str)
```

```
[the (det)] -> [dog (pobj)] -> [over (prep)] -> [jumps (ROOT)]
```