# Natural Language Processing: Assignment 4

Isaac Jefferson Lee

Email: isalee@student.ethz.ch

## Question 1

### Question 1 :: Part a)

First we prove that the sum of probabilities for strings of any fixed length is 1.

Clearly length 1 strings are just $\Sigma$ and their probabilities sum to 1.

Now for the inductive step, assume:

$$\sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = k} \tilde{p}(\boldsymbol{w}) = 1.$$

Then it follows that:

$$\sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = k+1} \tilde{p}(\boldsymbol{w}) = \sum_{w \in \Sigma} p(w) \left[ \sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = k} \tilde{p}(\boldsymbol{w}) \right] = \sum_{w \in \Sigma} p(w)[1] = 1.$$

And so we have proven that:

$$\sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = k = n} \tilde{p}(\boldsymbol{w}) = 1 \quad \forall n \in \mathbb{N}.$$

Since we have countably infinite number of strings in $\Sigma^*$, then it follows that:

$$\sum_{\boldsymbol{w} \in \Sigma^*} \tilde{p}(\boldsymbol{w}) = \sum_{n=0}^{\infty} \sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = n} \tilde{p}(\boldsymbol{w}) = \sum_{n=0}^{\infty} 1 \to \infty.$$

QED.

### Question 1 :: Part b)

Consider the strings of length $n$. By induction we can show that:

$$\sum_{\boldsymbol{w} \in \Sigma^* : |\boldsymbol{w}| = n} p(\boldsymbol{w}) = p(\text{EOS})(1 - p(\text{EOS}))^n.$$

$$\implies \sum_{\boldsymbol{w}\in\Sigma^*} p(\boldsymbol{w}) = \sum_{n=0}^{\infty} \sum_{\boldsymbol{w}\in\Sigma^*:|\boldsymbol{w}|=n} p(\boldsymbol{w})$$
$$= \sum_{n=0}^{\infty} p(\text{EOS})(1-p(\text{EOS}))^n$$
$$= p(\text{EOS}) \sum_{n=0}^{\infty} (1-p(\text{EOS}))^n$$
$$= p(\text{EOS}) \frac{1}{1-(1-p(\text{EOS}))} = \frac{p(\text{EOS})}{p(\text{EOS})} = 1$$

QED.

## Question 1 :: Part c)

$$\sum_{\boldsymbol{u}\in\Sigma^*} p(\boldsymbol{w}\boldsymbol{u}) = p_{\text{pre}}(\boldsymbol{w}) \underbrace{\sum_{\boldsymbol{u}\in\Sigma^*} p(\text{EOS}|\boldsymbol{w}\boldsymbol{u}) p(u_1|\boldsymbol{w}) \prod_{n=2}^{M} p(u_n|\boldsymbol{w}, u_1, ..., u_{n-1})}_{(\dagger)}.$$

(†) is the pathsum of the parse-sub-tree that has root at $w_N|w_0, ..., w_{N-1}$ and clearly by local normalization property with the argument in b) extended to the $n-$ gram case we see that (†) must sum to 1.

$$\implies \sum_{\boldsymbol{u}\in\Sigma^*} p(\boldsymbol{w}\boldsymbol{u}) = p_{\text{pre}}(\boldsymbol{w}).$$

QED.

## Question 1 :: Part d)

In order to calculate the probability of a sentence, we need the inside probability given by:

$$p_{\text{inside}}(\boldsymbol{w}|S) = \sum_{\boldsymbol{t}\in\mathcal{T}_S(\boldsymbol{w})} p(\boldsymbol{t}).$$

This quantity looks very similar to the normalizer which is calculated with normal CKY. So we can compute this using CKY where we replace exp(score(•)) with $p(\bullet)$, where $p$ is specified by the PCFG.

The pseudocode presented below is adapted from the course notes and should calculate the correct probability according to the above logic.

**Algorithm 1** CKY for Probability

```
def prob_cky(w, ⟨N, Σ, S, R, p⟩) :
N ← |w|
chart ← 0
for n = 1, ..., N do                          ▷ Fill the diagonal with leaf probabilities
    for X → w_n ∈ R do
        chart[n, n + 1, X] += p(X → w_n)
    end for
end for
for span = 2, ..., N do                                  ▷ Fill the rest of the chart
    for i = 1, ..., N − span + 1 do                      ▷ i is the start idx of the span
        k ← i + span                                     ▷ k is the end idx of the span
        for j = i + 1, ..., k − 1 do                     ▷ j is the breaking point of the span
            for X → Y Z ∈ R do
                chart[i, k, X] += p(X → Y Z) ∗ chart[i, j, Y] ∗ chart[j, k, Z]
            end for
        end for
    end for
end for
return chart[1, N + 1, S]
```

**Note:** In the pseudocode the variables given by chart[$i$, $j$, $Y$] represent the inside probabilities $p_{\text{inside}}(w_i, w_{i+1}, ..., w_j|Y)$, so effectively we are working bottom up and calculating the inside probabilities for each fixed span length and then increasing the span length and repeating until we span the entire sentence.

## Question 1 :: Part e)

Using our language model probability definition, if $\boldsymbol{s} \in \Sigma^*$, then:

$$p(\boldsymbol{s}) := p(\text{EOS}|s_0, ..., s_N) \prod_{n=1}^{N} p(s_n|s_0, ..., s_{n-1}) = p(s_0, s_1, ..., s_N, \text{EOS}).$$

Clearly under our PCFG this is equal to the probability that we yield each $s_i$:

$$= \sum_{\boldsymbol{t} \in \mathcal{T}_{\boldsymbol{S}}(\boldsymbol{s})} \underbrace{\prod_{X \to YZ \in \mathcal{T}_S(\boldsymbol{s})} p(X \to YZ)}_{\text{Non-terminals}} \underbrace{\prod_{X \to s_n \in \mathcal{T}_S(\boldsymbol{s})} p(X \to s_n)}_{\text{Terminals}} = \sum_{\boldsymbol{t} \in \mathcal{T}_S(\boldsymbol{s})} p(\boldsymbol{t}) =: p(S \overset{*}{\Longrightarrow} \boldsymbol{s}).$$

So now define $\boldsymbol{s} := \boldsymbol{wu}$ and we have:

$$p(\boldsymbol{wu}) = p(S \overset{*}{\Longrightarrow} \boldsymbol{wu}).$$

$$\Longrightarrow \sum_{\boldsymbol{u} \in \Sigma*} p(\boldsymbol{wu}) = \sum_{\boldsymbol{u} \in \Sigma*} p(S \overset{*}{\Longrightarrow} \boldsymbol{wu}).$$

## Question 1 :: Part f)

(If not in CNF, first convert to CNF).

Suppose we have $m$ non-terminals in our PCFG, $\{X_1, X_2, ..., X_m\}$, then define the matrix $A \in \mathbb{R}^{m \times m}$ by:

$$A_{ij} = p(X_i \to X_j \boldsymbol{\alpha}) = \sum_{k=1}^{m} \underbrace{p(X_i \to X_j X_k)}_{\text{chart[i,j,k]}}.$$

We can think of this matrix as the adjacency matrix for the left corner parse tree with weights given by the PCFG production probabilities. In assignment three we showed that for an adjacency matrix $A$, $(A^n)_{ij}$ encodes the sum of paths from $i$ to $j$. In this context this means that $(A^n)_{ij}$ represents the probability of getting from $X_i$ to $X_j$ along the left corner after exactly n derivations. Therefore if we find the Kleene Star then this will encode the probability of getting from $X_i$ to $X_j$ along the left corner for all possible $n \in \mathbb{N}$.

Recall for matrices, $A^* = (I - A)^{-1}$. So therefore we have:

$$p_{lc}(X_j|X_i) = A^*_{ij} = (I - A)^{-1}_{ij}.$$

Then to calculate $p_{lc}(X_j X_k|X_i)$ we can find the plc of $X'$ s.t $X' \to X_j X_k$ and then we note that:

$$p_{lc}(X_j X_k|X_i) = \sum_{X' \to X_j X_k} p_{lc}(X'|X_i) p(X' \to X_j X_k).$$

So using these steps outlined, our algorithm will look like:

**Algorithm 2** Left Corner Probability

```
def prob_left_corner(w, ⟨N, Σ, S, R, p⟩) :
m ← |N|
A ← 0 ∈ R^{m×m}
for i = 1, ..., m do
    for j = 1, ..., m do
        for k = 1, ..., m do
            A[i, j] += p(X_i → X_j X_k)
        end for
    end for
end for
A* ← (I − A)^{-1}                          ▷ This gives p_lc(X_j|X_i) ∀i, j
triplets ← 0 ∈ R^{m×m×m}                   ▷ triplets[i, j, k] := p_lc(X_j, X_k|X_i)
for i = 1, ..., m do
    for j = 1, ..., m do
        for k = 1, ..., m do
            for X_r → X_j X_k ∈ R do
                triplets[i, j, k] += A*[i, r] ∗ p(X_r → X_j X_k)
            end for
        end for
    end for
end for
return A*, triplets
```

**Note:** The first set of nested for loops clearly has $O(|\mathcal{N}|^3)$ complexity. Then the matrix inversion can also be done in $O(|\mathcal{N}|^3)$ time. Then for the next nested for loop, when we do for$X_r \to X_j X_k \in \mathcal{R}$ for fixed $j, k$, clearly worst case scenario we could have $|\mathcal{N}|$ production rules of this form. It follows that these nested for loops have runtime $O(|\mathcal{N}|^4)$, and therefore the overall runtime complexity of our algorithm is $O(|\mathcal{N}|^4)$.

## Question 1 :: Part g)

In order to compute the prefix probability of a string, we need the probability of producing the string with an arbitrary suffix. We therefore need to sum over al ways we can do this, considering all possible parent non-terminals. This means we must sum over all non-terminal sibling pair left corner probabilities. Then for each $Y, Z$ sibling pair, we consider the event that the subtree rooted at $Y$ yields the first part of the string and then the subtree rooted at $Z$ yields the remainder of the string, along with the arb. suffix. We can do this for all possible ways we can split the string into two parts. Mathematically this looks like:

$$\begin{aligned} p_{\text{pre}}(w_i...w_k|X) &= \sum_{\boldsymbol{u}\in\Sigma^*}\sum_{j=i}^{k-1}\sum_{Y,Z\in\mathcal{N}} p(X \overset{*}{\Longrightarrow} YZ\boldsymbol{\alpha})p(Y \overset{*}{\Longrightarrow} w_i...w_j)p(Z \overset{*}{\Longrightarrow} w_{j+1}...w_k\boldsymbol{u}) \\ &= \sum_{j=i}^{k-1}\sum_{Y,Z\in\mathcal{N}} p(X \overset{*}{\Longrightarrow} YZ\boldsymbol{\alpha})p(Y \overset{*}{\Longrightarrow} w_i...w_j)\left(\sum_{\boldsymbol{u}\in\Sigma^*} p(Z \overset{*}{\Longrightarrow} w_{j+1}...w_k\boldsymbol{u})\right) \\ &= \sum_{j=i}^{k-1}\sum_{Y,Z\in\mathcal{N}} p_{lc}(Y\ Z|X)p_{\text{inside}}(w_i...w_j)p_{\text{pre}}(w_{j+1}...w_k|Z) \end{aligned}$$

QED.

## Question 1 :: Part h)

Using our algorithm developed in part f) we can compute $p_{lc}(Y\ Z|X)$ and $p_{lc}(Y|X)\ \forall Y, Z, X \in \mathcal{N}$ in $O(|\mathcal{N}|^4)$ time. Then to get $p_{\text{pre}}(w_k|X)$ we note that:

$$\forall X \in \mathcal{N} p_{\text{pre}}(w_k|X) = p(X \to w_k) + \sum_{Y\to w_k\in\mathcal{R}} p_{lc}(Y|X)p(Y \to w_k).$$

So clearly pre-computing $p_{lc}(Y\ Z|X)$ and $p_{\text{pre}}(w_k|X)\ \forall Y, Z, X \in \mathcal{N},\ \forall k \in \{1, ..., N\}$ will have a worst case runtime complexity $O(|\mathcal{N}|^4 + N|\mathcal{N}|^2)$.

So we have $\forall X \in \mathcal{N}, p_{\text{pre}}(\boldsymbol{w}|X)\ \forall \boldsymbol{w} : |\boldsymbol{w}| = 1$. Then we note that:

$$\forall k \in \{1, ..., N\}\ \forall X \in \mathcal{N},\quad p_{\text{pre}}(w_{k-1}w_k|X) = \sum_{Y,Z\in\mathcal{N}} p_{lc}(Y\ Z|X)p_{\text{inside}}(w_{k-1}|Y)p_{\text{pre}}(w_k|Z).$$

Then applying the identity again, we get $\forall k \in \{1, ..., N\}\ \forall X \in \mathcal{N}$,

$$p_{\text{pre}}(w_{k-2}w_{k-1}w_k|X) = \sum_{j=k-2}^{k-1}\sum_{Y,Z\in\mathcal{N}} p_{lc}(Y\ Z|X)p_{\text{inside}}(w_{k-2}...w_j)p_{\text{pre}}(w_{j+1}...w_k).$$

We can repeat this recursive procedure until $|\boldsymbol{w}| = N$, to get $p_{\text{pre}}(w_1, ..., w_N|S)$ where the inside probabilities are calculated using CKY for probability as defined above.

Now examining the runtime of this naive application, recall that running CKY returns a chart where $\text{chart}[i, j, Y] := p_{\text{inside}}(w_i...w_j|Y)$ so we only need to run CKY once to populate all of the inside probabilities.

Consider each recursive iteration, $\forall k \in \{1, ..., N\}$, we have to sum over all neighbours $Y, Z \in \mathcal{N}$, and we do this $\forall X \in \mathcal{N}$ we also have the outer sum $\sum_{j=i}^{k-1}$ up to $k = N$, so we see that for

each iteration we have a worst case runtime $O(N^2|\mathcal{N}|^3)$. We then have $N$ iterations giving us $O(N * N^2|\mathcal{N}|^3)$. Then including the pre-computations we have $O(|\mathcal{N}|^4 + N^3\,|\mathcal{N}|^3)$.

Clearly we can repeat this method for $\boldsymbol{w} := w_1, \boldsymbol{w} := w_1w_2, ..., \boldsymbol{w} := w_1w_2...w_N$ and we will have computed $p_{\text{pre}}(w_1), p_{\text{pre}}(w_1w_2), ..., p_{\text{pre}}(w_1w_2...w_N)$ in $O(N * (|\mathcal{N}|^4 + N^3\,|\mathcal{N}|^3)) = O(N\,|\mathcal{N}|^4 + N^4\,|\mathcal{N}|^3)$.

## Question 1 :: Part i)

The idea is to apply the above method but store the recursively computed prefix probabilities in a chart.

**Algorithm 3** CKY for Prefix Probabilities

**def** prefix_cky($\boldsymbol{w}, \langle \mathcal{N}, \Sigma, S, \mathcal{R}, p \rangle$) :
$N \leftarrow |\boldsymbol{w}|$
chart $\leftarrow \mathbf{0} \in \mathbb{R}^{N \times N \times |\mathcal{N}|}$ ▷ chart$[i, k, X] := p_{\text{pre}}(w_i...w_k|X)$
$\forall X, Y, Z \in \mathcal{N}$ $p_{lc}(Y\ Z|X), p_{lc}(Y|Z) \leftarrow$ prob_left_corner($\boldsymbol{w}, \langle \mathcal{N}, \Sigma, S, \mathcal{R}, p \rangle$)
inside $\leftarrow$ prob_cky($\boldsymbol{w}, \langle \mathcal{N}, \Sigma, S, \mathcal{R}, p \rangle$)
**for** $k = 1, ..., N$ **do**
  **for** $X \in \mathcal{N}$ **do**
    chart$[k, k, X] \leftarrow p(X \rightarrow w_k)$
    **for** $Y \rightarrow w_k \in \mathcal{R}$ **do** ▷ Computing $p_{\text{pre}}(w_k|X)$
      chart$[k, k, X]$ += $p_{lc}(Y|X) * p(Y \rightarrow w_k)$
    **end for**
  **end for**
**end for**
**for** $i = 1, ..., N$ **do** ▷ span length
  **for** $k = 1, ..., N$ **do**
    **for** $X \in \mathcal{N}$ **do**
      **for** $j = i, i + 1, ..., k - 1$ **do**
        **for** $Y, Z \in \mathcal{N}$ **do**
          chart$[i, k, X]$ += $p_{lc}(Y\ Z|X) *$ inside$[i, j, Y] *$ chart$[j + 1, k, Z]$
        **end for**
      **end for**
    **end for**
  **end for**
**end for**
**return** chart$[1, 1, S]$, chart$[1, 2, S]$, ..., chart$[1, N, S]$

Clearly this run sin $O(N^3|\mathcal{N}|^3 + |\mathcal{N}|^4)$ time, since we share previously computed values in the chart.

### Question 1 :: Part j)

Recall that:

$$\begin{aligned} p_{\text{pre}}(w_1 \ldots w_N) &= \prod_{n=1}^{N} p(w_n | w_0 .. w_{n-1}) \\ &= p(w_N | w_0 \ldots w_{N-1}) \prod_{n=1}^{N-1} p(w_n | w_0 .. w_{n-1}) \\ &= p(w_N | w_0 \ldots w_{N-1}) p_{\text{pre}}(w_1 \ldots w_{N-1}) \end{aligned}$$

.

Re-arranging we get:

$$\implies p(w_N | w_0 \ldots w_{N-1}) = \frac{p_{\text{pre}}(w_1 \ldots w_{N-1} w_N)}{p_{\text{pre}}(w_1 \ldots w_{N-1})}.$$

This is useful for language generation because it tells us the probability of generating a new word given some context. This could be useful for e.g predictive text where $w_0 \ldots w_{N-1}$ are known and we would like to generate suggestions for $w_N$ using probabilities.

## Question 2

See attached .ipynb file.