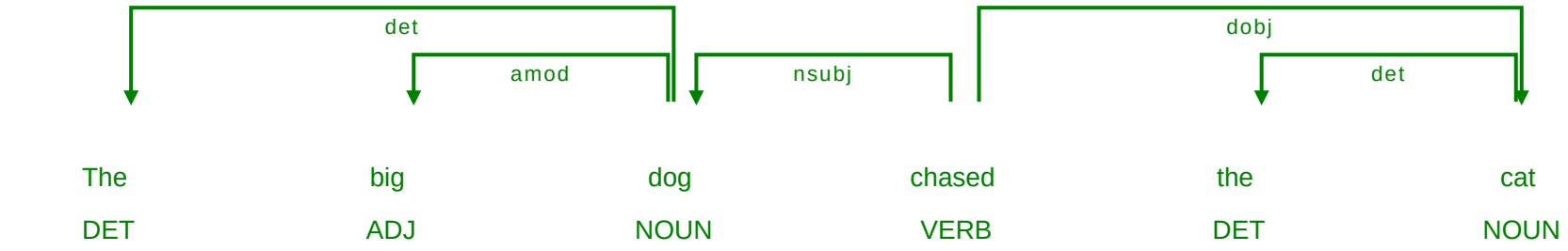
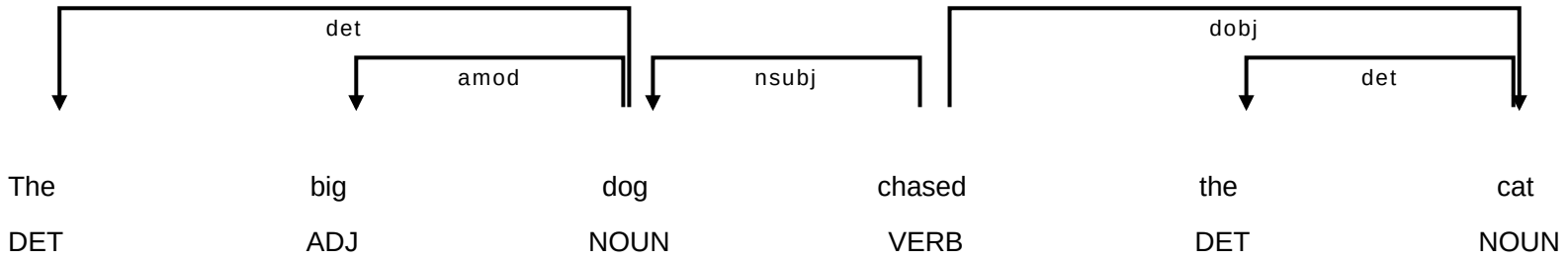
Task 5: Finding Multiple Patterns automatically in Sentences

In [1]:
```
import nltk
import spacy
import nltk
from spacy.matcher import Matcher
from spacy import displacy
nlp = spacy.load('en_core_web_sm')
matcher = Matcher(nlp.vocab)
```

In [2]:
```
pattern1 = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "dobj"}]
matcher.add("SubRootObject", [pattern1])
doc = nlp("The big dog chased the cat")
matches = matcher(doc)
displacy.render(doc, style='dep',options ={ "color" : "green","compact":True})
for pattern_id, start, end in matches:
    print("Matching Sentence: ", doc[start:end])
    print("Pattern Type: ", doc.vocab.strings[pattern_id])
    for token in doc[start:end]:
        print(f"Dependency: {token}---{token.dep_}")
```

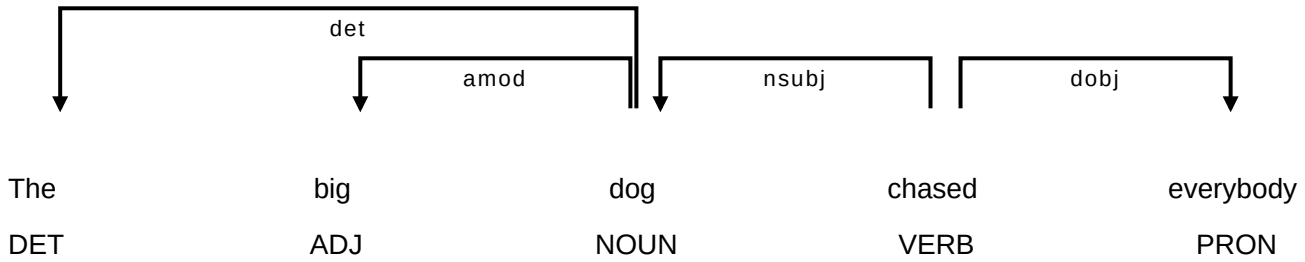


In [5]:
```
# Orignal Patternfor the original SVO pattern
orignal_pattern = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "dobj"}]
matcher.add("SubRootObject", [orignal_pattern])

# Updated Pattren for the updated sentence "The big dog chased the cat"
updated_pattern = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "det"}, {"DEP": "dobj"}]
matcher.add("SubRootDetObject", [updated_pattern])
doc = nlp("The big dog chased the cat")

matches = matcher(doc)

displacy.render(doc, style='dep', options ={ "color" : "black","compact":True})

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

    # Print dependency information for each token in the match
    for token in doc[start:end]:
        print(f"Dependency: {token.text}--{token.dep_}")
```



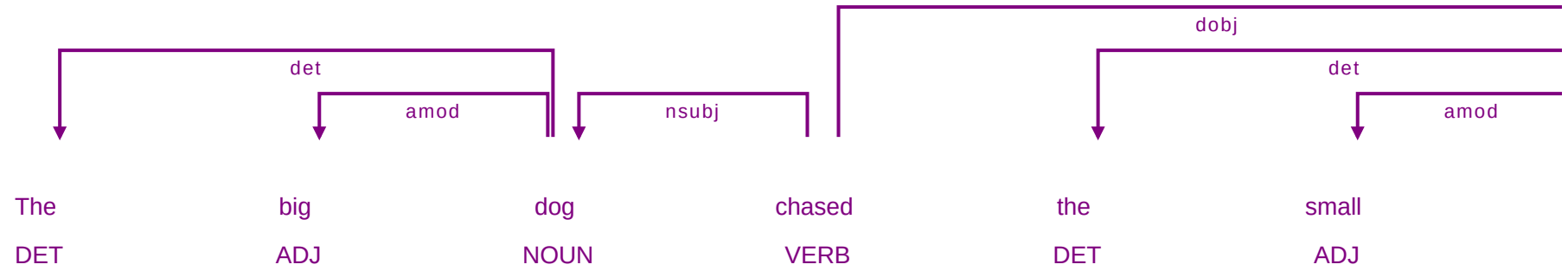
```
Matching Sentence: dog chased the cat
Pattern Type: SubRootDetObject
Dependency: dog--nsubj
Dependency: chased--ROOT
Dependency: the--det
Dependency: cat--dobj
```

In [6]:
```
# Orignal Patternfor the original SVO pattern
orignal_pattern = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "dobj"}]
matcher.add("SubRootObject", [orignal_pattern])

# Updated Pattren for the updated sentence "The big dog chased the cat"
updated_pattern = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "det"}, {"DEP": "dobj"}]
matcher.add("SubRootDetObject", [updated_pattern])

doc = nlp("The big dog chased everybody")

matches = matcher(doc)

displacy.render(doc, style='dep', options ={ "color" : "black","compact":True})

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

    # Print dependency information for each token in the match
    for token in doc[start:end]:
        print(f"Dependency: {token.text}--{token.dep_}")
```



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

In [7]:
```
# Pattern 1 for the original SVO pattern
pattern1 = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "dobj"}]
matcher.add("SubRootObject", [pattern1])

# Pattern 2 for the updated sentence "The big dog chased the cat"
pattern2 = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"DEP": "det"}, {"DEP": "dobj"}]
matcher.add("SubRootDetObject", [pattern2])

# Pattern 3 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")

matches = matcher(doc)

displacy.render(doc, style='dep',options ={ "color" : "purple","compact":True})

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(f"Dependency: {token.text}--{token.dep_}")
```



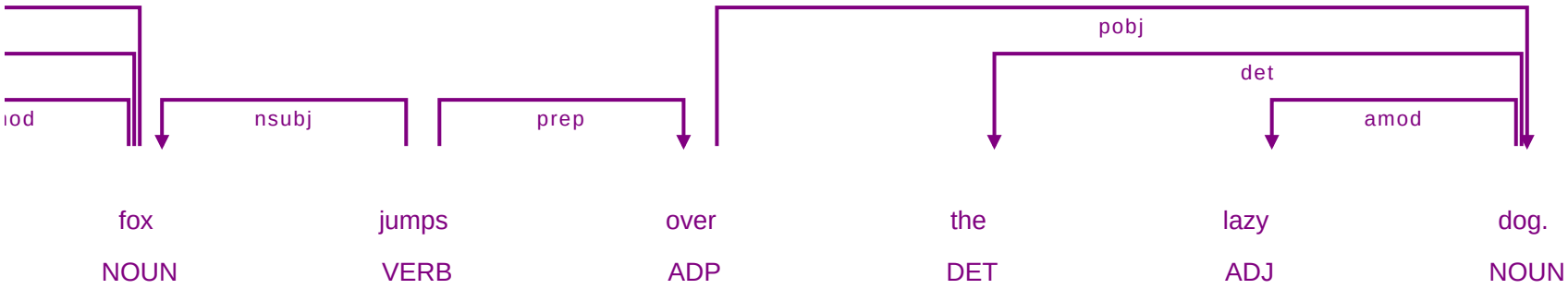

```
Matching Sentence: dog chased the small cat
Pattern Type: SubRootAdjDetObject
Dependency: dog--nsubj
Dependency: chased--ROOT
Dependency: the--det
Dependency: small--amod
Dependency: cat--dobj
```

In [71]:
```
pattern = [{"DEP": "nsubj"}, {"DEP": "ROOT"}, {"OP": "*"}, {"DEP": "dobj"}]
```

In [72]:
```
pattern
```

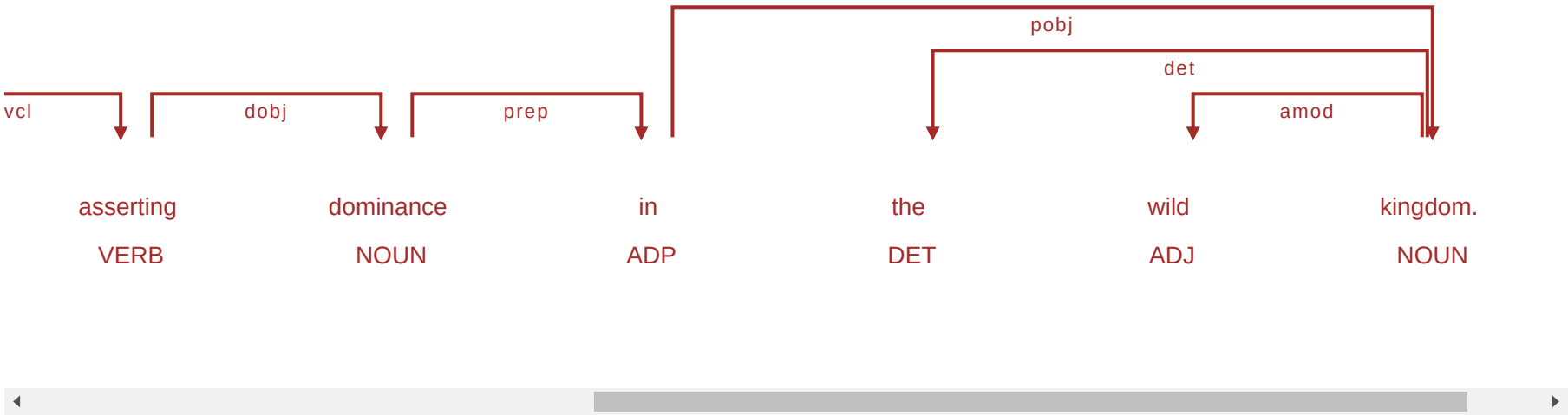
Out[72]:
```
[{'DEP': 'nsubj'}, {'DEP': 'ROOT'}, {'OP': '*'}, {'DEP': 'dobj'}]
```

In [9]:
```
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',options ={ "color" : "purple","compact":True} )

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

    # Print information for each token in the match
    for token in doc[start:end]:
        print(f"Dependency: {token.text}-{token.dep_}")
        print("-"*20)
```



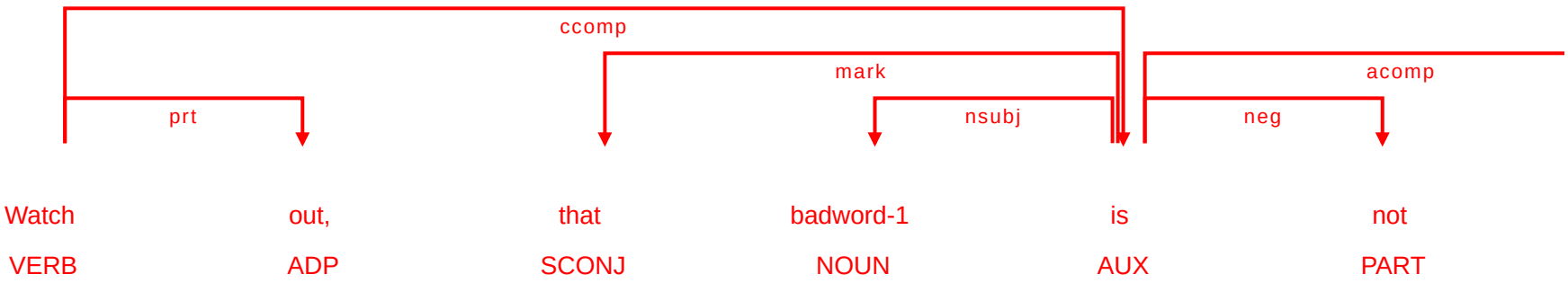

```
Matching Sentence: fox
Pattern Type: OneOrMoreNouns
Dependency: fox-nsubj
--------------------
Matching Sentence: dog
Pattern Type: OneOrMoreNouns
Dependency: dog-pobj
--------------------
```

In [15]:
```
pattern = [{"POS": "NOUN", "LENGTH": {">=": 10}}]
matcher.add("LongNoun", [pattern])

doc = nlp("The majestic lion roars, asserting dominance in the wild kingdom.")

matches = matcher(doc)

displacy.render(doc, style='dep', options ={ "color" : "brown","compact":True})

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(f"Token: {token.text} | POS: {token.pos_} | Length: {len(token.text)}")
    print("\n" + "="*40 + "\n")
```



```
Matching Sentence: lion
Pattern Type: OneOrMoreNouns
Token: lion | POS: NOUN | Length: 4

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

Matching Sentence: dominance
Pattern Type: OneOrMoreNouns
Token: dominance | POS: NOUN | Length: 9

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

Matching Sentence: kingdom
Pattern Type: OneOrMoreNouns
Token: kingdom | POS: NOUN | Length: 7

========================================
```

In [18]:
```
bad_word = ["badword-1", "badword-2", "badword-3"]

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

doc = nlp("Watch out, that badword-1 is not acceptable.")

matches = matcher(doc)

displacy.render(doc, style='dep', options ={ "color" : "red","compact":True})

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(f"Token: {token.text} | POS: {token.text.lower()}")
```



```
Matching Sentence: badword-1
Pattern Type: VulgarLanguage
Token: badword-1 | POS: badword-1
Matching Sentence: badword-1
Pattern Type: OneOrMoreNouns
Token: badword-1 | POS: badword-1
```

Task 6: Getting Replies

In [19]:
```python
def utterance(msg):
    nlp = spacy.load('en_core_web_sm')
    doc = nlp(msg)
    matcher = Matcher(nlp.vocab)
    pattern1 = [{"LEMMA": {"IN": ["salam", "assalam", "hi", "hello"]}}]
    matcher.add("greeting", [pattern1])
    matches = matcher(doc)

    if len(matches) == 0:
        print('Please rephrase your request. Be as specific as possible!')
        return

    for pattern_id, start, end in matches:
        if doc.vocab.strings[pattern_id] == "greeting":
            return "Welcome to Pizza ordering system"

# Example usage
user_input = "Hello"
bot_response = utterance(user_input)

# Print bot response
print("Bot:", bot_response)
```

```
Bot: Welcome to Pizza ordering system
```

In [20]:
```
msg = nlp("Hi")
utterance(msg)
```

Out[20]:
```
'Welcome to Pizza ordering system'
```

In [21]:
```
while True:
    message = input("You: ")
    if message.lower() == "quit":
        break
    else:
        print("Bot:", utterance(nlp(message)))
```

```
You: quit
```

In [23]:
```
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)
```

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

In [25]:
```
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"}
    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)
```

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

In [28]:
```
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.
```

In [ ]:

In [ ]: