Natural Language Processing MUD

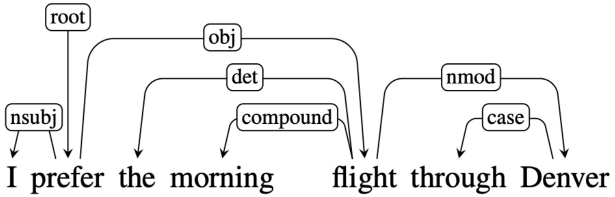
Peter Ganong and Maggie Shi

November 20, 2024

A better library? (via Professor Levy)

- None of these standard NLP libraries are really meant for a one-size-fits-all approach, because they're only as good as what they're trained on. It isn't that you need a "better" library; it's that you need a customized one to your application.
- Example: a student I had who was in the State Department, and she was analyzing votes and commentary on UN resolutions to look at whether increased money from China led to more favorable diplomatic stances. She wanted to get sentiment scores from the comments, but ran into the problem that "diplomatic speak" uses a super sanitized, very specific set of terms. Something that colloquially sounds very mildly negative, or even positive, may actually be a heavy-handed diplomatic smack-down. There's no solution to that except a sentiment model trained specifically on diplomatic texts. Likewise, a sentiment model trained on those diplomatic texts would do a terrible job judging sentiment in news articles, Twitter data, and so on.

Dependency Grammars I



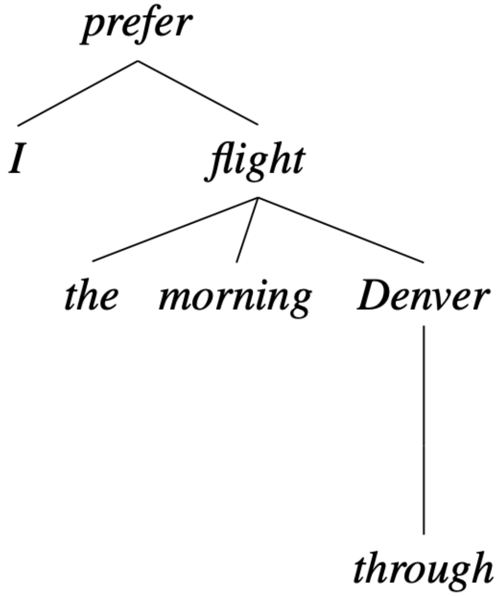
(19.1)

Relations among the words are illustrated above the sentence with directed, labeled arcs from **heads** to **dependents**. We call this a **typed dependency structure** because the labels are drawn from a fixed inventory of grammatical relations. A *root* node explicitly marks the root of the tree, the head of the entire structure.

Key new idea: Every sentence as a "root". Here it is the verb.

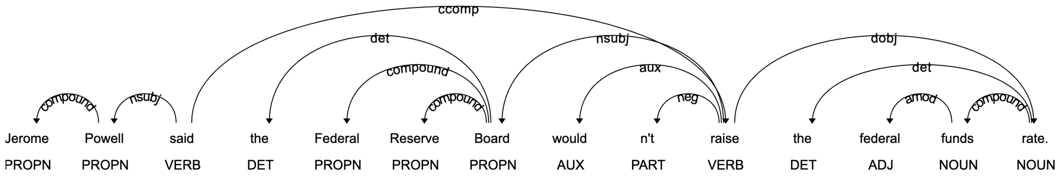
Source: Speech and Language Processing textbook by Jurasfky and Martin. **Link**

Dependency Grammars II



Revisit our example

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Jerome Powell said the Federal Reserve Board wouldn't
↪  raise the federal funds rate.")
```



said has children but not ancestors

```
for token in doc:
    if token.text == "said":
        for child in token.children:
            print(f"{child.text}")
```

```
Powell
raise
.
```

```
for token in doc:
    if token.text == "said":
        for anc in token.ancestors:
            print(f"{anc.text}")
```

Sentiment Analysis: subjectivity

It's just using labels word by word (but for subjective vs objective instead of positive vs negative)

```
from spacytextblob.spacytextblob import SpacyTextBlob
nlp = spacy.load('en_core_web_sm')
nlp.add_pipe('spacytextblob')
doc = nlp("I love this beautiful painting; it's so amazing!")
print(doc._.blob.subjectivity)
print([token._.blob.subjectivity for token in doc])
```

```
0.8333333333333334
[0.0, 0.6, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.9, 0.0]
```