WestlakeNLP

Natural Language Processing

Yue Zhang
Westlake University

西湖大學
WESTLAKE UNIVERSITY

Chapter 10

Predicting Tree Structures

Contents

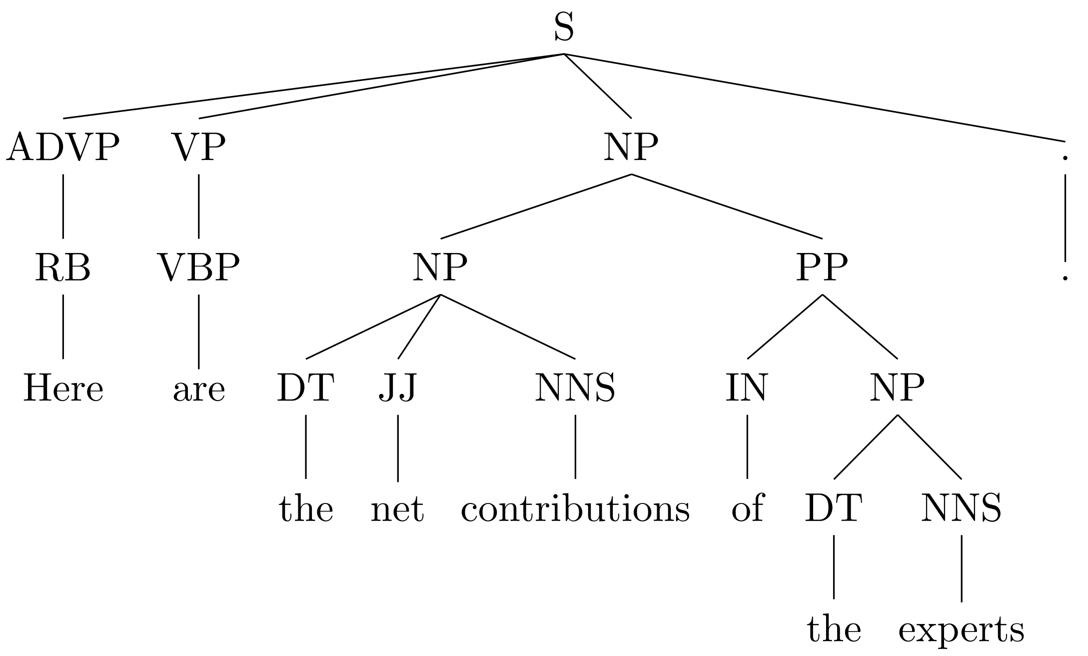
WestlakeNLP

Contents

WestlakeNLP

Generative Constituent Parsing

- Input: Here are the net contributions of the experts.
- Output:



Generative Constituent Parsing

- Input: Here are the net contributions of the experts.
- Output: A constituent tree represented by a bracketed structure.

```
(S
  (ADVP (RB "Here"))
  (VP (VBP "are"))
   (NP
     (NP (DT "the") (JJ "net") (NNS "contributions")))
     (PP (IN "of")(NP (DT "the") (NNS "experts"))))
  (. "."))
```

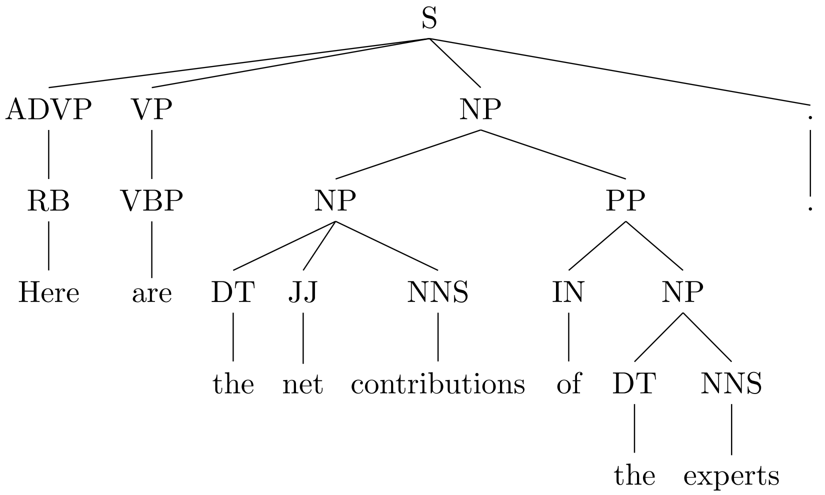
Generative Constituent Parsing

WestlakeNLP

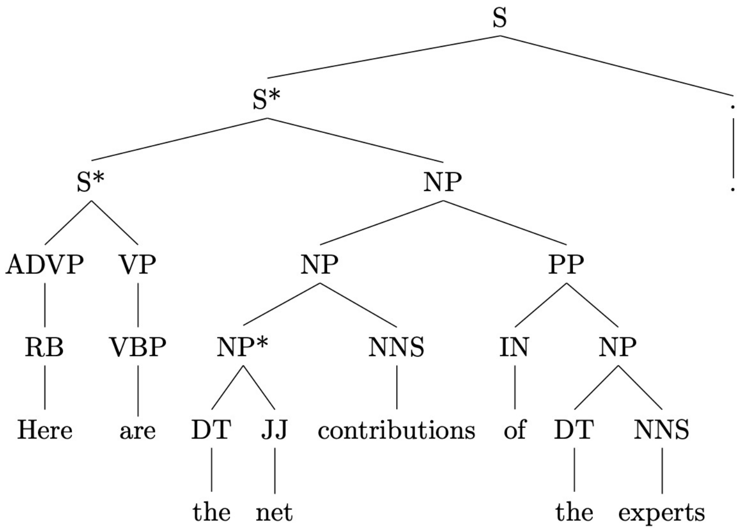
- Constituent tree binarization
  - Necessary for some common algorithms (e.g. CKY; shift-reduce)
- Modes of binarization
  - Left-binarization
  - Right-binarization
  - Head-binarization

Generative Constituent Parsing

- Constituent tree binarization
  - Left-binarization
  - Right-binarization
  - Head-binarization



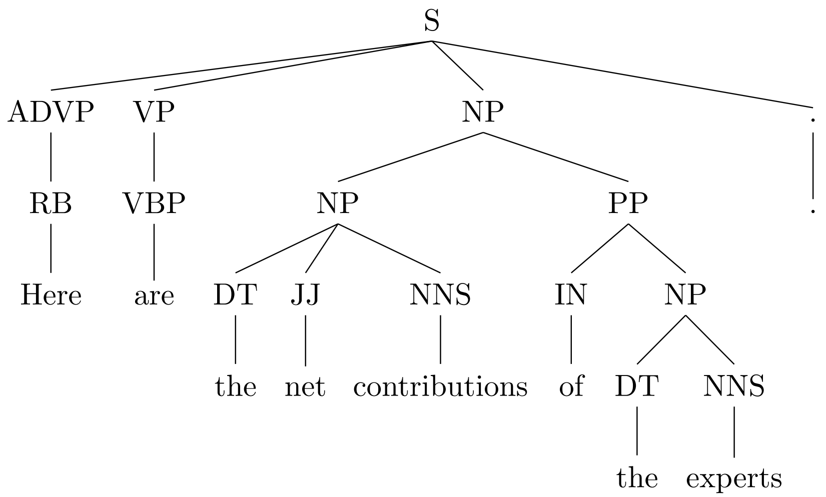
Original tree



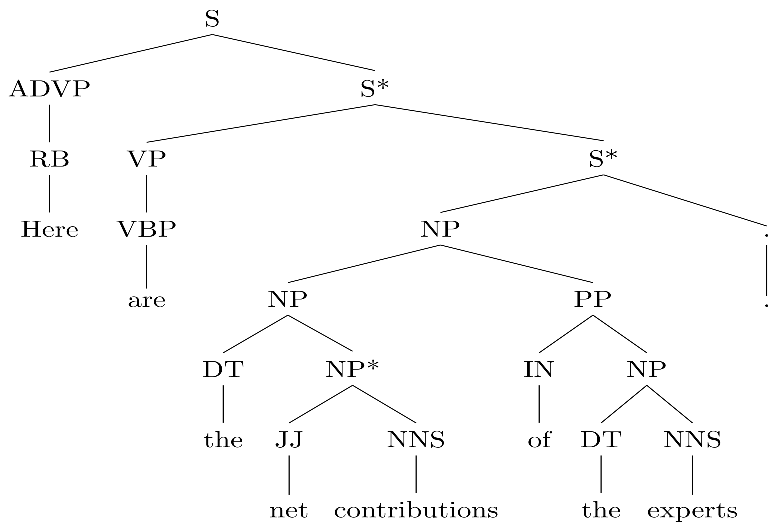
Left-binarization

Generative Constituent Parsing

- Constituent tree binarization
  - Left-binarization
  - Right-binarization
  - Head-binarization



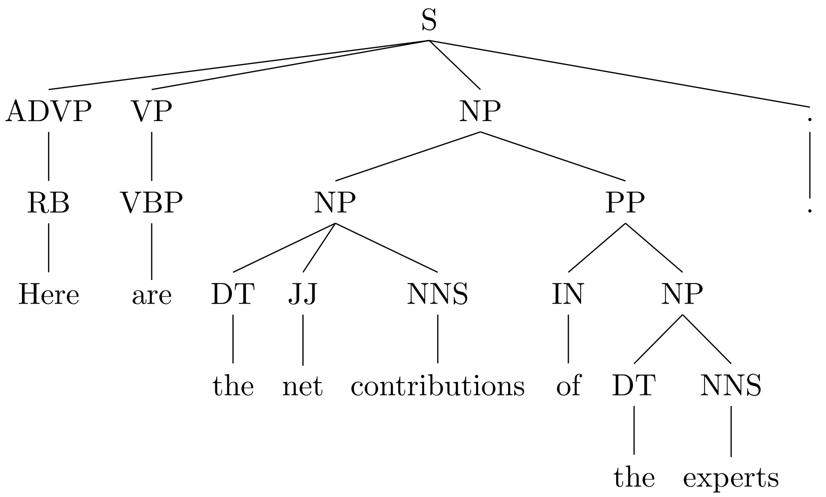
Original tree



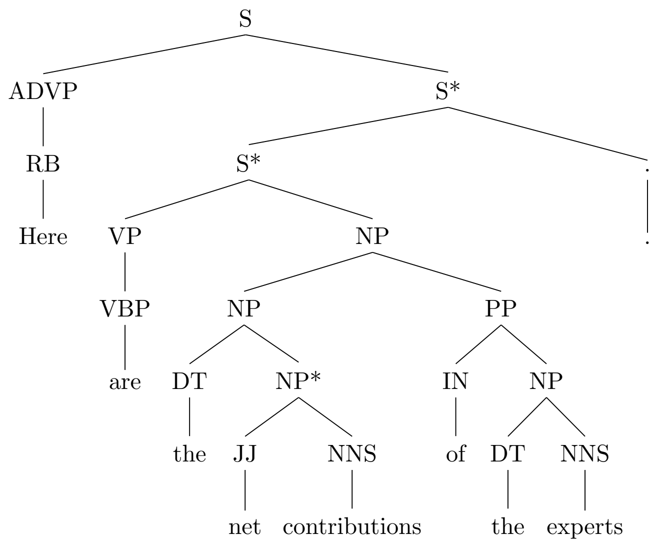
Right-binarization

Generative Constituent Parsing

- Constituent tree binarization
  - Left-binarization
  - Right-binarization
  - Head-binarization



Original tree



Head-binarization

Contents

WestlakeNLP

Probabilistic Context Free Grammar

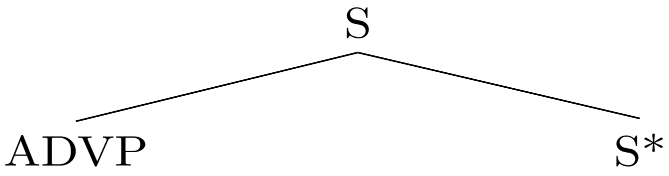
WestlakeNLP

- Derivation

S

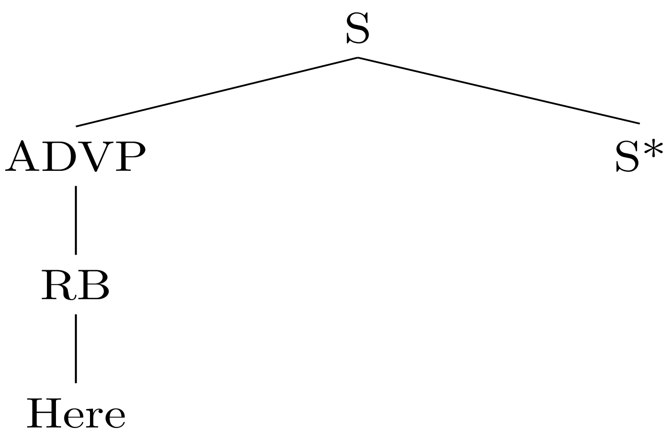
Probabilistic Context Free Grammar

- Derivation



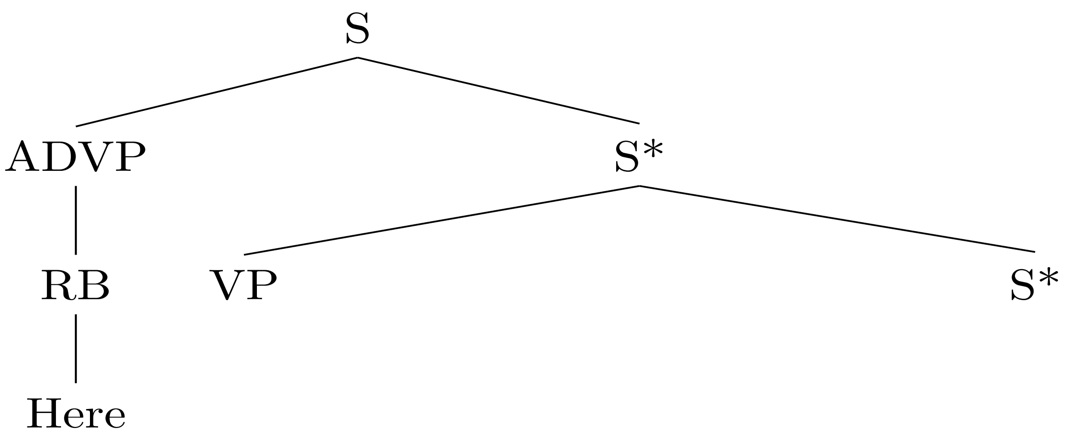
Probabilistic Context Free Grammar

- Derivation



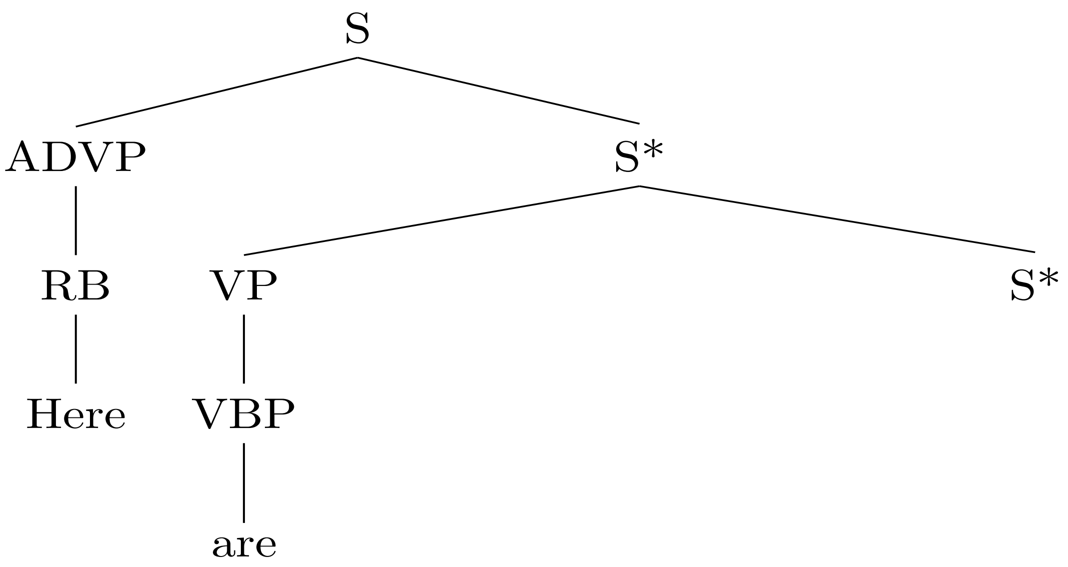
Probabilistic Context Free Grammar

- Derivation



Probabilistic Context Free Grammar

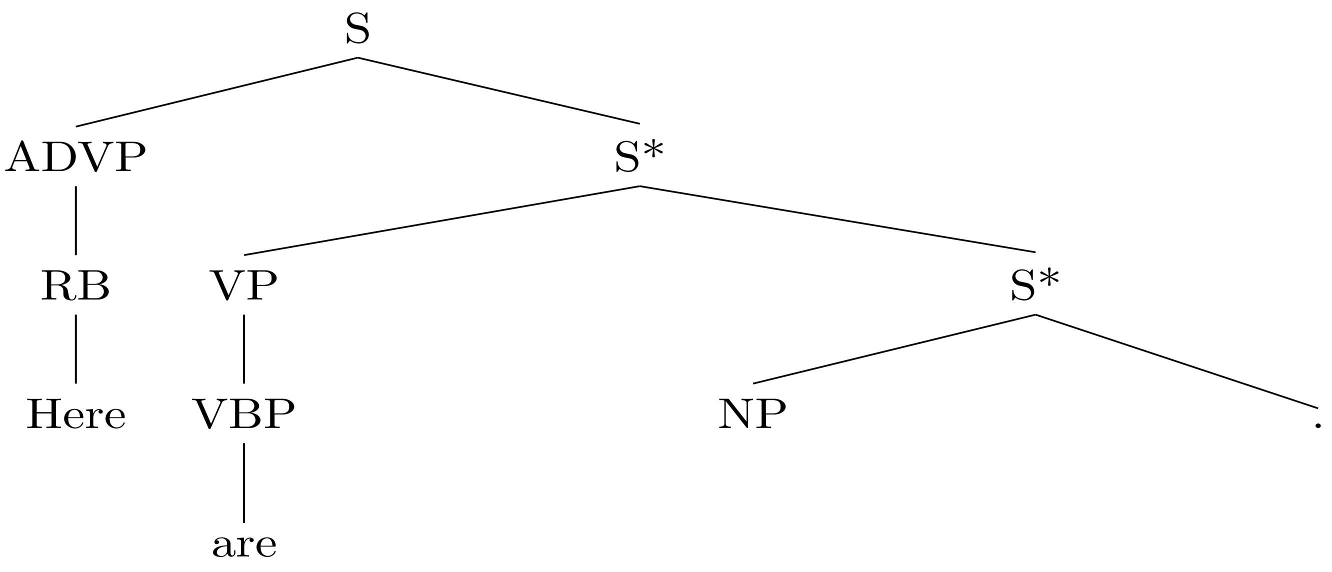
- Derivation



Probabilistic Context Free Grammar

WestlakeNLP

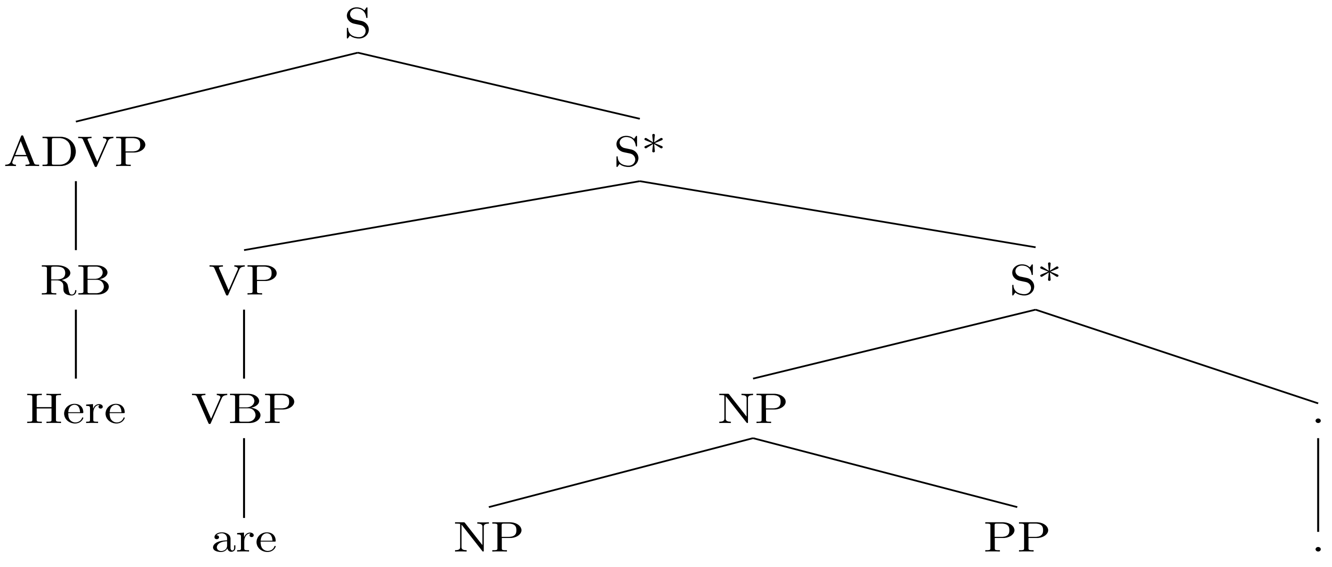
- Derivation



Probabilistic Context Free Grammar

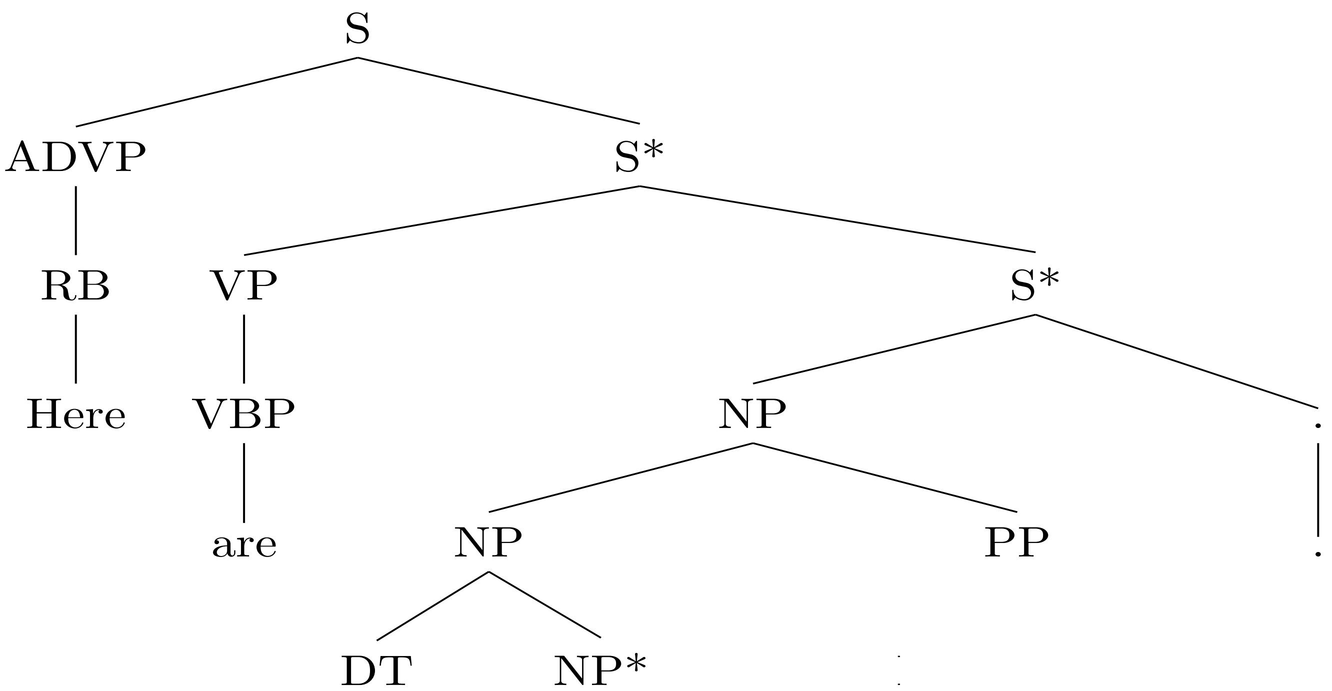
WestlakeNLP

- Derivation



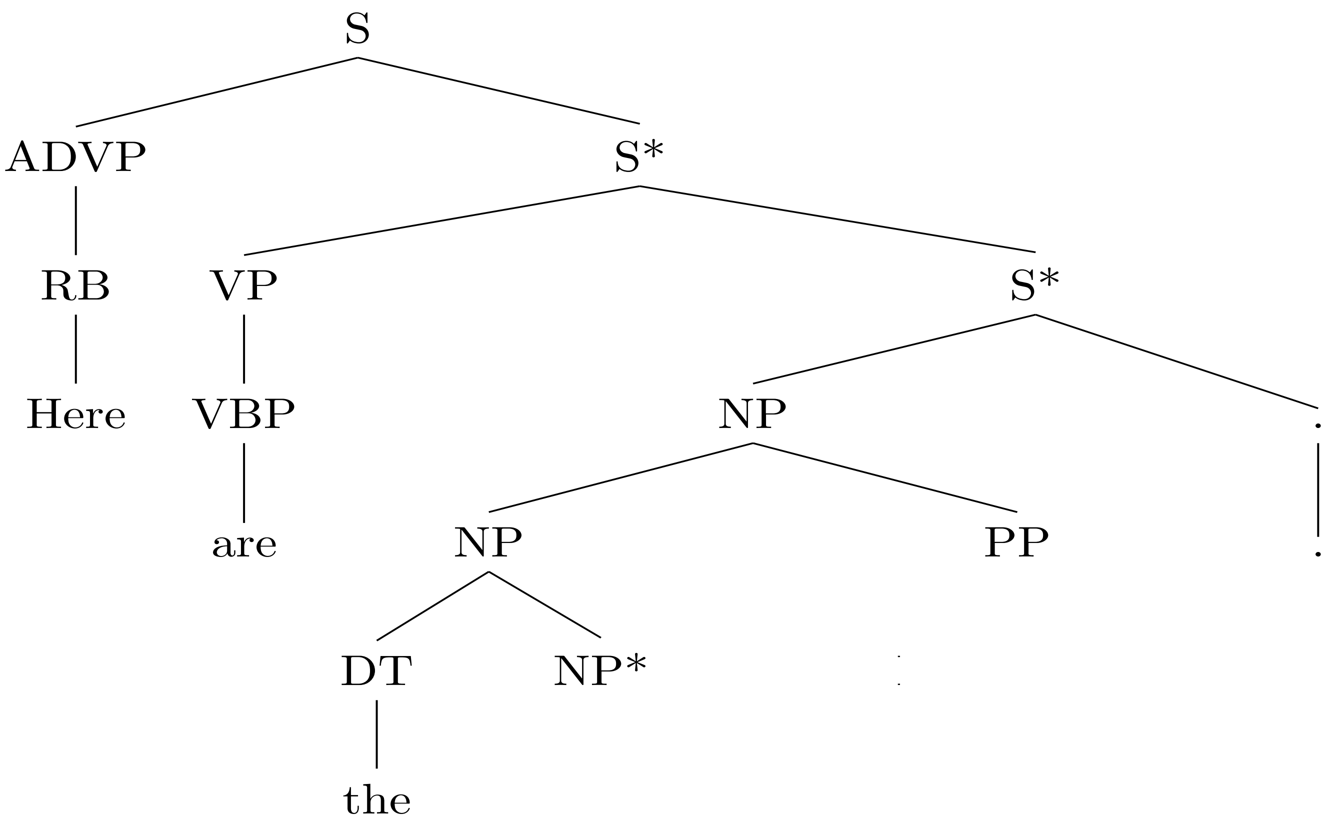
Probabilistic Context Free Grammar

- Derivation



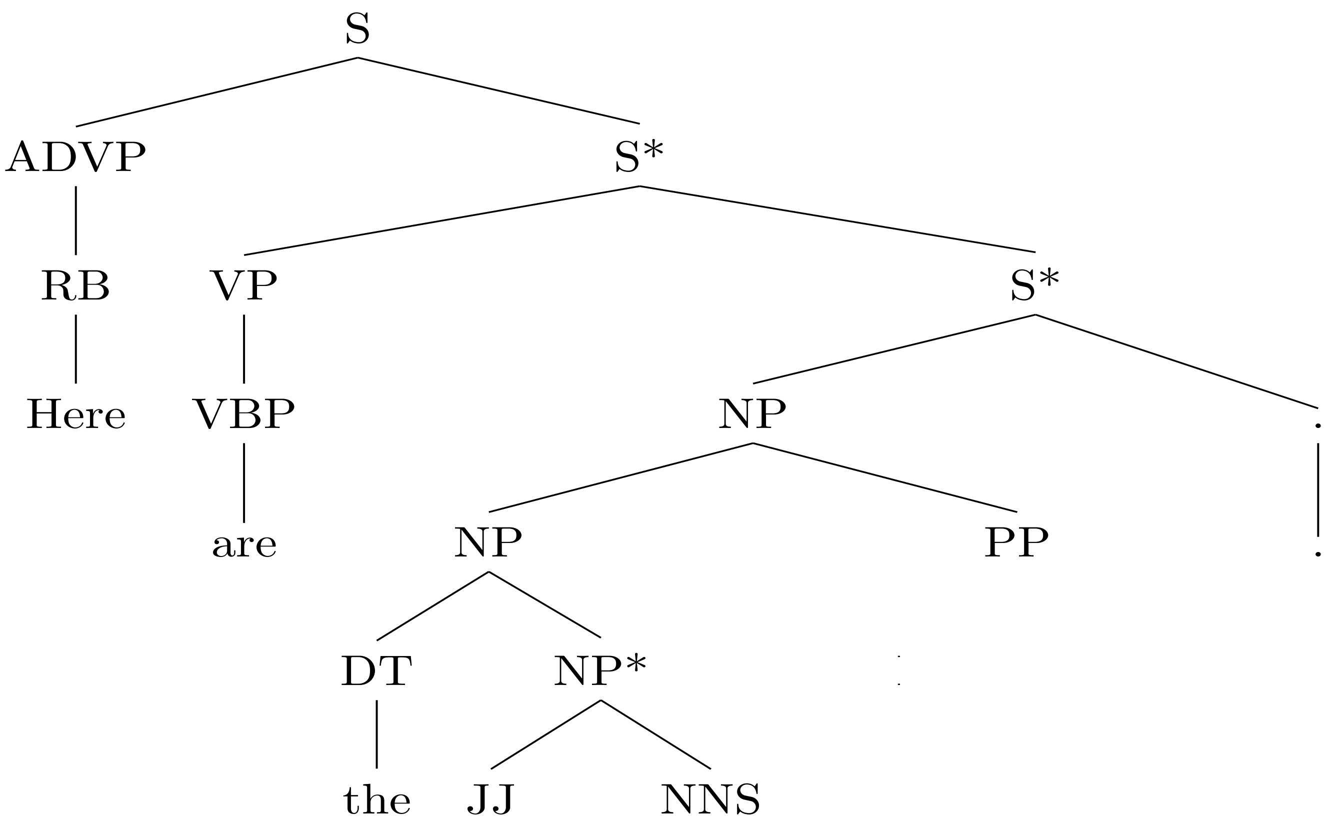
Probabilistic Context Free Grammar

- Derivation



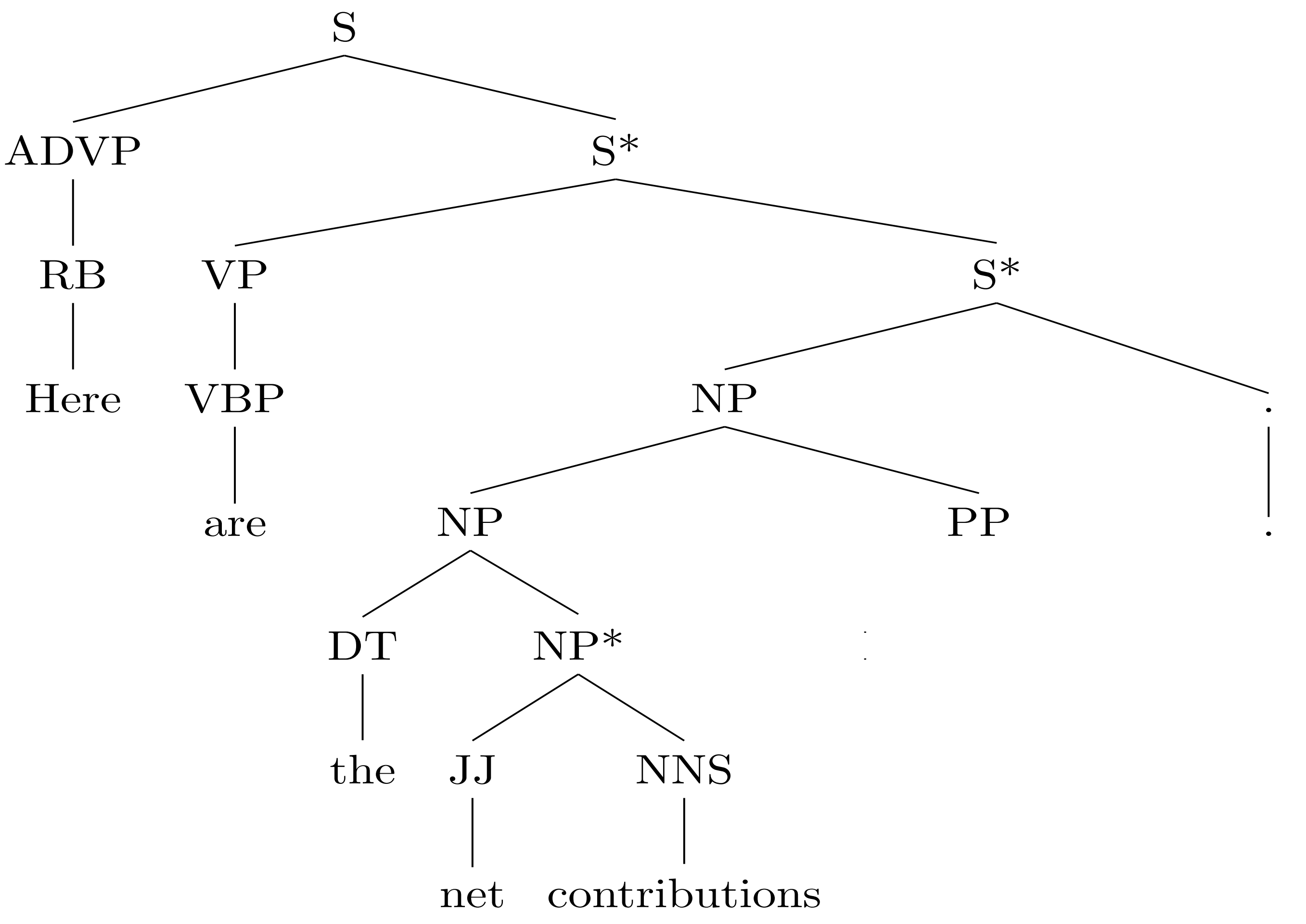
Probabilistic Context Free Grammar

- Derivation



Probabilistic Context Free Grammar

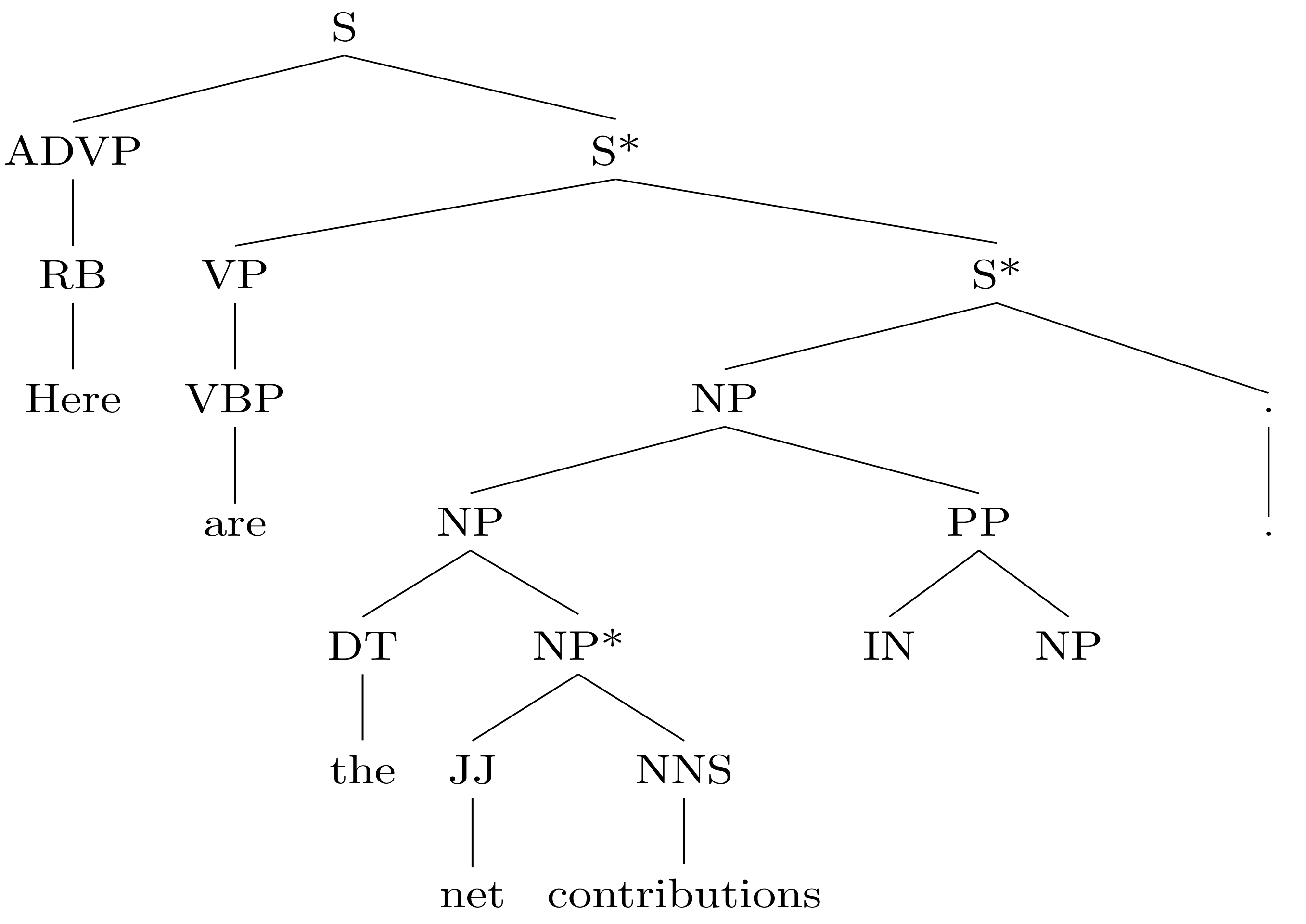
- Derivation



Probabilistic Context Free Grammar

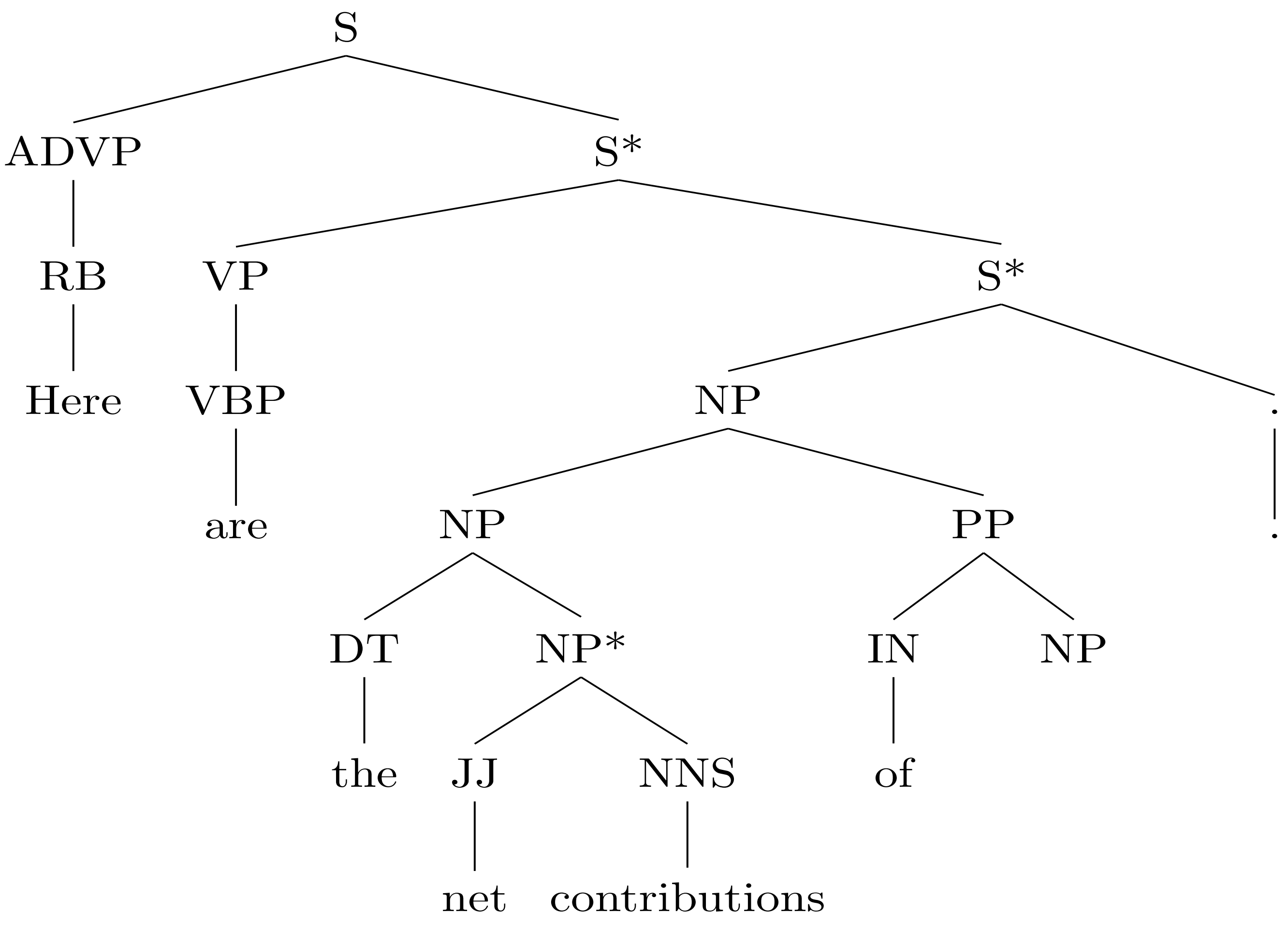
WestlakeNLP

- Derivation



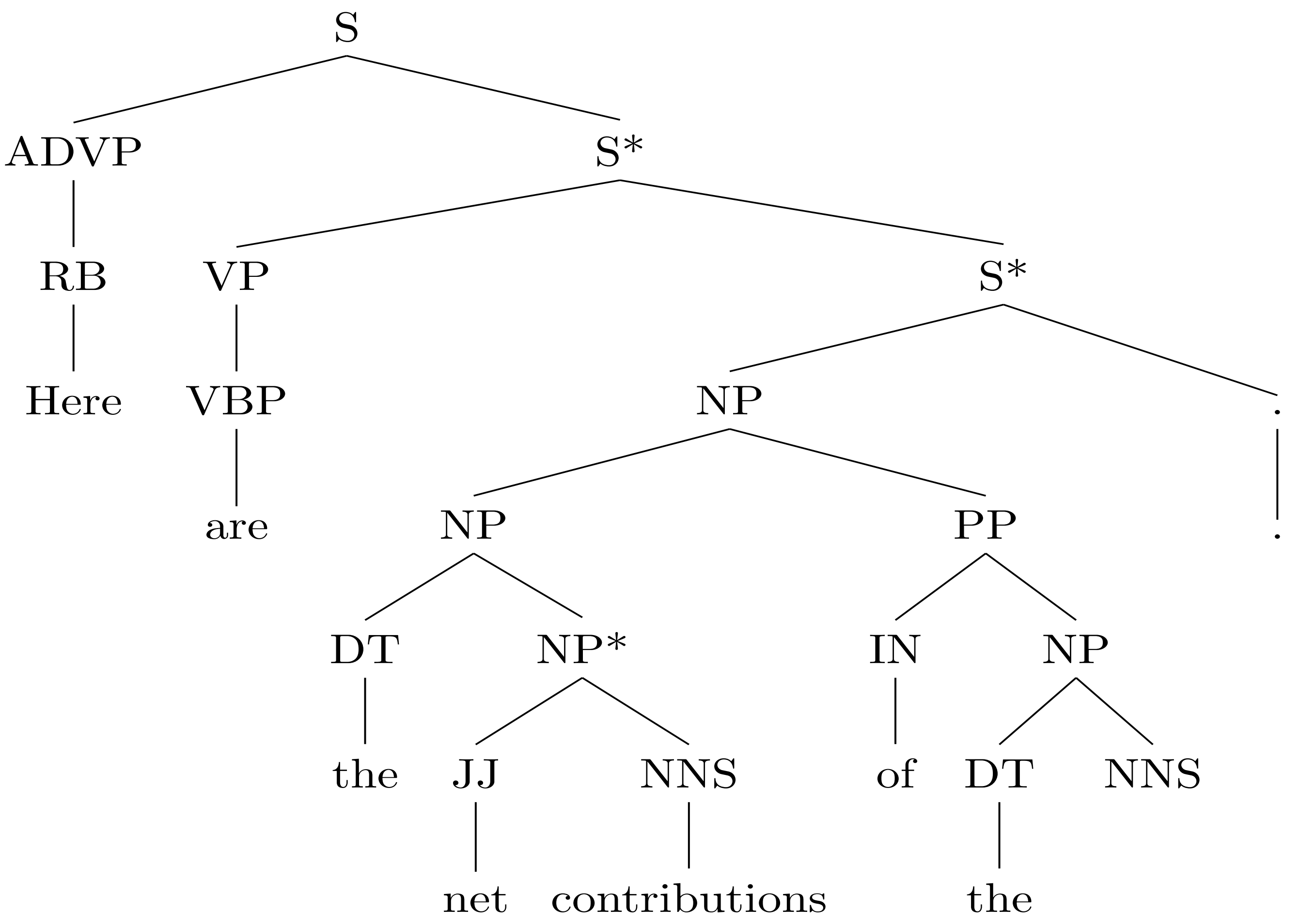
Probabilistic Context Free Grammar

- Derivation



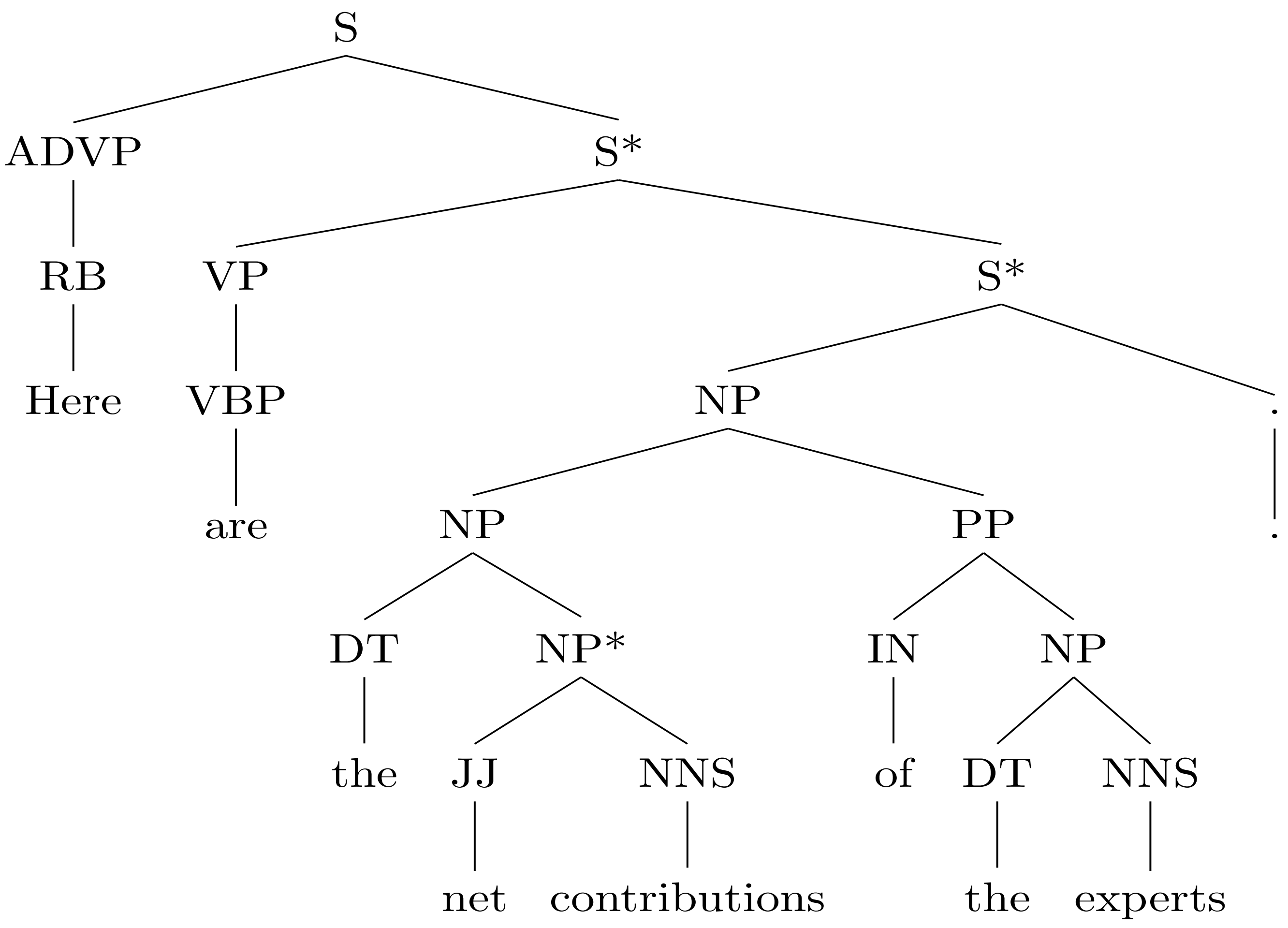
Probabilistic Context Free Grammar

- Derivation



Probabilistic Context Free Grammar

- Derivation



Probabilistic Context Free Grammar

- Context Free Grammars (CFG)

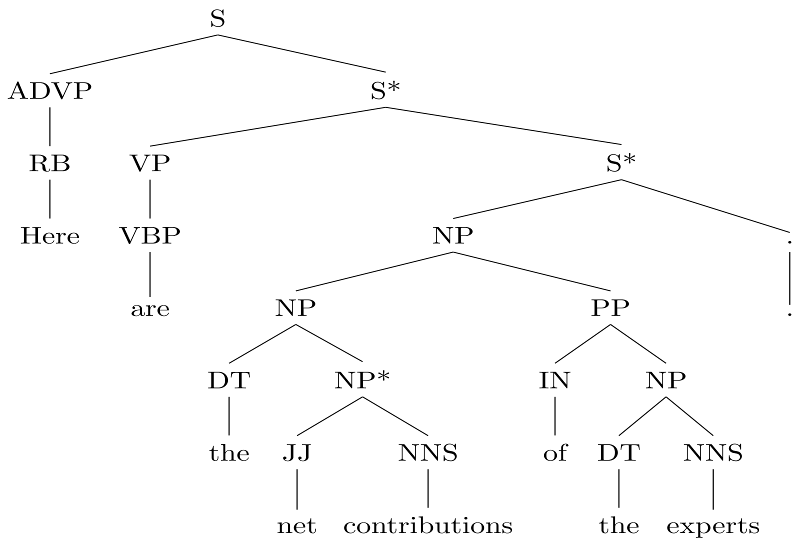
  Formally, a CFG is a 4-tuple: <N,Σ,R,S>.

  - N: the set of non-terminals (i.e. A,B,C,…)
  - Σ: the set of terminals (i.e. α,β,γ,…)
  - R: the set of production rules (i.e. A->BC, A->γ,…)
  - S: the start symbol

Probabilistic Context Free Grammar

- Derivation

  A sequence of rule applications that transforms a non-terminal node into a string is called a derivation.



$$\mathrm{S} \rightarrow \mathrm{ADVP}\ \mathrm{S}^*, \mathrm{ADVP} \rightarrow RB, S^* \rightarrow$$
$$S^*., S^* \rightarrow VP\ NP, VP \rightarrow VBP, NP \rightarrow$$
$$NP\ PP, NP \rightarrow DT\ NP^*, NP^* \rightarrow$$
$$JJ\ NNS, PP \rightarrow IN\ NP, NP \rightarrow DT\ NNS$$

$$RB \rightarrow Here, VBP \rightarrow are, DT \rightarrow$$
$$the, JJ \rightarrow net, NNS \rightarrow contributions,$$
$$IN \rightarrow of, NNS \rightarrow experts$$

Probabilistic Context Free Grammar

- Probabilistic Context Free Grammars (PCFG)
  - A probabilistic context-free grammar (PCFG) is a CFG augmented with rule probabilities.
  - The probability of a grammar rule ($A \rightarrow \gamma$) is denoted as:

$$P(A \rightarrow \gamma)$$

Probabilistic Context Free Grammar

WestlakeNLP

- The probability of a one-step derivation

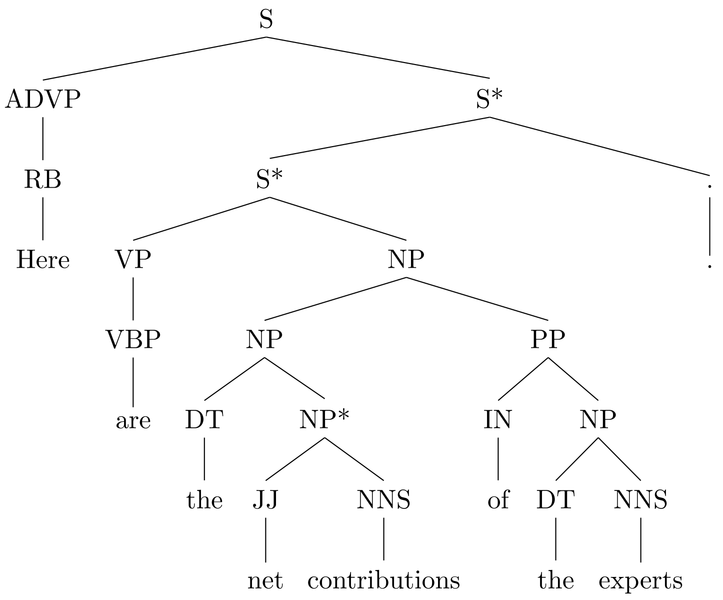
$$P\left(\alpha \xRightarrow{A\to\gamma} \beta\right) = P(A \to \gamma) = P(\gamma|A)$$

- The probability of a multi-step derivation

$$P\left(\alpha \xRightarrow{A_1\to\gamma_1} \beta_1 \xRightarrow{A_2\to\gamma_2} \beta_2 \Longrightarrow \dots \xRightarrow{A_k\to\gamma_k} \beta_k\right) = \prod_{i=1}^{k} P(A_i \to \gamma_i)$$

Probabilistic Context Free Grammar

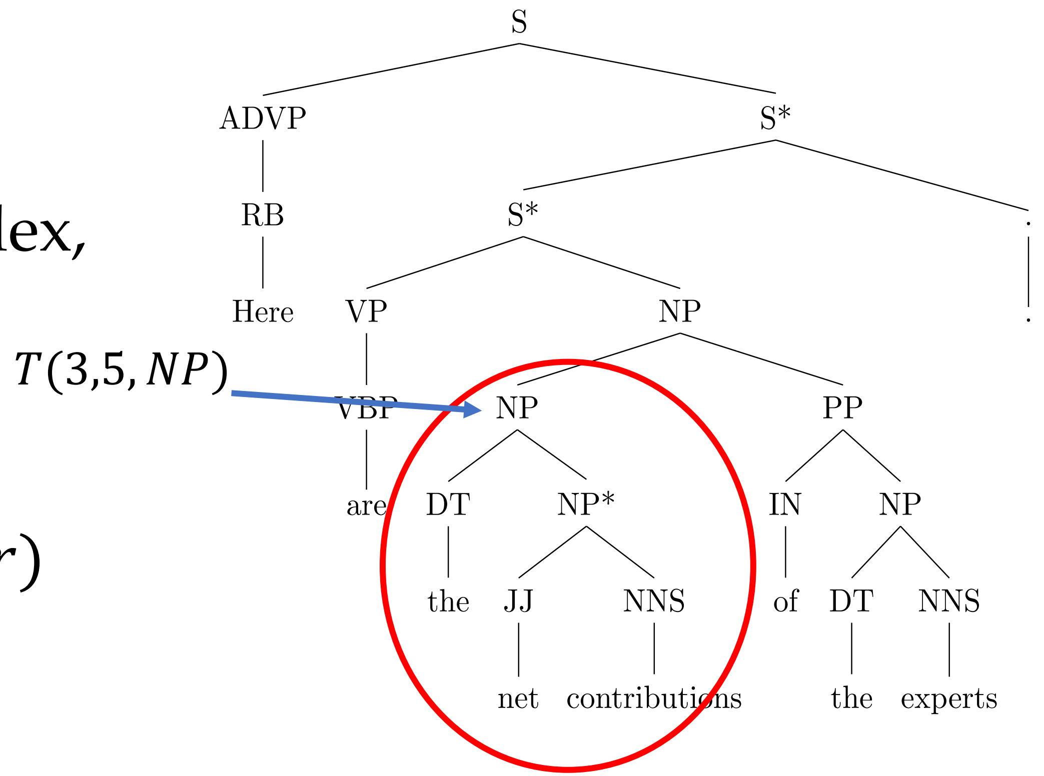
- P(S => Here are the net contributions of the experts .) is:



$$
\begin{aligned}
&\mathrm{P}(\mathrm{S} \rightarrow \mathrm{ADVP}\ \mathrm{S}^{*})\mathrm{P}(\mathrm{ADVP} \rightarrow RB)\mathrm{P}(S^{*} \\
&\rightarrow S^{*}.)\mathrm{P}(S^{*} \rightarrow VP\ NP)\mathrm{P}(VP \rightarrow VBP)\mathrm{P}(NP \\
&\rightarrow NP\ PP)\mathrm{P}(NP \rightarrow DT\ NP^{*})\mathrm{P}(NP^{*} \\
&\rightarrow JJ\ NNS)\mathrm{P}(PP \rightarrow IN\ NP)\mathrm{P}(NP \\
&\rightarrow DT\ NNS)\mathrm{P}(RB \rightarrow Here)\mathrm{P}(VBP \\
&\rightarrow are)\mathrm{P}(DT \rightarrow the, JJ \rightarrow net)\mathrm{P}(NNS \\
&\rightarrow contributions)\mathrm{P}(IN \rightarrow of)\mathrm{P}(NNS \\
&\rightarrow experts)
\end{aligned}
$$

Probabilistic Context Free Grammar

- The probability of a subtree $T(b,e,c)$
  - b and e represent the start and end index,
  - c is the constituent label



$$P\big(T(b,e,c)\big) = \prod_{r \in T(b,e,c)} P(r)$$

- Given $c \text{->} c_1 c_2$,

$$P\big(T(b,e,c)\big) = P\big(T(b,k,c_1)\big)P\big(T(k+1,e,c_2)\big)P(c \text{->} c_1 c_2)$$

Probabilistic Context Free Grammar

- Training PCFG $D = \{(W_i, T_i)\}|_{i=1}^{N}$
  - Given a training corpus, each parameter can be estimated using:

$$P(\gamma|A) = P(A \rightarrow \gamma) = \frac{count(A \rightarrow \gamma)}{count(A)}$$

$$= \frac{\sum_{i=1}^{N} count(A \rightarrow \gamma, T_i)}{\sum_{i=1}^{N} count(A, T_i)}$$

Contents

CKY Decoding

WestlakeNLP

- Given a sentence $W_{1:n} = w_1 w_2 \dots w_n$, the most probable tree is built bottom-up.
- The subtree over the span $w_i w_{i+1} \dots w_{i+s-1}$ with a constituent label c is denoted as: $\hat{T}(i, i+s-1, c)$

$$score(\hat{T}(i, i+s-1, c)) = \max_{c_1, c_2 \in C, j \in [i+1, \dots, i+s-1]} (score(\hat{T}(i, j-1, c_1)) + score(\hat{T}(j, i+s-1, c_2)) + \log P(c \rightarrow c_1 c_2))$$

- The most probable derivation must consist of the most probable sub derivations.

CKY Decoding

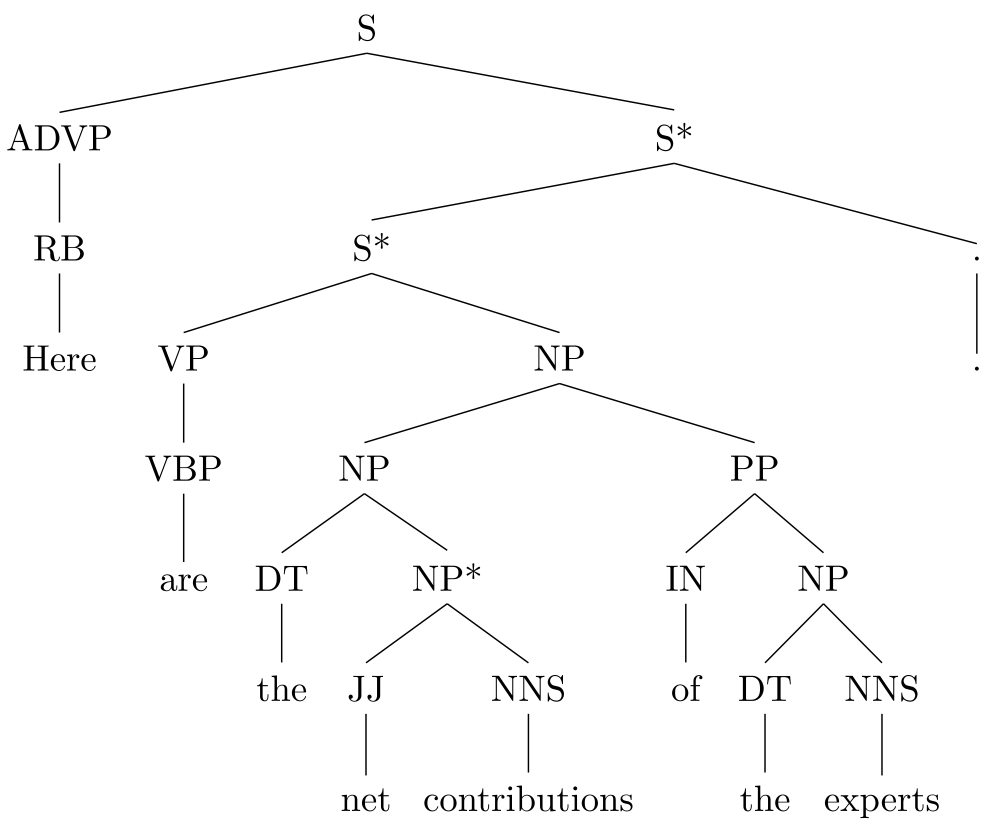
```
Input: W_{1:n} = w_1 w_2 ... w_n, PCFG model P(c → c_1 c_2), P(c → w);
Variables: chart, bp;
Initialisation:
for i ∈ [1, ..., n] do                                        ▷ start index
    for c ∈ C do                                              ▷ constituent label
        chart[1][i][c] ← log P(c → w_i);
for s ∈ [2, ..., n] do                                        ▷ size
    for i ∈ [1, ..., n − s + 1] do                            ▷ start index
        for c ∈ C do                                          ▷ constituent label
            chart[s][i][c] ← −∞;
            bp[s][i][c] ← −1;
Algorithm:
for s ∈ [2, ..., n] do                                        ▷ size
    for i ∈ [1, ..., n − s + 1] do                            ▷ start index
        for j ∈ [i + 1, ..., i + s − 1] do                    ▷ split point
            for c, c_1, c_2 ∈ C do                            ▷ c → c_1 c_2
                score ← chart[j − i][i][c_1] + chart[s − j + i][j][c_2]
                              + log P(c → c_1 c_2);
                if chart[s][i][c] < score then
                    chart[s][i][c] ← score;
                    bp[s][i][c] ← (j, c_1, c_2);
Output: FindDerivation(bp[n][1][arg max_c chart[n][1][c]]);
```

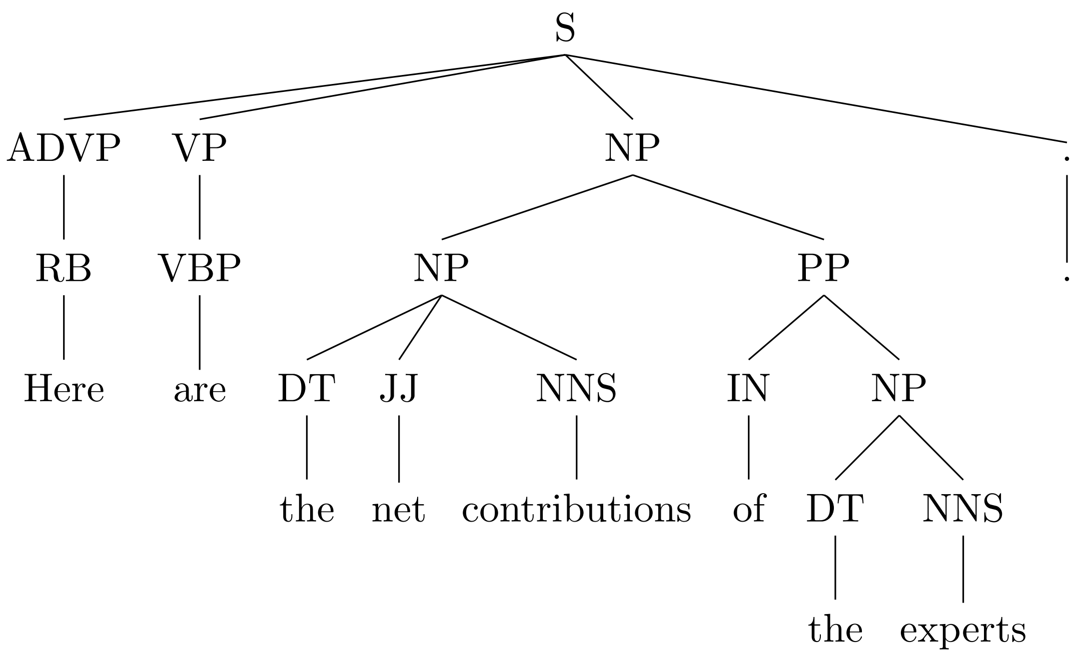
Contents

Evaluating Constituent Parser Outputs

- Evaluating spans (b, e, c)

$(3, 5, NP)$ $(1, 1, RB)$ $(1, 1, ADVP)$





Evaluating Constituent Parser Outputs

- Constituent (b: begin, e: end, c: constituent label)
- Precision
  - the percentage of constituents in the output set that are correct
- Recall
  - the percentage of gold-standard constituents that are identified in the parser output
- F-score
  - 2PR/(P+R)
- Labeled (b, e, c), unlabeled (b, e).

Contents

WestlakeNLP

Calculating Marginal Probabilities

WestlakeNLP

- Given the sentence $W_{1:n} = w_1 w_2 \dots w_n$, the marginal probability for the span $w_i w_{i+1} \dots w_j$ with a constituent label c is:

$$P(Sp(i,j,c)|W_{1:n}) \propto P(Sp(i,j,c), W_{1:n})$$

- In particular,

$$P(Sp(i,j,c), W_{1:n}) = P(S \overset{*}{\Rightarrow} w_1 w_2 \dots w_{i-1} c w_{j+1} w_{j+2} \dots w_n \overset{*}{\Rightarrow} W_{1:n}$$

$$= P\left(S \overset{*}{\Rightarrow} w_1 w_2 \dots w_{i-1} c w_{j+1} w_{j+2} \dots w_n\right) P\left(c \overset{*}{\Rightarrow} w_i w_{i+1} \dots w_j\right)$$

(independence assumption)

$$= \left(\sum_{rules \in\, Gen(W_{1:n}(S)[c \overset{*}{\Rightarrow} w_i \dots w_j])} \prod_{r \in rules} P(r)\right) \cdot \left(\sum_{rules \in Gen\left(w_{i:j}(c)\right)} \prod_{r \in rules} P(r)\right)$$

Calculating Marginal Probabilities

WestlakeNLP

- inside probability：

$$P(c \overset{*}{\Rightarrow} w_i w_{i+1} \dots w_j) = \sum_{rules \in Gen(w_{i:j}(c))} P(c \overset{rules}{\Longrightarrow} w_i w_{i+1} \dots w_j)$$

$$= \sum_{rules \in Gen(w_{i:j}(c))} \prod_{r \in rules} P(r)$$

- outside probability:

$$P(S \overset{*}{\Rightarrow} w_1 w_2 \dots w_{i-1} c w_{j+1} w_{j+2} \dots w_n)$$

$$= \sum_{rules \in Gen(W_{1:n}(S)[c \overset{*}{\Rightarrow} W_{i:j}])} P(S \overset{rules}{\Longrightarrow} w_1 w_2 \dots w_{i-1} c w_{j+1} w_{j+2} \dots w_n)$$

$$= \sum_{rules \in Gen(W_{1:n}(S)[C \overset{*}{\Rightarrow} W_{i:j}])} \prod_{r \in rules} P(r)$$

Calculating Marginal Probabilities

WestlakeNLP

- Both the inside probability and the outside probability can be calculated in polynomial time using dynamic programming.
- Assume our grammar conforms to CNF, and use inside (i, j, c) to denote $P(c \overset{*}{\Rightarrow} w_{i:j})$, then we have:

$$inside(i, j, c)$$

$$= \sum_{k \in [i+1, \ldots, j]} \sum_{c_1, c_2 \in C} inside\,(i, k-1, c_1) \times inside(k, j, c_2) \times P(c \rightarrow c_1 c_2)$$

$$outside(i, j, c)$$

$$= \sum_{k \in [j+1, \ldots, n]} \sum_{c', c_2 \in C} outside\,(i, k, c') \times inside(j+1, k, c_2) \times P(c' \rightarrow c c_2)$$

$$+ \sum_{k \in [1, \ldots, i-1]} \sum_{c', c_2 \in C} outside\,(k, j, c') \times inside(k, i-1, c_2) \times P(c' \rightarrow c_2 c)$$

Inside algorithm

WestlakeNLP

Input: sentence $W_{1:n} = w_1 w_2 \ldots w_n$, PCFG model $P(c \rightarrow c_1 c_2)$, $P(c \rightarrow w)$, start index b, the end index e, the constituent label X;
Variables: $inside$;
Initialisation:

```
for i ∈ [1, ..., n] do
    for j ∈ [i + 1, ..., n] do
        for c ∈ C do
            inside[i][j][c] ← 0;
    for c ∈ C do
        inside[i][i][c] ← P(c → w_i);
```

Algorithm:

```
for s ∈ [2, ..., n] do
    for i ∈ [1, ..., n − s + 1] do
        for j ∈ [i + 1, ..., i + s − 1] do
            for c, c_1, c_2 ∈ C do                          ▷ c → c_1 c_2
                inside[i][i + s − 1][c] ← inside[i][i + s − 1][c]+
                  inside[i][j−1][c_1] ∗ inside[j][i+s−1][c_2] ∗ P(c → c_1 c_2);
```

Output: $inside[b][e][X]$;

Outside algorithm

WestlakeNLP

Input: The sentence $W_{1:n} = w_1 w_2 \dots w_n$, the PCFG model, the start index b, the end index e, the constituent label X;

Variables: $outside$;

Initialisation:

$$
\begin{array}{l}
\textbf{for } i \in [1, \dots, n] \textbf{ do} \\
\quad \textbf{for } j \in [i, \dots, n] \textbf{ do} \\
\quad\quad \textbf{for } c \in C \textbf{ do} \\
\quad\quad\quad outside[i][j][c] \leftarrow 0; \\
outside[1][n][S] \leftarrow 1;
\end{array}
$$

Algorithm:

$$
\begin{array}{l}
\textbf{for } s \in [1, \dots, n] \textbf{ do} \\
\quad \textbf{for } i \in [1, \dots, n-s+1] \textbf{ do} \\
\quad\quad \textbf{for } j \in [i+1, \dots, i+s-1] \textbf{ do} \\
\quad\quad\quad \textbf{for } c, c_1, c_2 \in C \textbf{ do} \qquad \triangleright\ c \rightarrow c_1 c_2 \\
\quad\quad\quad\quad outside[i][j-1][c_1] \leftarrow outside[i][j-1][c_1] + \\
\quad\quad\quad\quad outside[i][i+s-1][c] * inside[j][i+s-1][c_2] * P(c \rightarrow c_1 c_2); \\
\quad\quad\quad\quad outside[j][i+s-1][c_2] \leftarrow outside[j][i+s-1][c_2] + \\
\quad\quad\quad\quad\ outside[i][i+s-1][c] * inside[i][j-1][c_1] * P(c \rightarrow c_1 c_2);
\end{array}
$$

Output: $outside[b][e][X]$;

Contents

More Features for Constituent Parsing

WestlakeNLP

- The feature set of PCFG is rather simple for disambiguation.
- Methods to integrate richer features:

  (1). Extend the generative story of PCFGs.

  (2). Use a discriminative model to accommodate overlapping features.

Contents

WestlakeNLP

Lexicalized PCFGs

WestlakeNLP

- Disambiguation
  - rule1: VP-->VB NP
  - rule2: VP-->VB

  If V is a transitive verb, it is difficult to decide which rule is better.
- Solution

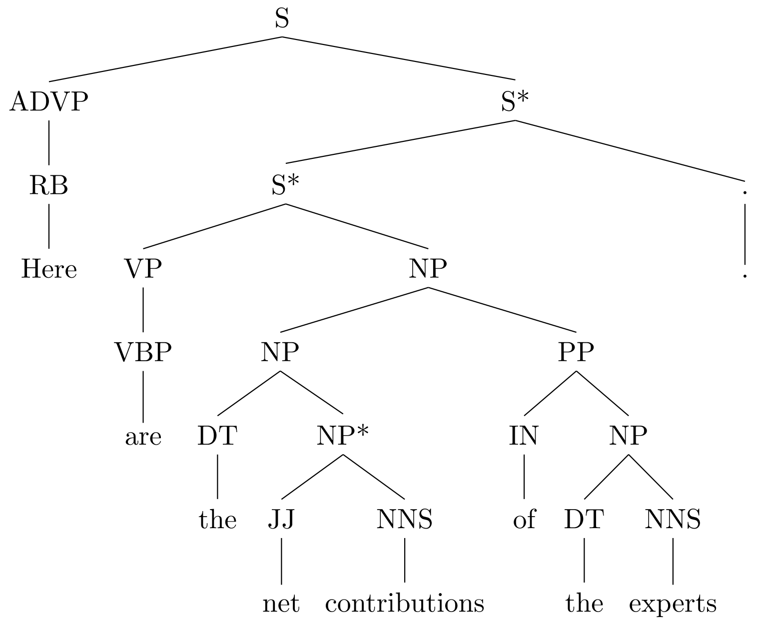
  Enrich PCFG constituent labels with lexical information.

  e.g. VP[eat] --> VB[eat] NP

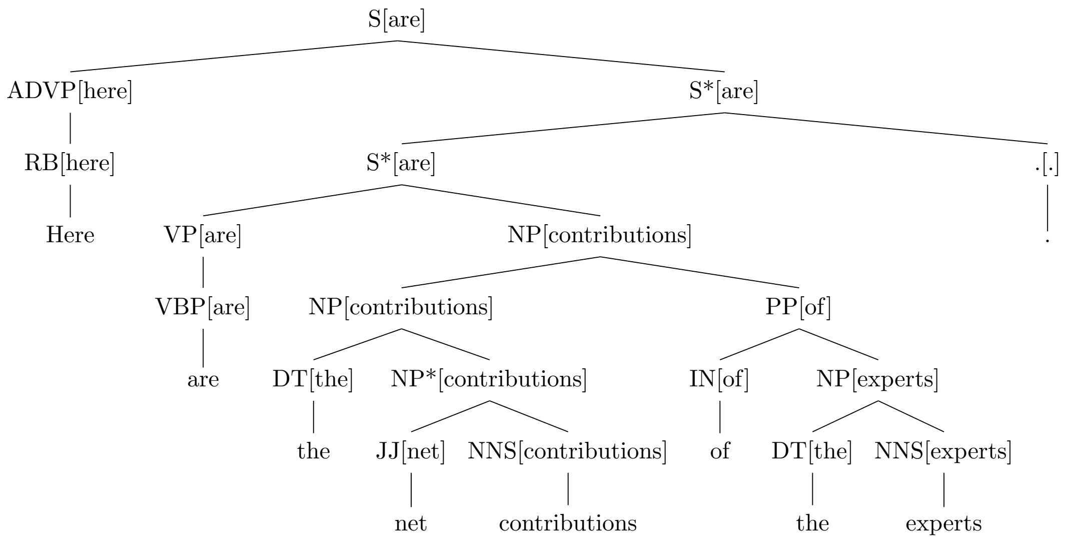
  VP[eat] --> VB[eat] NP[pizza]

Lexicalized PCFGs

- A head-lexicalized constituent tree



Head-binarization



Head-lexicalized version

Lexicalized PCFGs

WestlakeNLP

- Estimation using MLE

$$P(VP[like] \rightarrow VB[like]\ NP) = \frac{count(VP[like] \rightarrow VB[like]\ NP)}{count(VP[like])}$$

- Advantage: easy to disambiguate
- Disadvantage: sparse

  To avoid zero-probability, one can use back-off.

Lexicalized PCFGs

- Decoding
  - Different with original CKY:
  - (1). chart[s][i][c][h]: the probability of the highest scored constituent over the text span $w_i, \ldots w_{i+s-1}$

    i is the start index, s is the span size, c is the constituent label, and h is the head position.
  - (2). complexity: O($n^5|C|^3$)

CKY algorithm for head lexicalized PCFG

```
Input: W_{1:n} = w_1 w_2 ... w_n;
Variables: chart, bp;
Initialisation:
for i ∈ [1,...,n] do                                          ▷ start index
    for c ∈ C do                                              ▷ constituent label
        chart[1][i][c][i] ← log P(c[w_i] → w_i);
for s ∈ [2,...,n] do                                          ▷ size
    for i ∈ [1,...,n−s+1] do                                  ▷ start index
        for h ∈ [i,...,i+s−1] do                              ▷ head
            for c ∈ C do                                      ▷ constituent label
                chart[s][i][c][h] ← −∞;
                bp[s][i][c][h] ← −1;
Algorithm:
for s ∈ [2,...,n] do                                          ▷ size
    for i ∈ [1,...,n−s+1] do                                  ▷ start index
        for j ∈ [i+1,...,i+s−1] do                            ▷ split point
            for h_1 ∈ [i,...,j−1] do                          ▷ left span head
                for h_2 ∈ [j,...,i+s−1] do                    ▷ right span head
                    for h ∈ {h_1, h_2} do                     ▷ head
                        for c, c_1, c_2 ∈ C do                ▷ c → c_1 c_2
                            score ← chart[j−i][i][c_1][h_1] + chart[s−j+
                              i][j][c_2][h_2] + log P(c[h] → c_1[w_{h_1}] c_2[w_{h_2}]);
                            if chart[s][i][c][h] < score then
                                chart[s][i][c][h] ← score;
                                bp[s][i][c][h] ← (j, c_1, c_2, h_1, h_2);
Output: FindDerivation(bp[n][1][arg max_{c,h} chart[n][1][c]]);
```

Contents

WestlakeNLP

Discriminative Linear Models for Constituent Parsing

- Discriminative models map constituent trees for a given sentence into feature vectors.

Given a sentence $W_{1:n}$ and a constituent Tree T , the score is:

$$score(W_{1:n}, T) = \vec{\theta} \cdot \vec{\phi}(W_{1:n}, T)$$

$score(W_{1:n}, T)$: the global feature vector,

$\vec{\phi}$: the model parameter vector.

Discriminative Linear Models for Constituent Parsing

- Feature Factorization
  - The global feature vector can be factorized into local feature components,

$$\vec{\phi}(W_{1:n}, T) = \sum_{r \in T} \vec{\phi}\,(W_{1:n}, r)$$

  - Suppose $T(i, j, c)$ consists of two subtrees $T(i, k, c_1)$ and $T(k+1, j, c_2)$ then

$$Score(T(i,j,c)) = Score(T(i,k,c_1)) + Score(k+1,j,c_2) + \vec{\theta}\vec{\phi}(W_{1:n}, c \rightarrow c_1 c_2)$$

Discriminative Linear Models for Constituent Parsing

WestlakeNLP

- Find the tree with the highest score by dynamic program.

$$score\left(\hat{T}(i,j,h,c)\right)$$

$$= argmax_{k\in[i,\dots,j-1],h_1\in[i,\dots,k],h_2\in[k+1,\dots,j],c_1,c_2\in C}$$

$$\left(score\left(\hat{T}(i,k,h_1,c_1)\right) + score\left(\hat{T}(k+1,j,h_2,c_2)\right)\right.$$

$$\left. + \vec{\theta}\cdot\vec{\phi}(W_{1:n}, c[w_h] \rightarrow c_1[w_{h_1}]c_2[w_{h_2}])\right)$$

Decoding with CKY

Input: $W_{1:n} = w_1 w_2 \ldots w_n$, $\vec{\theta}$ — model parameters;
Variables: *chart*, *bp*;
Initialisation:

for $i \in [1, \ldots, n]$ **do** ▷ start index
  for $c \in C$ **do** ▷ constituent label
    $chart[1][i][c][i] \leftarrow \vec{\theta} \cdot \phi(W_{1:n}, c[w_i] \rightarrow w_i)$;

for $s \in [2, \ldots, n]$ **do** ▷ size
  for $i \in [1, \ldots, n-s+1]$ **do** ▷ start index
    for $h \in [i, \ldots, i+s-1]$ **do** ▷ head
      for $c \in C$ **do** ▷ constituent label
        $chart[s][i][c][h] \leftarrow -\infty$;
        $bp[s][i][c][h] \leftarrow -1$;

Algorithm:

for $s \in [2, \ldots, n]$ **do** ▷ size
  for $i \in [1, \ldots, n-s+1]$ **do** ▷ start index
    for $j \in [i+1, \ldots, i+s-1]$ **do** ▷ split point
      for $h_1 \in [i, \ldots, j-1]$ **do** ▷ left head
        for $h_2 \in [j, \ldots, i+s-1]$ **do** ▷ right head
          for $h \in \{h_1, h_2\}$ **do** ▷ head
            for $c, c_1, c_2 \in C$ **do** ▷ $c \rightarrow c_1 c_2$
              $score \leftarrow chart[j-i][i][c_1][h_1]$
              $+chart[s-j+i][j][c_2][h_2] + \vec{\theta} \cdot \vec{\phi}(W_{1:n}, c[w_h] \rightarrow$
              $c_1[w_{h_1}] c_2[w_{h_2}])$;
              if $chart[s][i][c][h] < score$ **then**
                $chart[s][i][c][h] \leftarrow score$;
                $bp[s][i][c][h] \leftarrow (j, c_1, c_2, h_1, h_2)$;

Output: $\textsc{FindDerivation}\Big(bp[n][1][\arg\max_{c,h} chart[n][1][c][h]]\Big)$;

Contents

WestlakeNLP

Training Log-linear Models for Constituent Parsing

WestlakeNLP

- The conditional probability of T for a sentence is :

$$P(T|W_{1:n}) = \frac{\exp(\vec{\theta}\cdot\vec{\phi}(W_{1:n},T))}{\sum_{T' \in \, Gen(W_{1:n})} \exp(\vec{\theta}\cdot\vec{\phi}(W_{1:n},T'))}$$

- Given the training set D, the training objective is to maximize the log-likelihood of D:

$$\begin{aligned}
\hat{\vec{\theta}} &= argmax_{\vec{\theta}} \log P(D) \\
&= argmax_{\vec{\theta}} \log \prod_i P\,(T_i|W_i) \quad (i.i.d.) \\
&= argmax_{\vec{\theta}} \sum_i \log \frac{\exp(\vec{\theta}\cdot\vec{\phi}(W_i,T_i))}{\sum_{T' \in Gen(W_i)} \exp\,(\vec{\theta}\cdot\vec{\phi}(W_i,T'))} \\
&= argmax_{\vec{\theta}} \sum_i (\vec{\theta}\cdot\vec{\phi}(W_i,T_i) - \log \sum_{T' \in Gen(W_i)} \exp\,(\vec{\theta}\cdot\vec{\phi}(W_i,T')))
\end{aligned}$$

Training Log-linear Models for Constituent Parsing

WestlakeNLP

- Due to feature factorization, we have

$$\sum_{r \in GenR(W_i)} E_{T' \sim P(T'|W_i)} \left(\vec{\phi}(W_i, r) \cdot 1(r \in T')\right) = \sum_{r \in GenR(W_i)} E_{r \sim P(r|W_i)} \vec{\phi}(W_i, r)$$

- If we can calculate the marginal probabilities $P(T'|W_i)$, then the expectation of $\vec{\phi}(W_i, r)$ over all possible T′ is equivalent to the expectation over all possible r given W_i .

Training Log-linear Models for Constituent Parsing

- Calculating marginal rule probabilities

$$P(r|W_{1:n}) = \sum_{T \in \, Gen(W_{1:n}) s.t. r \in T} P\,(T|W_{1:n})$$

$$= \sum_{T \in \, Gen(W_{1:n}) s.t. r \in T} \prod_{r' \in T} \exp\,(\vec{\theta} \cdot \vec{\phi}(W_{1:n}, r'))$$

- Divide the marginal probability into three parts which can be calculated by modified CKY algorithm.

$$P(r|W_{1:n}) =$$

$$InsideScore(b, b' - 1, c_1, h_1, W_{1:n}) InsideScore(b', e, c_2, h_2, W_{1:n})$$

$$Outside\,(b, e, c, h, W_{1:n}) \exp(\vec{\theta} \cdot \vec{\phi}(W_{1:n}, r))$$

Contents

Training Log-linear Models for Constituent Parsing

- Training goal

  For both structured perceptron and SVM models, the training goal is to ensure a outside score margin between gold-standard outputs and incorrect outputs.

Training Log-linear Models for Constituent Parsing

WestlakeNLP

- Given a set of training data $D = \{(W_i, T_i)\}|_{i=1}^{N}$,
  - The training objective of structured perceptron is to minimize：

$$\sum_{i=1}^{N} \max\left(0, \max_{T' \in Gen(W_i)}\left(\vec{\theta} \cdot \vec{\phi}(W_i, T')\right) - \vec{\theta} \cdot \vec{\phi}(W_i, T_i)\right)$$

  - The training objective of structured perceptron is to minimize:

$$\frac{1}{2}||\vec{\theta}||^2 + C(\sum_{i=1}^{N} \max\left(0,1 - \vec{\theta} \cdot \vec{\phi}(W_i, T_i) + \max_{T' \neq T_i}(\vec{\theta} \cdot \vec{\phi}(W_i, T'))\right)\right)$$

$where\ T' \in Gen(w_i).$

Contents

WestlakeNLP

Reranking

- Non-local features
  - useful for disambiguation
  - but add time complexity of decoding

- Reranking
  - integrate non-local features
  - without additional asymptotic complexity

Reranking

- Steps of reranking

  (1). obtain output candidates from a base parser with local features.

  (1.1). obtain a fixed number of k-best candidates

  (1.2). obtain a set of candidates that score higher than a threshold

  (2). rescore the set of candidates, considering the score output by base model and non-local features.

Reranking

WestlakeNLP

- Testing

Input: the set of sentences $D = \{(\overline{W}_i, TS_i)\}|_{i=1}^{N}$

the set of n_i-best candidates $TS_i = \{T_i^1, T_i^2, \dots, T_i^{n_i}\}$

then the score of T_i^j given by the reranker is:

$$score(T_i^j) = \vec{\theta} \cdot \vec{\phi}(W_i, T_i^j)$$

$$\vec{\phi}(W_i, T_i^j) = \langle base\_score(T_i^j), f_1(W_i, T_i^j), f_2(W_i, T_i^j), \dots, f_m(W_i, T_i^j)$$

where $f_k(W_i, T_i^j)$ $(k \in [1, \dots, m])$ are non-local features,

and $base\_score(T_i^j)$ is the score given by the base parser.

Reranking

- Training a reranking model using log-likelihood loss
  - Training data: $D = \{(W_i, \{T_i\} \cup TS_i)\}|_{i=1}^{N}$
  - The training objective is:

$$\log P(D) = \sum_i \log P(T_i|W_i)$$

$$= \sum_i \left((score(T_i)) - \log\left(e^{score(T_i)} + \sum_{j=1}^{n_i} e^{score(T_i^j)}\right)\right)$$

Reranking

- Training a reranking model using log-likelihood loss
  - SGD can be used for optimization, then the local gradient is:

$$\frac{\partial}{\partial\vec{\theta}}\log P(T_i|W_i) = \vec{\phi}(W_i,T_i) - (P(T_i|W_i)\cdot\vec{\phi}(W_i,T_i) + \sum_{j=1}^{n_i} P\left(T_i^j|W_i\right)\cdot\vec{\phi}\left(W_i,T_i^j\right))$$

$$= (1 - P(T_i|W_i))\cdot\vec{\phi}(W_i,T_i) + \sum_{j=1}^{n_i} P\left(T_i^j|W_i\right)\cdot\vec{\phi}(W_i,T_i^j)$$

Reranking

- Training a reranking model using large-margin loss
  - The objective function is to minimize the score margin:

$$\max(0, 1 + \max_{T' \in TS}(\vec{\theta} \cdot \vec{\phi}(W_i, T')) - \vec{\theta} \cdot \vec{\phi}(W_i, T_i))$$

  - SGD can be used for optimization, then the local gradient is:

$$\begin{cases} 0 & \text{if } \vec{\theta} \cdot \vec{\phi}(W_i, T_i) > \max_{T' \in TS_i} \vec{\theta} \cdot \vec{\phi}(W_i, T') \\ \vec{\phi}(W_i, T_i) - \max_{T' \in TS_i} \vec{\phi}(W_i, T') & \text{otherwise} \end{cases}$$

Contents

WestlakeNLP

Beyond Sequences and Trees

- Common underlying modeling techniques
  - probability chain rule, independency assumption
  - Bayes rule
  - dynamic programs
- Dynamic programs and feature constraints
  - Efficiency
  - Accuracy
- More alternatives to discuss

Summary

- Generative constituent parsing, binarization, probabilistic context free grammars (PCFGs)
- CKY algorithm, inside-outside algorithm, lexicalized PCFGs
- Log-linear models for discriminative constituent parsing
  Large-margin models for discriminative constituent parsing
- Reranking