gpt_visualize_pcfg_out

December 2, 2025

[6]:
```
import json

with open("models/pcfg_gpt.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 -> NN [ 0.10329171396140749 ]
NP -> NP NN [ 0.08376844494892167 ]
NP -> NP PP [ 0.08104426787741204 ]
NP -> DT NN [ 0.07945516458569807 ]
NP -> NP NP [ 0.06538024971623156 ]
NP -> NP , [ 0.05743473325766175 ]
NP -> NNS [ 0.05402951191827469 ]
NP -> PRP [ 0.04790011350737798 ]
NP -> NP CC [ 0.039727582292849034 ]
NP -> DT JJ [ 0.03881952326901249 ]
NP -> JJ NNS [ 0.037911464245175934 ]
NP -> JJ NN [ 0.03745743473325766 ]
NP -> NP SBAR [ 0.027014755959137344 ]
NP -> NP NNS [ 0.02292849035187287 ]
NP -> DT NNS [ 0.017707150964812714 ]
NP -> NP : [ 0.011577752553916005 ]
NP -> NN NN [ 0.011350737797956867 ]
NP -> NN NNS [ 0.009534619750283769 ]
NP -> DT [ 0.009307604994324632 ]
NP -> NP JJ [ 0.008626560726447218 ]
```

[7]:
```
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})")
```

[8]:
```
pretty_print_lhs(pcfg, "S")
pretty_print_lhs(pcfg, "VP")
```

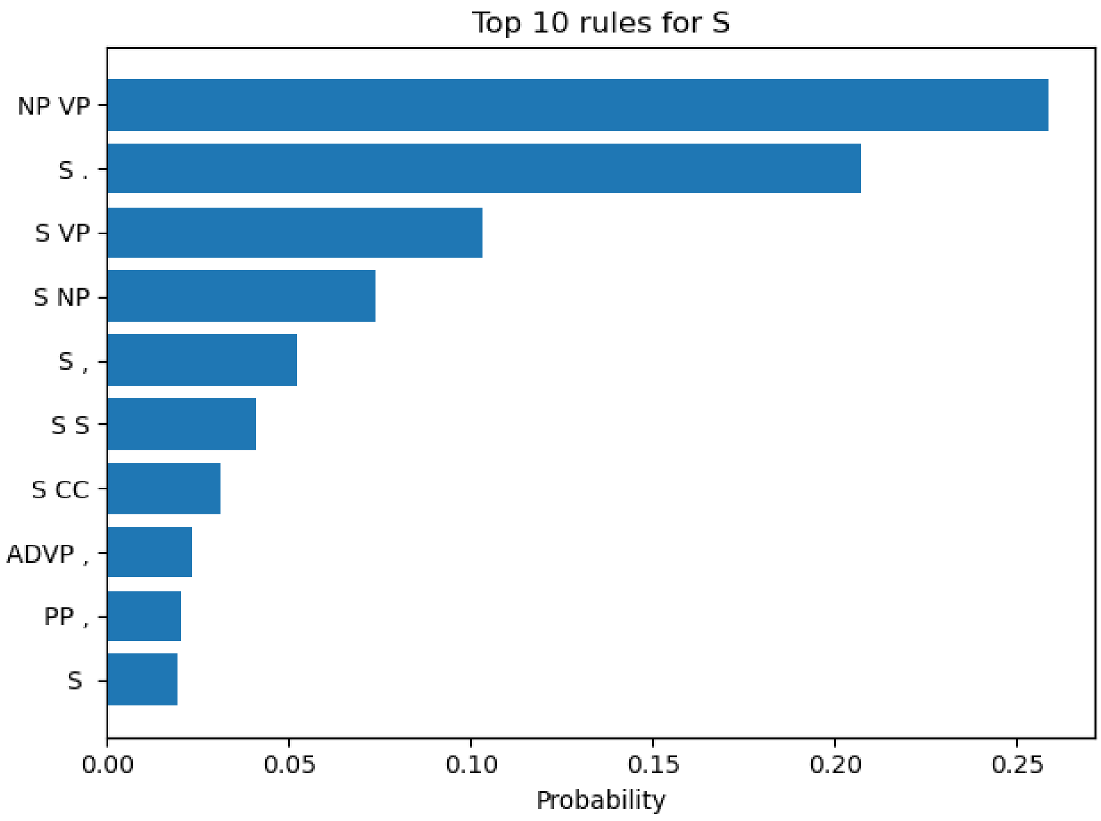
```
Rules for S:
  S     - NP VP                      (p = 0.311)
  S     - S .                        (p = 0.269)
  S     - S VP                       (p = 0.100)
  S     - S NP                       (p = 0.060)
  S     - S S                        (p = 0.042)
  S     - S ,                        (p = 0.028)
  S     - PP ,                       (p = 0.025)
  S     - SBAR ,                     (p = 0.020)
  S     - S :                        (p = 0.019)
  S     - NP ADVP                    (p = 0.019)
Rules for VP:
  VP    - VP PP                      (p = 0.073)
  VP    - VBZ NP                     (p = 0.071)
  VP    - VP VP                      (p = 0.066)
  VP    - VB NP                      (p = 0.065)
  VP    - VBD NP                     (p = 0.064)
  VP    - VP ,                       (p = 0.041)
  VP    - VBN PP                     (p = 0.033)
  VP    - VP NP                      (p = 0.030)
  VP    - VBP NP                     (p = 0.030)
  VP    - MD VP                      (p = 0.025)
```

[9]:
```
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")
```


Top 10 rules for S
NP VP
S .
S VP
S NP
S ,
S S
S CC
ADVP ,
PP ,
S
Probability