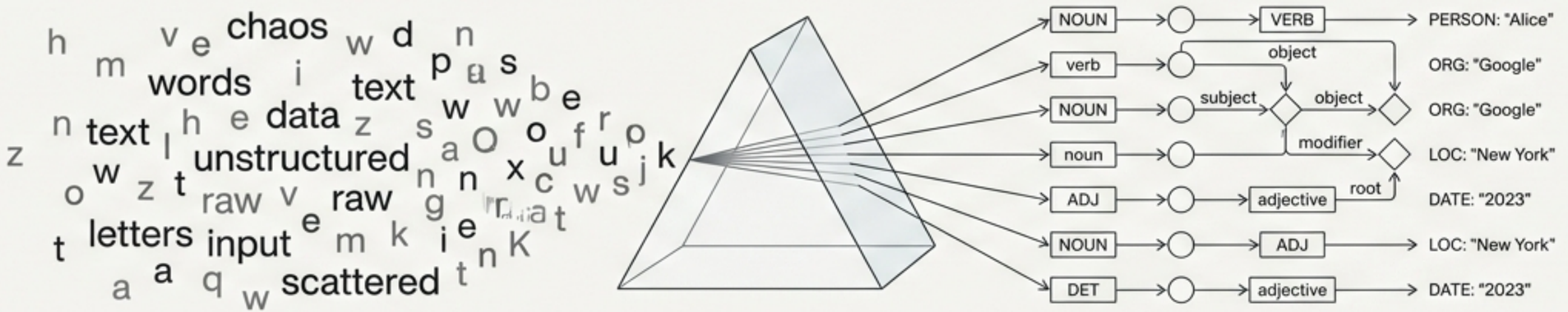
From Words to Meaning

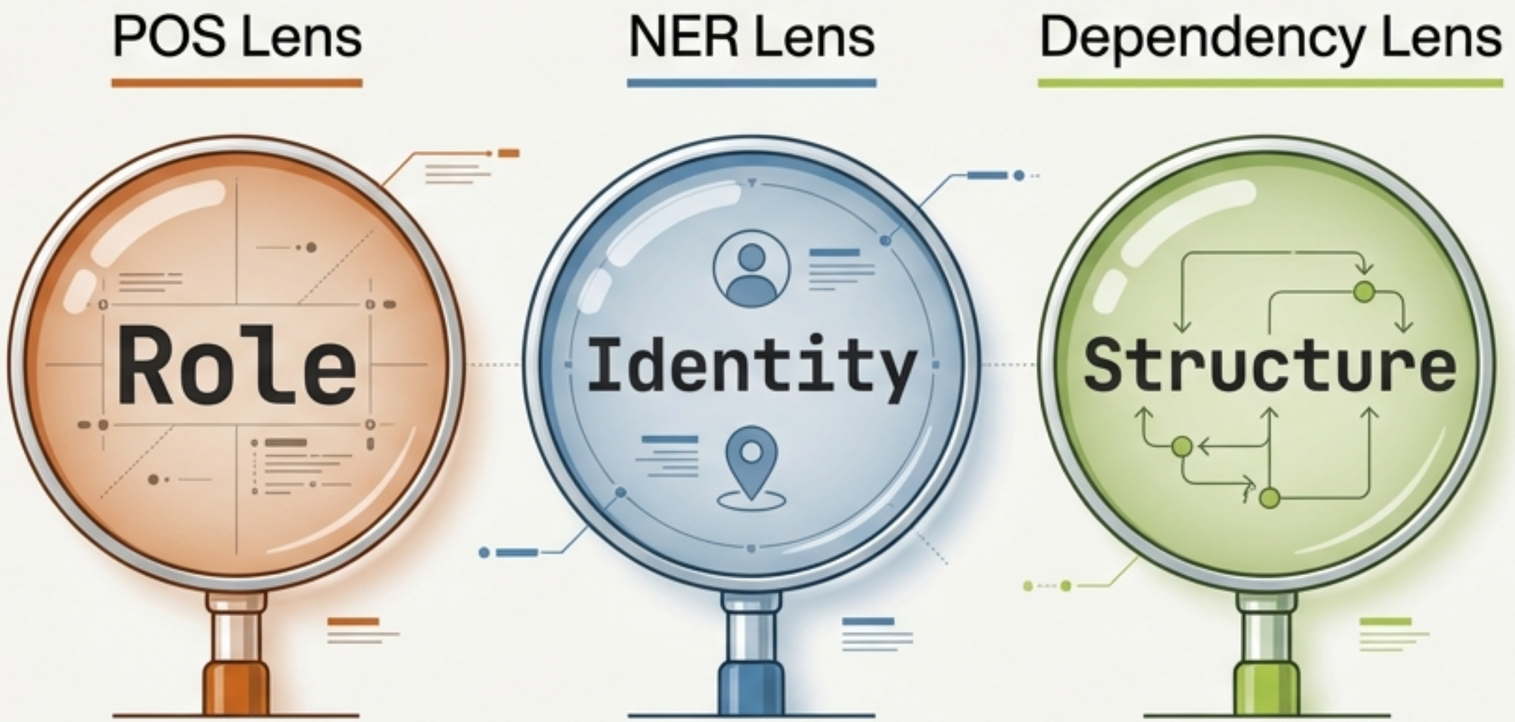
A Simple Guide to NLP: POS Tagging, NER, and Dependency Parsing



Beyond Tokenization

We have moved past the basics of breaking text into tokens. To achieve understanding, we apply three distinct lenses to the data.

→

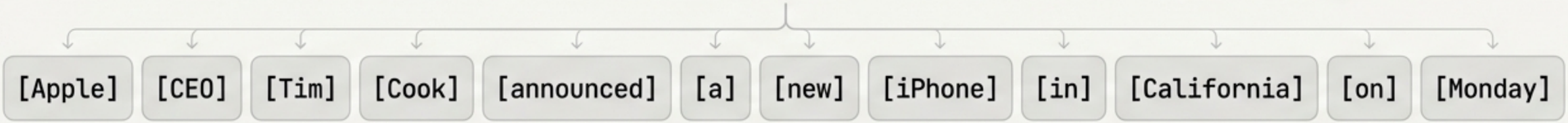


Grammar • Real-World Entities • Relationships

Our Anchor Sentence

We will use this single sentence to demonstrate every layer of the pipeline.

Apple CEO Tim Cook announced a new iPhone in California on Monday.



Layer 1: Part of Speech (POS)

POS tagging assigns a grammatical role to each token.
Context determines the tag, resolving ambiguity.

Book
a ticket

VERB (Action)

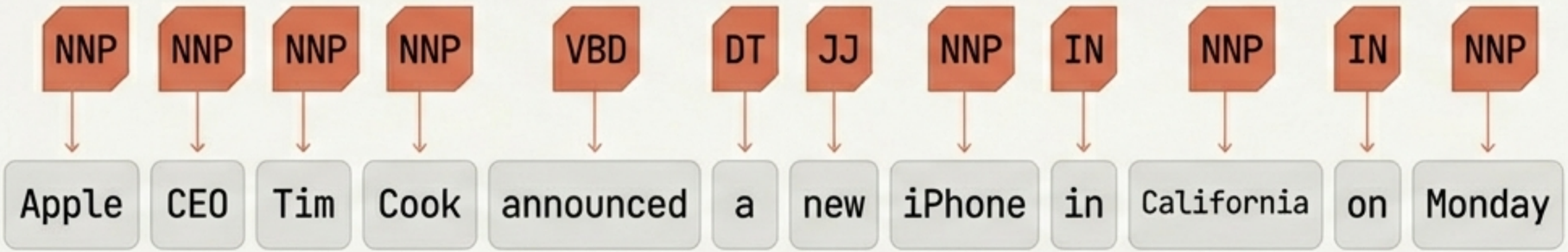
VERB (Action)

Same spelling,
different meaning.

Read
a book

NOUN (Object)

Decoding the Tags

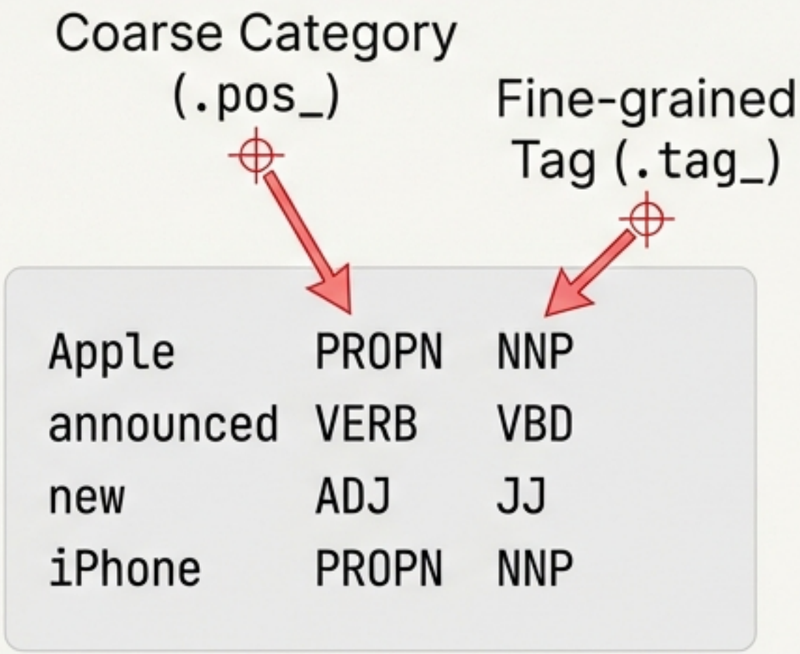


POS Key

- NNP — Proper Noun
- VBD — Verb (Past)
- JJ — Adjective
- DT — Determiner
- IN — Preposition

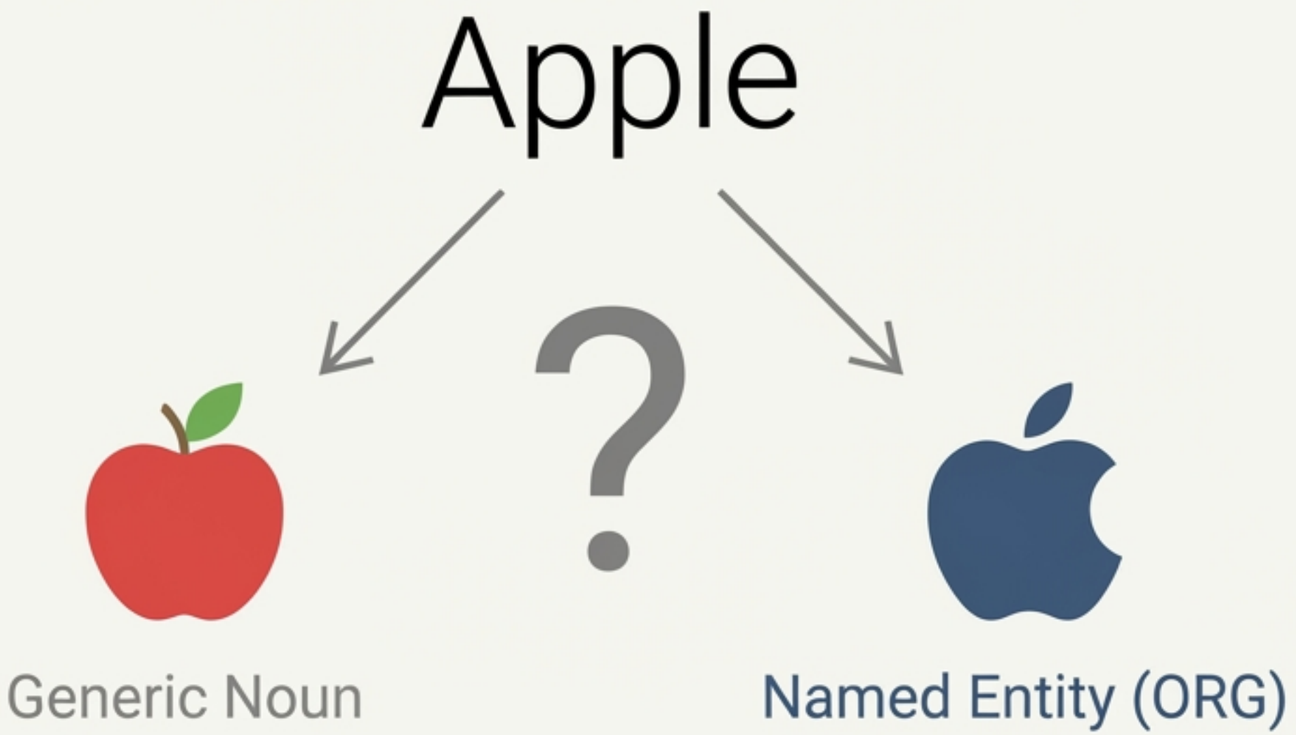
Implementation: POS Tagging

```python
import spacy

nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple CEO Tim Cook announced a new iPhone...")

for token in doc:
    print(token.text, token.pos_, token.tag_)
```

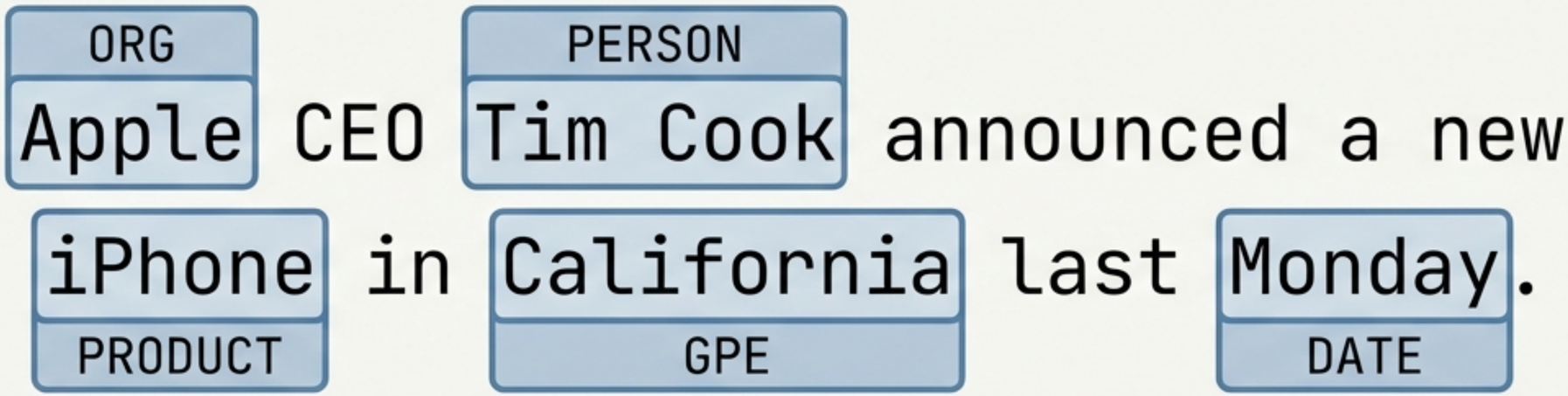


Layer 2: Named Entity Recognition (NER)

NER identifies and classifies real-world objects embedded in the text.
It distinguishes generic nouns from specific entities.



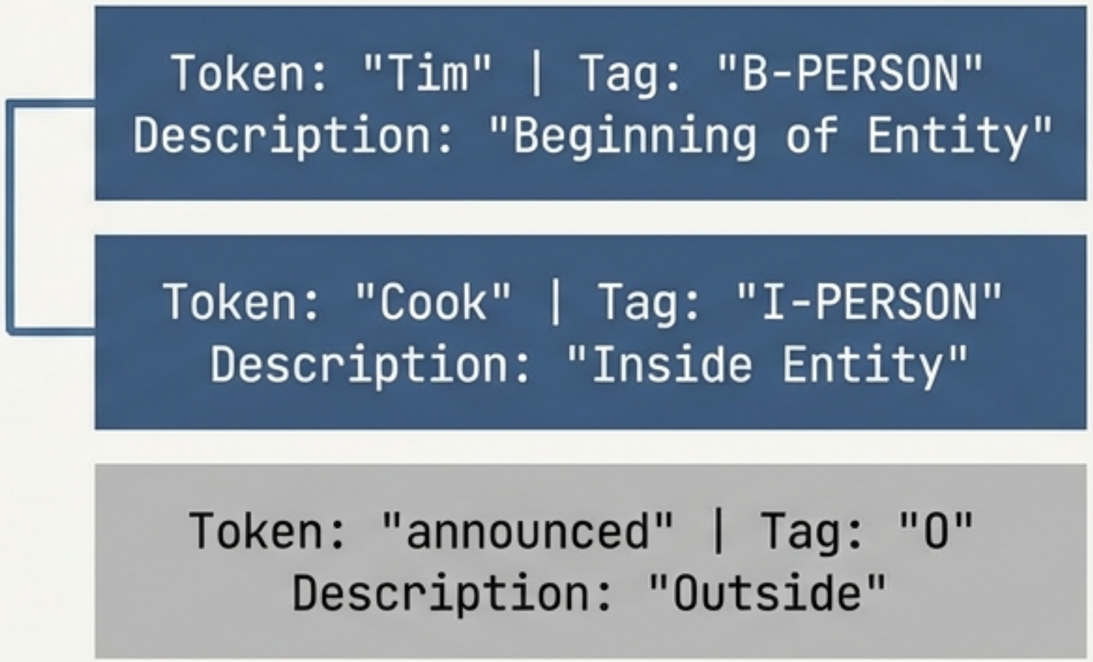
NER in Action



● ORG: Organization 🌍 GPE: Geo-Political Entity 🚗 PRODUCT: Object/Vehicle/Food

How Machines See Entities: The IOB Method

To capture multi-word entities like ‘Tim Cook”, models use Inside-Outside-Beginning tags.



Implementation: NER & IOB

Extracting Entities

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

```
Tim Cook  PERSON
```

Inspecting IOB Tags

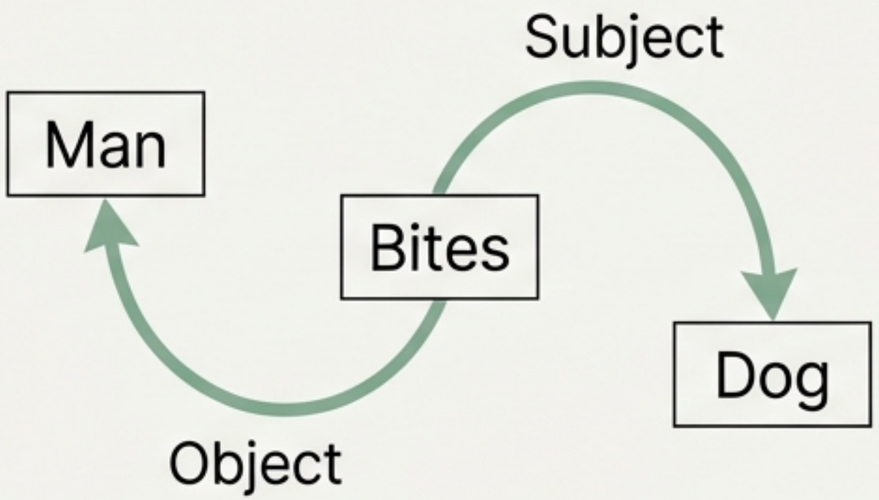
```
for token in doc:
    print(token.text, token.ent_iob_,
          token.ent_type_)
```

```
Tim    B    PERSON
Cook   I    PERSON
Apple  B    ORG
```

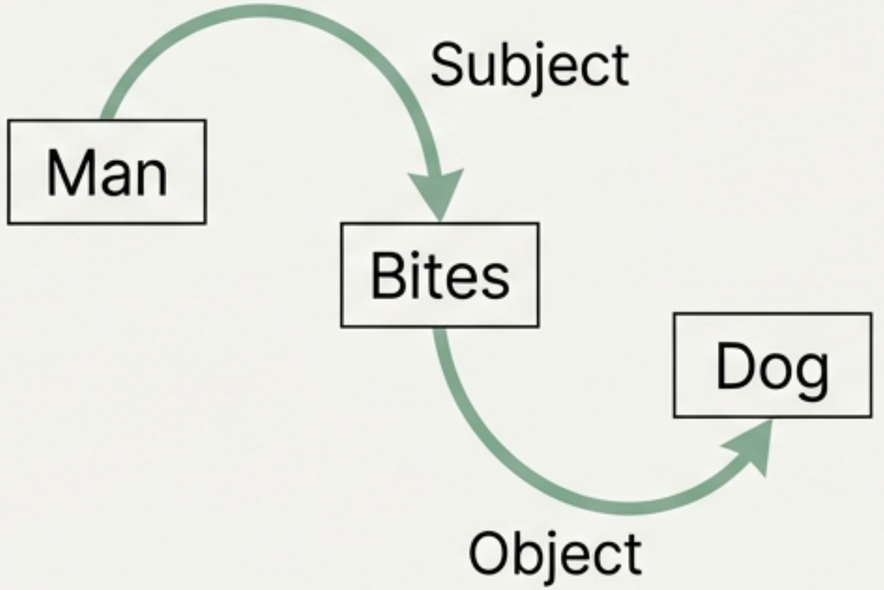
Layer 3: Dependency Parsing

This layer maps the grammatical structure. It identifies the Subject, the Root (Verb), and the Object to determine “who did what to whom”.

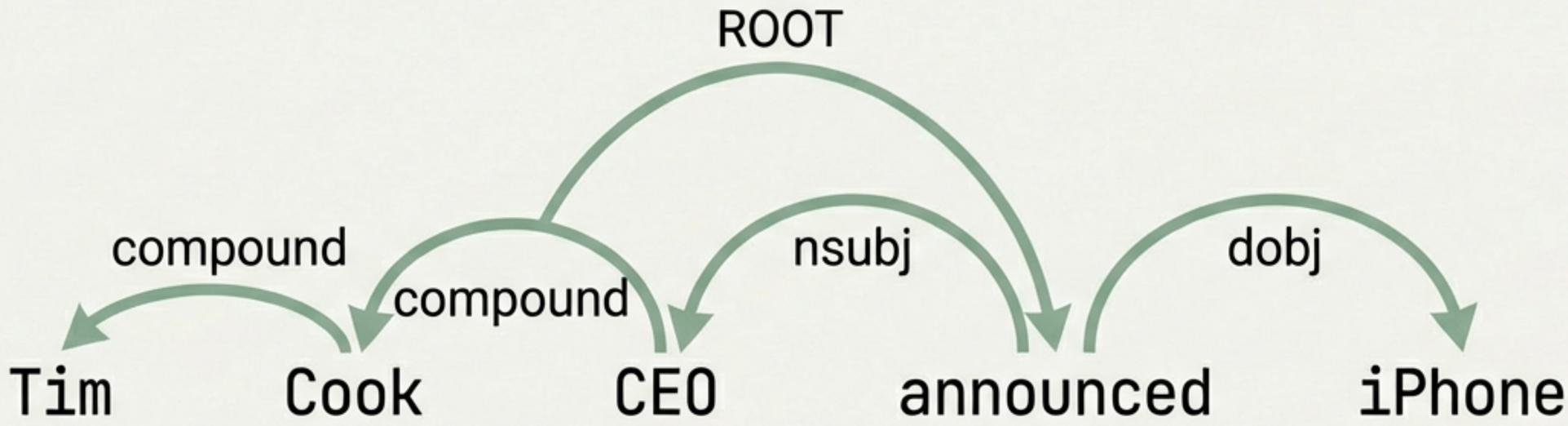
Scenario A: “Dog bites man.”



Scenario B: “Man bites dog.”



Visualizing Relationships



Tim Cook (Subject) + Announced (Root) + iPhone (Object)

Dependency parsing visualizes the grammatical structure and relationships within a sentence, making it possible to extract meaning.

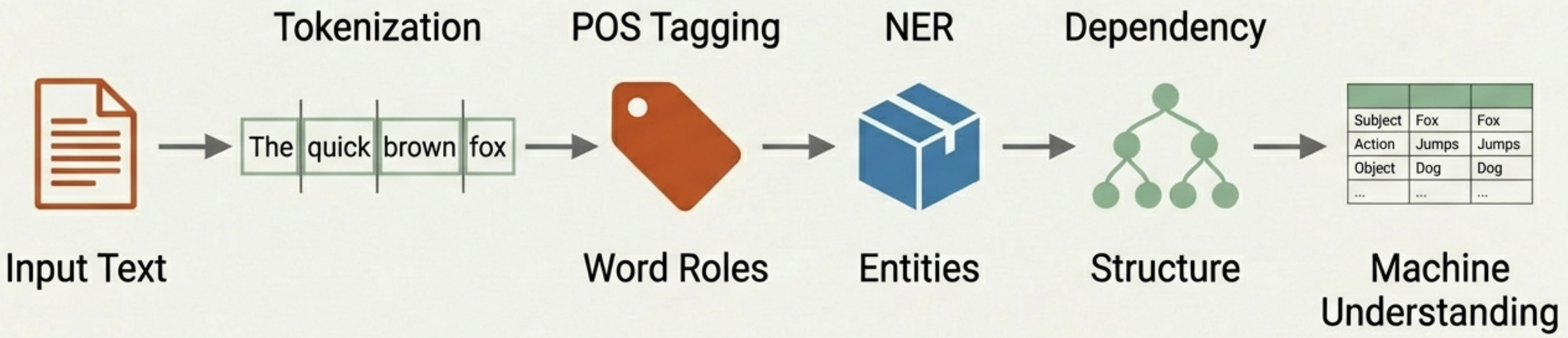
Implementation: Dependency Parsing

```
for token in doc:
    print(f"{token.text}
      --({token.dep_})-->
      {token.head.text}")
```

```
.
"Cook       --(nsubj)-->
          announced"
"announced   --(ROOT)-->
             --(ROOT)-->
          announced"

"iPhone      --(dobj)-->
             announced"

"new         --(amod)-->
             iPhone"
.
```

nsubj: Nominal Subject
dobj: Direct Object
amod: Adjective Modifier

The Complete Understanding Pipeline



Why This Matters: Real-World Applications

Chatbots

Requires POS & Dependency to understand complex commands.
E.g., Distinguishing 'Book a flight' (Action) from 'Read a book' (Object).

Search Engines

Requires NER to differentiate specific entities.
E.g., Knowing that 'Apple' refers to the company, not the fruit, based on context.

Text Summarization

Requires Dependency Parsing to identify the main Subject and Action (Root) to strip away non-essential adjectives.