visualize_pcfg_personal

December 2, 2025

[1]:
```
import json

with open("models/pcfg_personal.json", "r", encoding="utf-8") as f:
    pcfg = json.load(f)

# list all nonterminals
sorted(pcfg.keys())[:50]

# inspect NP in detail
lhs = "NP"
for rule in sorted(pcfg[lhs], key=lambda r: r["prob"], reverse=True)[:20]:
    print(lhs, "->", " ".join(rule["rhs"]), "[", rule["prob"], "]")
```

```
NP -> NP PP [ 0.1025508889461479 ]
NP -> NP NN [ 0.0770419994846689 ]
NP -> NP NP [ 0.07034269518165422 ]
NP -> NP , [ 0.06956969853130636 ]
NP -> NN [ 0.06518938417933522 ]
NP -> DT NN [ 0.05745941767585674 ]
NP -> NP NNP [ 0.04637979902087091 ]
NP -> NP CC [ 0.03968049471785622 ]
NP -> DT JJ [ 0.03761917031692863 ]
NP -> PRP [ 0.03040453491368204 ]
NP -> NNS [ 0.03014686936356609 ]
NP -> NNP [ 0.027312548312290647 ]
NP -> NP NNS [ 0.022416902860087608 ]
NP -> NNP , [ 0.022159237309971658 ]
NP -> NP  [ 0.021901571759855708 ]
NP -> JJ NN [ 0.021643906209739758 ]
NP -> JJ NNS [ 0.02061324400927596 ]
NP -> NP SBAR [ 0.01958258180881216 ]
NP -> DT [ 0.013656274156145324 ]
NP -> NNP NNP [ 0.012883277505797475 ]
```

[2]:
```
def pretty_print_lhs(pcfg: dict, lhs: str, top_n: int = 10) -> None:
    if lhs not in pcfg:
        print(f"{lhs} not in grammar")
        return
```

```
    print(f"Rules for {lhs}:")
    rules = sorted(pcfg[lhs], key=lambda r: r["prob"], reverse=True)[:top_n]
    for r in rules:
        rhs = " ".join(r["rhs"])
        print(f"  {lhs:5s} - {rhs:25s}  (p = {r['prob']:.3f})")
```

[3]:
```
pretty_print_lhs(pcfg, "S")
pretty_print_lhs(pcfg, "VP")
```

```
Rules for S:
  S     - NP VP                     (p = 0.259)
  S     - S .                       (p = 0.207)
  S     - S VP                      (p = 0.103)
  S     - S NP                      (p = 0.074)
  S     - S ,                       (p = 0.052)
  S     - S S                       (p = 0.041)
  S     - S CC                      (p = 0.031)
  S     - ADVP ,                    (p = 0.023)
  S     - PP ,                      (p = 0.020)
  S     - S                         (p = 0.019)
Rules for VP:
  VP    - VP PP                     (p = 0.084)
  VP    - VB NP                     (p = 0.075)
  VP    - VP VP                     (p = 0.068)
  VP    - VP ,                      (p = 0.059)
  VP    - VP CC                     (p = 0.046)
  VP    - VBD NP                    (p = 0.037)
  VP    - VBN PP                    (p = 0.034)
  VP    - MD VP                     (p = 0.032)
  VP    - VP NP                     (p = 0.032)
  VP    - VBZ NP                    (p = 0.030)
```

[4]:
```
import matplotlib.pyplot as plt

def plot_rule_probs(pcfg: dict, lhs: str, top_n: int = 10):
    rules = sorted(pcfg[lhs], key=lambda r: r["prob"], reverse=True)[:top_n]
    labels = [" ".join(r["rhs"]) for r in rules]
    probs = [r["prob"] for r in rules]

    plt.figure()
    plt.barh(range(len(probs)), probs)
    plt.yticks(range(len(probs)), labels)
    plt.gca().invert_yaxis()
    plt.xlabel("Probability")
    plt.title(f"Top {top_n} rules for {lhs}")
    plt.tight_layout()
    plt.show()
```

[5]:
```
plot_rule_probs(pcfg, "S")
```