Name: Muhammad Ghazanfar

Roll#: 20P-0567

Sec: 7A

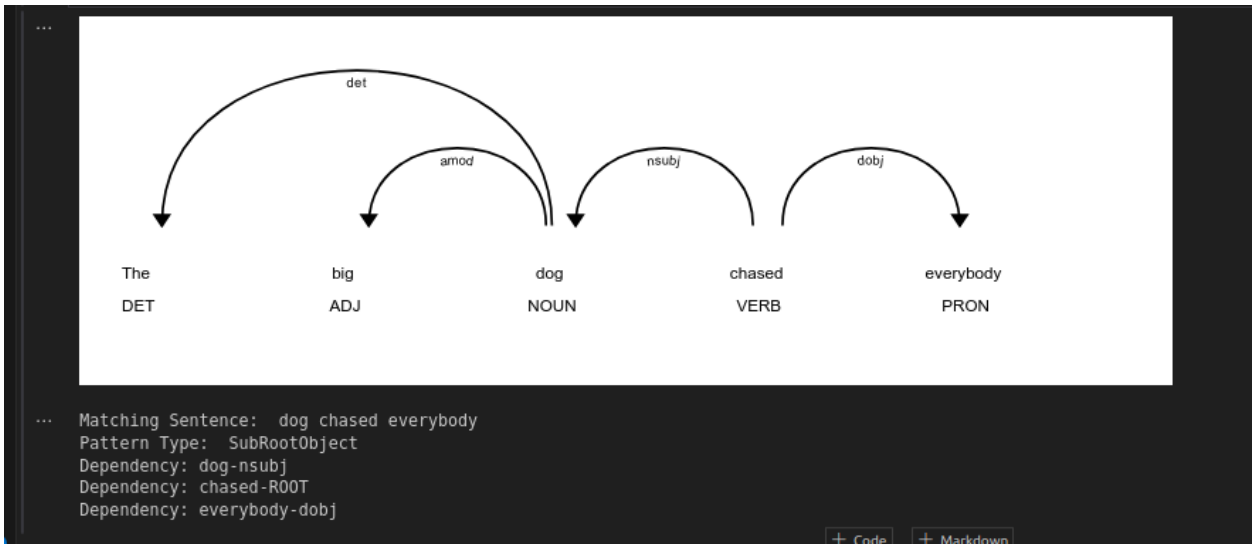
Course: NLP

Lab#04

Task 5: Finding Multiple Patterns automatically in Sentences

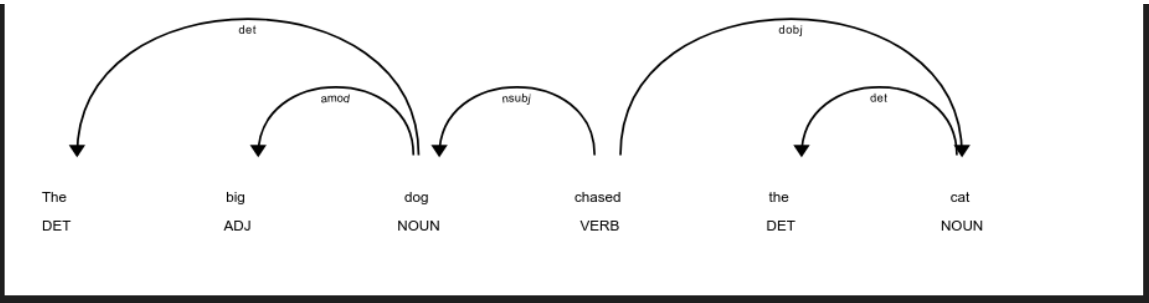
Attempt the following Questions:

1. What text and dependencies did the above code catch for the sentence “The big dog chased everybody”.




```
Matching Sentence:  dog chased everybody
Pattern Type:  SubRootObject
Dependency: dog-nsubj
Dependency: chased-ROOT
Dependency: everybody-dobj
```

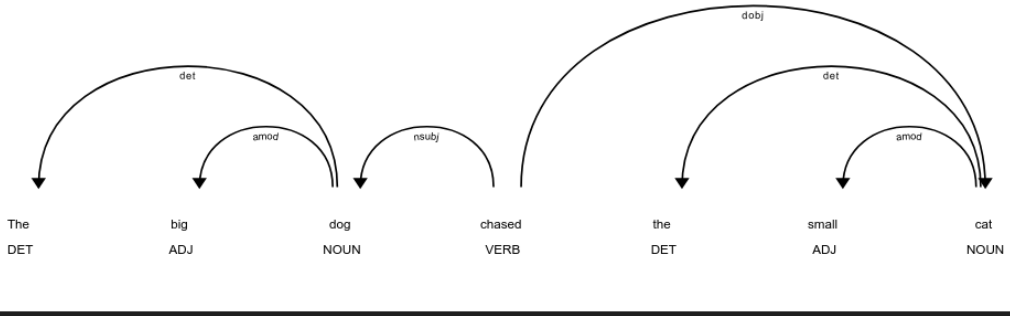
2. Change the sentence to “The big dog chased the cat”. Does the pattern catch the SVO pattern? If not, add another pattern2 to the matcher. The pattern should be DEP: nsubj, DEP: ROOT, DEP: det, DEP:dobj. When done, update matcher.add("SubRootDetObject", [pattern2])




```
Matching Sentence: dog chased the cat
Pattern Type: SubRootDetObject
Dependency: dog-nsubj
Dependency: chased-ROOT
Dependency: the-det
Dependency: cat-dobj
```

3. Now design a third pattern for the sentence “The big dog chased the small cat”.

```
pattern3 = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "det"}, {"DEP": "amod"}, {"DEP": "dobj"}]
matcher.add("SubRootAdjDetObject", [pattern3])

doc = nlp("The big dog chased the small cat")
```




```
Matching Sentence: dog chased the small cat
Pattern Type: SubRootAdjDetObject
Dependency: dog-nsubj
Dependency: chased-ROOT
Dependency: the-det
Dependency: small-amod
Dependency: cat-dobj
```

Attempt the following Questions:

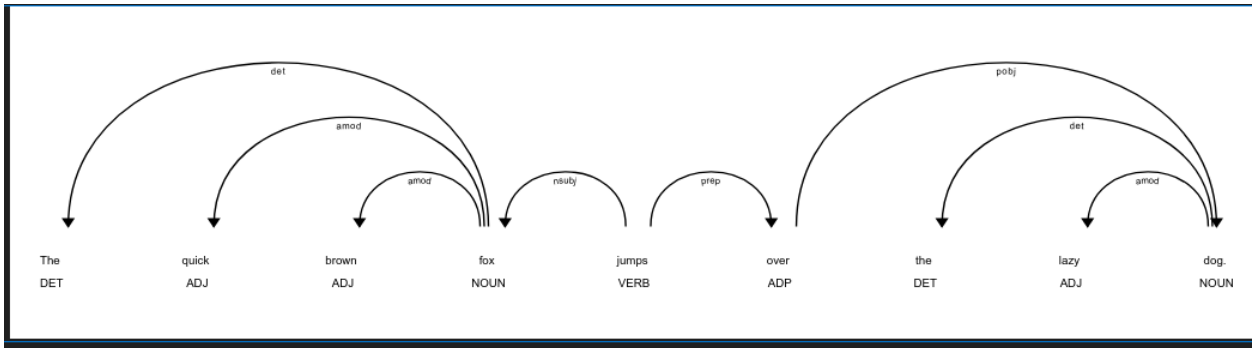
1. **Design a pattern to identify a noun at least one time.**

```
pattern1 = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"OP": "*"}, {"DEP": "dobj"}]
```

0.0s Python

```
import spacy
from spacy.matcher import Matcher
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

matcher = Matcher(nlp.vocab)

# Pattern to identify a noun at least one time
pattern = [{"POS": "NOUN", "OP": "+"}]
matcher.add("OneOrMoreNouns", [pattern])

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

matches = matcher(doc)

displacy.render(doc, style='dep')

for pattern_id, start, end in matches:
    print("Matching Sentence:", doc[start:end].text)
    print("Pattern Type:", doc.vocab.strings[pattern_id])

    for token in doc[start:end]:
        print("Token: {} | POS: {}".format(token.text, token.pos_))
    print("\n" + "="*40 + "\n")
```

1.0s Python

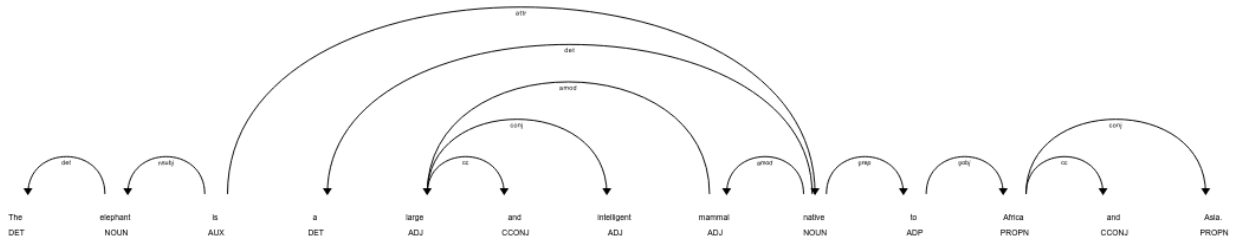



```
Matching Sentence: fox
Pattern Type: OneOrMoreNouns
Token: fox | POS: NOUN

========================================

Matching Sentence: dog
Pattern Type: OneOrMoreNouns
Token: dog | POS: NOUN
```

2. Design a pattern to identify a noun of length >= 10 characters.

```python
import spacy
from spacy.matcher import Matcher
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

matcher = Matcher(nlp.vocab)

# Pattern to identify a noun of length >= 10 characters
pattern = [{"POS": "NOUN", "LENGTH": {">=": 10}}]
matcher.add("LongNoun", [pattern])

doc = nlp("The elephant is a large and intelligent mammal native to Africa and Asia.")

matches = matcher(doc)

displacy.render(doc, style='dep')

for pattern_id, start, end in matches:
    print("Matching Sentence:", doc[start:end].text)
    print("Pattern Type:", doc.vocab.strings[pattern_id])

    for token in doc[start:end]:
        print("Token: {} | POS: {} | Length: {}".format(token.text, token.pos_, len(token.text)))
    print("\n" + "="*40 + "\n")
```

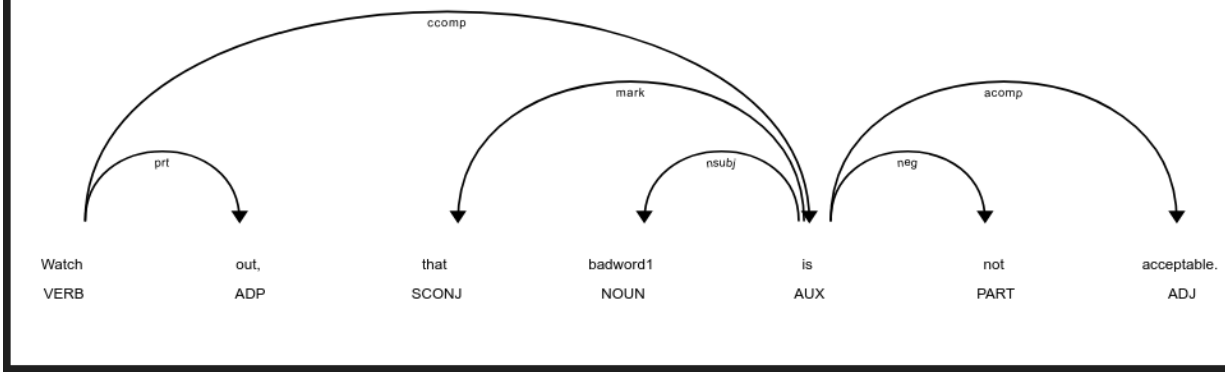
0.8s Python



3. **Design a pattern to identify vulgar language (Hint: you will need usage of IN, or NOT_IN).**

```python
import spacy
from spacy.matcher import Matcher
from spacy import displacy

nlp = spacy.load('en_core_web_sm')

matcher = Matcher(nlp.vocab)

vulgar_words = ["badword1", "badword2", "badword3"]

pattern = [{"LOWER": {"IN": vulgar_words}}]
matcher.add("VulgarLanguage", [pattern])

doc = nlp("Watch out, that badword1 is not acceptable.")

matches = matcher(doc)

displacy.render(doc, style='dep')

for pattern_id, start, end in matches:
    print("Matching Sentence:", doc[start:end].text)
    print("Pattern Type:", doc.vocab.strings[pattern_id])

    for token in doc[start:end]:
        print("Token: {} | Lowercase: {}".format(token.text, token.text.lower()))
    print("\n" + "="*40 + "\n")
```

✓ 1.2s Python




```
Matching Sentence: badword1
Pattern Type: VulgarLanguage
Token: badword1 | Lowercase: badword1
```

Task 6: Getting Replies

1. **Extend the code by adding pattern and matches if a user enters: “I would like to order a pizza”. The bot should ask about which pizza type he/she wants.**

```python
def utterance(msg):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(msg)
    matcher = Matcher(nlp.vocab)

    greeting_pattern = [{"LEMMA": {"IN": ["salam", "assalam", "hi", "hello"]}}]
    matcher.add("greeting", [greeting_pattern])

    order_pizza_pattern = [{"LEMMA": {"IN": ["order"]}}, {"LOWER": "a"}, {"LOWER": "pizza"}]
    matcher.add("order_pizza", [order_pizza_pattern])

    matches = matcher(doc)

    if len(matches) == 0:
        print('Please rephrase your request. Be as specific as possible!')
        return None

    for pattern_id, start, end in matches:
        if doc.vocab.strings[pattern_id] == "greeting":
            return "Welcome to Pizza ordering system"
        elif doc.vocab.strings[pattern_id] == "order_pizza":
            return "Sure! What type of pizza would you like to order?"


user_input = "I would like to order a pizza"
bot_response = utterance(user_input)

print("Bot:", bot_response)
```

✓ 1.4s Python

```
Bot: Sure! What type of pizza would you like to order?
```

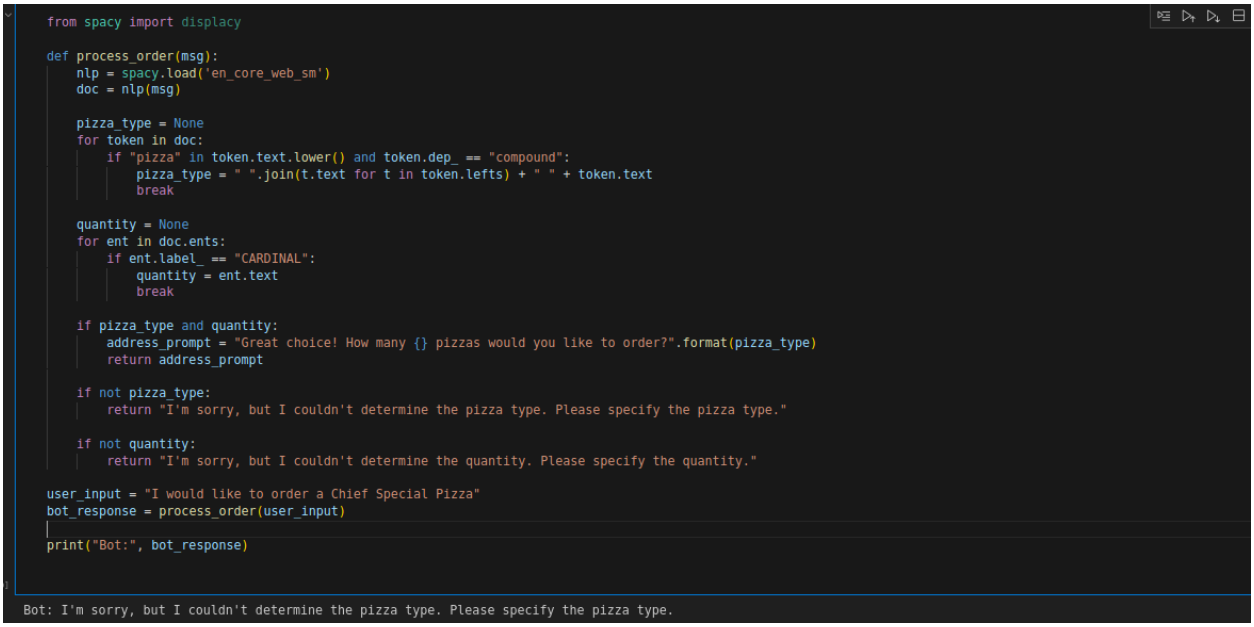
2. **Extend the code by adding pattern and matches if a user enters: “I would like to complain about an order”.**

```python
def utterance(msg):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(msg)
    matcher = Matcher(nlp.vocab)

    greeting_pattern = [{"LEMMA": {"IN": ["salam", "assalam", "hi", "hello"]}}]
    matcher.add("greeting", [greeting_pattern])

    order_pizza_pattern = [{"LEMMA": {"IN": ["order"]}}, {"LOWER": "a"}, {"LOWER": "pizza"}]
    matcher.add("order_pizza", [order_pizza_pattern])

    complaint_pattern = [{"LEMMA": {"IN": ["complain", "complaint"]}}, {"LOWER": "about"}, {"LOWER": "an"}, {"LOWER": "order"}]
    matcher.add("complaint_order", [complaint_pattern])

    matches = matcher(doc)

    if len(matches) == 0:
        print('Please rephrase your request. Be as specific as possible!')
        return None

    for pattern_id, start, end in matches:
        if doc.vocab.strings[pattern_id] == "greeting":
            return "Welcome to Pizza ordering system"
        elif doc.vocab.strings[pattern_id] == "order_pizza":
            return "Sure! What type of pizza would you like to order?"
        elif doc.vocab.strings[pattern_id] == "complaint_order":
            return "I'm sorry to hear that. Please provide more details about your complaint."

user_input = "I would like to complain about an order"
bot_response = utterance(user_input)

print("Bot:", bot_response)
```

✓ 1.0s Python

```
Bot: I'm sorry to hear that. Please provide more details about your complaint.
```

3. **In respone to what pizza type user wants, the user may want to enter “Chief Special Pizza”. Use the .lefts (mentioned in Lab 03) to get the pizza type. Ask about quantity. Use Cardinal as ent type to get the quantity, and place the order. Ask for address, and confirm the user with address.**



```
from spacy import displacy

def process_order(msg):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(msg)

    pizza_type = None
    for token in doc:
        if "pizza" in token.text.lower() and token.dep_ == "compound":
            pizza_type = " ".join(t.text for t in token.lefts) + " " + token.text
            break

    quantity = None
    for ent in doc.ents:
        if ent.label_ == "CARDINAL":
            quantity = ent.text
            break

    if pizza_type and quantity:
        address_prompt = "Great choice! How many {} pizzas would you like to order?".format(pizza_type)
        return address_prompt

    if not pizza_type:
        return "I'm sorry, but I couldn't determine the pizza type. Please specify the pizza type."

    if not quantity:
        return "I'm sorry, but I couldn't determine the quantity. Please specify the quantity."

user_input = "I would like to order a Chief Special Pizza"
bot_response = process_order(user_input)

print("Bot:", bot_response)
```

```
Bot: I'm sorry, but I couldn't determine the pizza type. Please specify the pizza type.
```