1 Definitions

The definition of a context-free grammar is extended to define a probabilistic context-free grammar (PCFG).

Definition 1. A PCFG is defined as $\mathbf{G} = (\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S, p)$, where $\mathbf{V}$ is an alphabet (a set of symbols), $\mathbf{\Sigma} \subset \mathbf{V}$ is the set of all terminal symbols of $\mathbf{V}$, $S \in \mathbf{V} - \mathbf{\Sigma}$ is the designated start symbol, $\mathbf{R} \subseteq (\mathbf{V} - \mathbf{\Sigma}) \times \mathbf{V}^*$ is a set of rules, and $p : \mathbf{R} \to \mathbb{R}$ is a function that satisfies the following two properties:

$$\begin{aligned} p(r) &\geq 0, \forall r \in \mathbf{R} \\ \sum_{r \in \mathbf{R}(A)} p(r) &= 1, \forall A \in \mathbf{V} - \mathbf{\Sigma} \end{aligned} \tag{1}$$

Function p is called the probability function of the PCFG, and $p(r)$ is the probability of rule r. In essence, each non-terminal symbol can be viewed as an experiment; each rule that replaces the non-terminal symbol is a possible outcome of the experiment. The probability function p assigns probabilities to each outcome of each experiment. When using the rules to produce a string, each substitution of a non-terminal symbol can be viewed as a trial of the experiment corresponding to that non-terminal symbol.

2 Chomsky normal form for PCFGs: Algorithm

This section presents the algorithm for transforming a probabilistic context-free grammar $\mathbf{G} = \{\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S, p\}$ into an equivalent in Chomsky normal form. It consists of the same three stages; the first stage removes long rules, the second stage removes e-rules, and the third stage removes short rules. However, in each stage, the probability function p should be modified appropriately, so that $\Pr\{s\}$ remains the same for every $s \in \mathbf{\Sigma}^*$.

The first stage deals with long rules. Algorithm 2.1 describes this process. Note that after each iteration, $\mathbf{G}$ remains indeed a PCFG, meaning that function p satisfies both properties. After this algorithm is applied, $\mathbf{G}$ contains rules of length 1, length 2, and e-rules.

Algorithm 2.1: Removing long rules from a PCFG

```
Input  : {V, Σ, R, S}
Output: {V, Σ, R, S} with no long rules
for r ∈ R : r = A → B_1 B_2 ... B_n, n > 2 do
    R = R − {r};
    V = V ∪ {C_i^r : i = 1, ..., n − 2};
    R = R ∪ {A → B_1 C_1^r};
    Set p(A → B_1 C_1^r) = p(r);
    R = R ∪ {C_i^r → B_{i+1} C_{i+1}^r : i = 1, ..., n − 3};
    Set p(C_i^r → B_{i+1} C_{i+1}^r) = 1, i = 1, ..., n − 3;
    R = R ∪ {C_{n−2}^r → B_{n−1} B_n};
    Set p(C_{n−2}^r → B_{n−1} C_n^r) = 1;
```

The second stage removes all e-rules. Computing the set $\mathbf{E}$ is required once again. However, we for each erasable symbol we need to compute the probability of the symbol being erased. In particular, let $A \in \mathbf{E}$; since $\mathbf{G}$ does not contain long rules any more we can obtain

$$\begin{aligned} \Pr\{e|A\} = &\Pr\{A \Rightarrow e\} \\ &+ \sum_{B \in \mathbf{V}} \Pr\{A \Rightarrow B\} \Pr\{e|B\} \\ &+ \sum_{B,C \in \mathbf{V}} \Pr\{A \Rightarrow BC\} \Pr\{e|B\} \Pr\{e|C\} \end{aligned} \tag{2}$$

Unfortunately this recursive formula is not suitable for being programmed, as infinite recursions can occur. However, $\Pr\{A \Rightarrow e\}$ can be obtained directly from $\mathbf{G}$ and the two summations need only be computed over the rules of $\mathbf{R}_1(A)$ and $\mathbf{R}_2(A)$ respectively. Additionally, the first factor of each product of the summations (namely $\Pr\{A \Rightarrow B\}$ and $\Pr\{A \Rightarrow BC\}$) can also be obtained directly from $\mathbf{G}$. Thus, for a given set $\mathbf{E}$, we need to compute $l = \|\mathbf{E}\|$ probabilities, particular $\Pr\{e|A\}, \forall A \in \mathbf{E}$. If we treat these probabilities as unknowns, and apply Equation 2 for each one, we can create a system of l equations and l unknowns; however, each such equation is of quadratic form.

To formulate the equations we first need to enumerate the symbols of $\mathbf{E}$. The enumeration can be arbitrary; in the following we assume that we have selected one enumeration and we will refer to the i-th element of $\mathbf{E}$ in this enumeration as $(\mathbf{E})_i$. The system is given in matrix notation in the following equation

$$\mathsf{x} = \left(\mathsf{I}_l \otimes \mathsf{x}^T\right) \begin{bmatrix} \mathsf{A}_1 \\ \vdots \\ \mathsf{A}_l \end{bmatrix} \mathsf{x} + \mathsf{B}'\mathsf{x} + \mathsf{c} \tag{3}$$

where x is the $l \times 1$ vector of unknowns, A_i, B are $l \times l$ matrices ($i = 1, \dots, l$), c is an $l \times 1$ vector, I_l is the $l \times l$ identity matrix, and $\otimes$ is the Kronecker product. Furthermore, $\mathsf{x}[i] = \Pr\{e|(\mathbf{E})_i\}$, $\mathsf{A}_i[j,k] = \Pr\{(\mathbf{E})_i \Rightarrow (\mathbf{E})_j(\mathbf{E})_k\}$, $\mathsf{B}'[i,j] = \Pr\{(\mathbf{E})_i \Rightarrow (\mathbf{E})_j\}$ and $\mathsf{c}[i] = \Pr\{(\mathbf{E})_i \Rightarrow e\}$. All values of matrices $\mathsf{A}_i, \mathsf{B}', \mathsf{c}$ are obtained directly from $\mathbf{G}$, and $\mathsf{B}'[i,i] = 0$ since no rules of the form $A \to A$ are allowed.

By setting $\mathsf{B} = \mathsf{B}' - \mathsf{I}_l$, we can define the function $f : [0,1]^l \mapsto \mathbb{R}^l$

$$f(\mathsf{x}) = \left(\mathsf{I}_l \otimes \mathsf{x}^T\right) \begin{bmatrix} \mathsf{A}_1 \\ \vdots \\ \mathsf{A}_l \end{bmatrix} \mathsf{x} + \mathsf{B}\mathsf{x} + \mathsf{c} \tag{4}$$

Also note that the Jacobian of f is easily obtained, and equal to

$$\mathcal{D}f(x) = \left(\mathsf{I}_l \otimes \mathsf{x}^T\right) \begin{bmatrix} \mathsf{A}_1 + \mathsf{A}_1^T \\ \vdots \\ \mathsf{A}_l + \mathsf{A}_l^T \end{bmatrix} + \mathsf{B} \tag{5}$$

We can now re-write Equation 3 as

$$f(\mathsf{x}) = 0_l \tag{6}$$

where 0_l is the $l \times 1$ all-zeros vector. We can obtain the solution of this equation by solving the following optimisation problem

$$\min_{\mathsf{x}} f(\mathsf{x})^T f(\mathsf{x}) \tag{7}$$

This is a least-squares problem. In essence, we have to minimise the level-2 norm of f; furthermore, f is a convex function with a known Jacobian. We can thus obtain the solution easily, using any method that solves such problems, such as the 'Gauss-Newton' or the 'Levenberg-Marquardt' methods.

Once we have solved this optimisation problem and obtained the probabilities of erasing for all symbols of $\mathbf{E}$, we can remove all e-rules from the PCFG using the Algorithm 2.2.

Finally, we need to remove the short rules that remain in $\mathbf{G}$. As with CFGs, we also need to compute the sets $\mathbf{D}(A), \forall A \in \mathbf{V}$. However, for each $\mathbf{D}(A), A \in (\mathbf{V} - \mathbf{\Sigma})$ we also need to compute the probabilities $\Pr\{B|A\}, \forall B \in \mathbf{D}(A)$. In order to computer these probabilities we first define the set $\mathbf{D}'(A)$, which contains all symbols of $\mathbf{V} - \mathbf{\Sigma}$ that can produce A

$$\mathbf{D}'(A) \triangleq \{B \in \mathbf{V} - \mathbf{\Sigma} : B \Rightarrow^* A\} \tag{8}$$

This set can be computed either in a similar fashion as $\mathbf{E}$ (note that in fact $\mathbf{E} = \mathbf{D}'(e)$) or directly from the sets $\mathbf{D}(A), \forall A \in \mathbf{V}$. Note that for the first method of computing $\mathbf{D}'(A)$ we only need to take into account short rules, since no e-rules are currently present in $\mathbf{R}$. We can now write

$$\Pr\{B|A\} = \Pr(A \Rightarrow B) + \sum_{C \in \mathbf{V} - \mathbf{\Sigma}} \Pr\{A \to C\} \Pr\{C \Rightarrow^* B\} \tag{9}$$

Algorithm 2.2: Removing e-rules from a CFG

Input : $\{\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S\}$ with no long rules, $\mathbf{E}$, $\Pr(e|A) \forall A \in \mathbf{E}$
Output: $\{\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S\}$ with no long rules and no e-rules
$\mathbf{R} = \mathbf{R} - \{A \to e\}$;
for $(A \to BC) \in \mathbf{R}$ **do**
  if $B \in \mathbf{E}$ **then**
    $\mathbf{R} = \mathbf{R} \cup \{A \to C\}$;
    Set $p(A \to C) = p(A \to BC) \Pr\{e|B\}$;
  if $C \in \mathbf{E}$ **then**
    $\mathbf{R} = \mathbf{R} \cup \{A \to B\}$;
    Set $p(A \to B) = p(A \to BC) \Pr\{e|C\}$;

Given a set $\mathbf{D}'(A)$ we can compute all probabilities of the form $\Pr\{A|B\}, B \in \mathbf{D}'(A)$. In particular, similarly to e-rules, we can treat $\Pr\{A|B\}$ as unknowns and create $l = \|\mathbf{D}'(A)\|$ equations with l unknowns. Assuming an arbitrary enumeration of $\mathbf{D}'(A)$, we can write

$$\mathsf{x} = \mathsf{A}'\mathsf{x} + \mathsf{b} \tag{10}$$

where x is the $l \times 1$ vector of unknowns, A' is a $l \times l$ matrix, and b is a $l \times 1$ vector. Furthermore, $\mathsf{A}'[i,j] = \Pr\{(\mathbf{D}(A))_i \Rightarrow (\mathbf{D}(A))_j\}$ and $\mathsf{b}[i] = \Pr\{A \Rightarrow (\mathbf{D}(A))_i\}$. All values of A' and b are obtained directly from $\mathbf{G}$, and $\mathsf{A}'[i][j] = 0$ since no rules of the form $A \to A$ are allowed. By setting $\mathsf{A} = \mathsf{A}' - I_l$ we can re-write Equation 10 as

$$\mathsf{A}\mathsf{x} = -\mathsf{b} \tag{11}$$

which is directly solvable. An important note is that one of the unknowns is $\Pr\{A|A\}$. The interpretation of this unknown as a probability is not correct. However, this unknown is only required to solve 11. We can now proceed with the final stage of the algorithm, shown in Algorithm 2.3.

Algorithm 2.3: Removing short rules from a PCFG

Input : $\{\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S\}$ with no long rules and no e-rules
Output: $\{\mathbf{V}, \mathbf{\Sigma}, \mathbf{R}, S\}$ in Chomsky normal form
$\mathbf{R} = \mathbf{R} - \{A \to B : A, B \in \mathbf{V}\}$;
for $(A \to BC) \in \mathbf{R}$ **do**
  $\mathbf{R} = \mathbf{R} \cup \{A \to B'C' : B' \in \mathbf{D}(B) - \{B\}, C' \in \mathbf{D}(C) - \{C\}\}$;
  Set $p(A \to B'C') = p(A \to BC) \Pr\{B \Rightarrow B'\} \Pr\{C \Rightarrow C'\}$;
for $(A \to BC) \in \mathbf{R} : A \in \mathbf{D}(S)$ **do**
  $\mathbf{R} = \mathbf{R} \cup \{S \to BC\}$;
  Set $p(S \to BC) = p(S \to A) p(A \to BC)$;