## Byte Pair Encoding (BPE)

**function** BYTE-PAIR ENCODING(strings $C$, number of merges $k$) **returns** vocab $V$

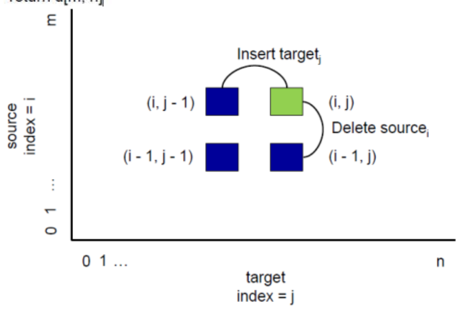
$V \leftarrow$ all unique characters in $C$ # initial set of tokens is characters
**for** $i = 1$ **to** $k$ **do** # merge tokens $k$ times
  $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in $C$
  $t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
  $V \leftarrow V + t_{NEW}$ # update the vocabulary
  Replace each occurrence of $t_L, t_R$ in $C$ with $t_{NEW}$ # and update the corpus
**return** $V$

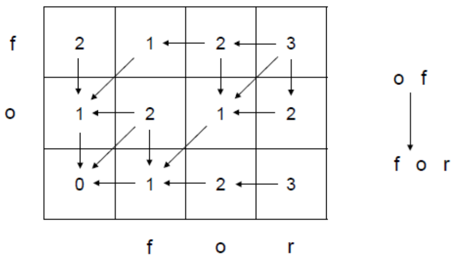
| corpus | | vocabulary |
|---|---|---|
| 5 | l o w _ | _, d, e, i, l, n, o, r, s, t, w |
| 2 | l o w e s t _ | |
| 6 | n e w e r _ | |
| 3 | w i d e r _ | |
| 2 | n e w _ | |

Merge e r to er

## Minimum Edit Distance

**function** min-edit-distance(source, target) **returns** min-distance
m ← length(source)
n ← length(target)
create a matrix d[m + 1, n + 1]
d[0, 0] ← 0
**for** each row i **from** 1 **to** m **do**
  d[i, 0] ← d[i – 1, 0] + del-cost(source$_i$)
**for** each column j **from** 1 **to** n **do**
  d[0, j] ← d[0, j – 1] + ins-cost(target$_j$)
**for** each row i **from** 1 **to** m **do**
  **for** each column j **from** 1 **to** n **do**
    d[i, j] ← min(d[i – 1, j] + del-cost(source$_i$),
      d[i, j – 1] + ins-cost(target$_j$),
      d[i – 1, j – 1] + subst-cost(source$_i$, target$_j$))
return d[m, n]




## N-Gram Count

$P(w_{1:n}) = P(w_1 w_2 \ldots w_n)$
$P(w_{1:n}) = P(w_1) \times P(w_2|w_1) \times \ldots \times P(w_n|w_{1:n-1})$

## Bigram

$$P(w_{i+1}|w_i) = \frac{C(w_i w_{i+1})}{C(w_i)}$$

## N-Gram

$$P(w_n|w_{1:n-1}) = \frac{C(w_{1:n})}{C(w_{1:n-1})}$$

## Perplexity

$$PP = \frac{1}{\sqrt[n]{P(w_{1:n})}}$$

- Lower perplexity better

## Add-$k$ Smoothing

$$P(w_{i+1}|w_i) = \frac{C(w_i w_{i+1}) + k}{C(w_i) + kV} = \frac{C^*(w_i w_{i+1})}{C(w_i)}$$

## Discount

$$d = \frac{C^*(w_i w_{i+1})}{C(w_i w_{i+1})}$$

## Practical Issue

- Use `logprob` to avoid underflow
- $p_1 \times p_2 \times \ldots \times p_n$
  - $\propto \log(p_1 \times p_2 \times \ldots \times p_n)$
    - $\propto \log p_1 + \log p_2 + \cdots + \log p_n$

## Interpolation

$$\sum_i \lambda_i = 1$$
$$P(w_0|w_{-1}) = \lambda_1 P(w_0|w_{-1}) + \lambda_2 P(w_0)$$

## Entropy

$$H(w_{1:n}) = -\sum_{w_{1:n} \in L} p(w_{1:n}) \log_2 p(w_{1:n})$$

## Cross Entropy

$$H(p, m) = \lim_{n \to \infty} -\frac{1}{n} \sum_{w_{1:n} \in L} p(w_{1:n}) \log_2 m(w_{1:n})$$

## Bayes' Theorem

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

## Naïve Bayes Classifier

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$
$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$
$$= \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$
$$= \underset{c \in C}{\operatorname{argmax}} P(w_{1:n}|c)P(c)$$

- Assume the word probabilities are independent

$$= \underset{c \in C}{\operatorname{argmax}} P(w_1|c) \cdot \ldots \cdot P(w_n|c) \cdot P(c)$$
$$= \underset{c \in C}{\operatorname{argmax}} P(c) \cdot \textstyle\prod_{i=1}^n P(w_i|c)$$
$$= \underset{c \in C}{\operatorname{argmax}} (\log P(c) + \textstyle\sum_{i=1}^n \log P(w_i|c))$$

## Parameter Estimation

$$P(c_i) = \frac{\text{count}(c_i)}{Total}$$

$$P(w_i|c_j) = \frac{\text{count}(w_i, c_j)}{\sum_{w \in V} \text{count}(w, c_j)}$$

## Add-$k$ smoothing

$$P(w_i|c_j) = \frac{\text{count}(w_i, c_j) + k}{(\sum_{w \in V} \text{count}(w, c_j)) + |V|}$$

## Confusion Matrix

| | $T$ | $F$ |
|---|---|---|
| $T$ | $TP$ | $FP$ |
| $F$ | $FN$ | $TN$ |

| Precision | Recall | Accuracy |
|---|---|---|
| $\frac{TP}{TP+FP}$ | $\frac{TP}{TP+FN}$ | $\frac{TP+TN}{TP+FP+TN+FN}$ |

$F$-Score

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

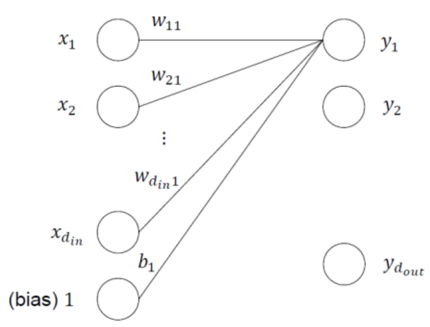
| $\beta < 1$ | $\beta = 1$ | $\beta > 1$ |
|---|---|---|
| Precision | Balance | Recall |

## Linear Models

$$y = f(x) = x \cdot W + b$$
$$x \in \mathbb{R}^{d_{in}}, W \in \mathbb{R}^{d_{in} \times d_{out}}, b \in \mathbb{R}^{d_{out}}$$

$$y = [x_1 \quad \cdots \quad x_{d_{in}} \quad 1] \begin{bmatrix} w_{11} & \cdots & w_{1d_{out}} \\ \vdots & \ddots & \vdots \\ w_{d_{in}1} & \cdots & w_{d_{in}d_{out}} \\ b_1 & \cdots & b_{d_{out}} \end{bmatrix}$$

$$x_1 \cdot w_{11} + x_2 \cdot w_{21} + \cdots + x_{d_{in}} \cdot w_{d_{in}1} + 1 \cdot b_1 = y_1$$




## Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

## Logistic Regression

$$\sigma(x \cdot W + b) = \frac{1}{1 + e^{-(x \cdot W + b)}}$$

## Softmax Function

$$\text{softmax}(x) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

## Multinomial Logistic Regression

$$\text{softmax}(x \cdot W + b)$$

## Training as Optimization

$$\hat{\Theta} = \operatorname{argmin} \left( \frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \Theta), y_i) + \lambda R(\Theta) \right)$$

## Loss functions

$$\frac{1}{n} \sum_{i=1}^{n} L(f(x_i, \Theta), y_i)$$

## Binary Cross-Entropy Loss (Logistic Loss)

$$L_{Logistic}(\hat{y}, y) = -y \log_2 \hat{y} - (1 - y) \log_2 (1 - \hat{y})$$

- Used in binary classification
- Output conditional probability

## Cross-Entropy Loss (Negative Log Likelihood)

$$L_{cross-entropy}(\hat{y}, y) = -\sum_i y_i \log_2 \hat{y}_i$$

- Output multinomial distribution over labels

## Squared (quadratic) loss

$$L(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$$

## Regularization

$$\lambda R(\Theta)$$

### $L_1$ regularization

$$R_{L_1}(W) = \|W\|_1 = \sum_{i,j} |W_{i,j}|$$

### $L_2$ regularization

$$R_{L_2}(W) = \|W\|_2^2 = \sum_{i,j} W_{i,j}{}^2$$

## Gradient Descent

Find $v$ such that $L(w) > L(w, v)$ and $\|v\|$ is small

$$L(w,v) \approx L(w) + v \cdot \nabla L(w)$$

$$\nabla L(w) = \left(\frac{\partial L}{\partial w_0}, \dots, \frac{\partial L}{\partial w_n}\right)$$

$$v = -\alpha \nabla L(w)$$

## Rectified Linear Unit (ReLU)

$$\mathrm{ReLU}(x) = \phi(x) = g(xW + b)$$

$$W = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 0 & -1 \end{pmatrix}$$

## Hyperbolic Tangent (tanh)

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\frac{d}{dx}\tanh(x) = \frac{(e^x + e^{-x})^2 - (e^x - e^{-x})^2}{(e^x + e^{-x})^2}$$

$$= 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2$$

$$= 1 - \tanh^2(x)$$

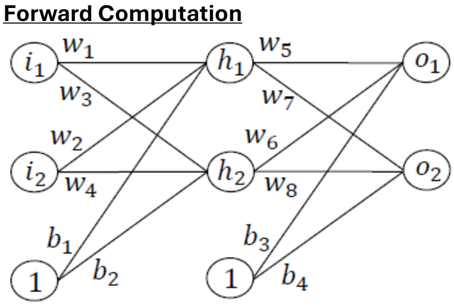
## Neural Network

$$h_1 = g_1(xW_1 + b_1)$$

$$h_2 = g_2(h_1 W_2 + b_2)$$

$$y = h_2 W_3 + b_3$$

## Forward Computation




$$[s_1 \quad s_2] = [i_1 \quad i_2 \quad 1] \begin{bmatrix} w_1 & w_3 \\ w_2 & w_4 \\ b_1 & b_2 \end{bmatrix}$$

$$[h_1 \quad h_2] = \sigma([s_1 \quad s_2])$$

$$[s_3 \quad s_4] = [h_1 \quad h_2 \quad 1] \begin{bmatrix} w_5 & w_7 \\ w_6 & w_8 \\ b_3 & b_4 \end{bmatrix}$$

$$[o_1 \quad o_2] = \sigma([s_3 \quad s_4])$$

$$s_1 = w_1 i_1 + w_2 i_2 + b_1$$

$$s_2 = w_3 i_1 + w_4 i_2 + b_2$$

$$h_1 = \frac{1}{1+e^{-s_1}}, \qquad h_2 = \frac{1}{1+e^{-s_2}}$$

$$s_3 = w_5 h_1 + w_6 h_2 + b_3$$

$$s_4 = w_7 h_1 + w_8 h_2 + b_4$$

$$o_1 = \frac{1}{1+e^{-s_3}}, \qquad o_2 = \frac{1}{1+e^{-s_4}}$$

$$L = \frac{1}{2}[(o_1 - t_1)^2 + (o_2 - t_2)^2]$$

## Backward Computation

### Layer 2

$$\frac{\partial L}{\partial w_5} = \frac{\partial L}{\partial o_1} \frac{\partial o_1}{\partial s_3} \frac{\partial s_3}{\partial w_5}$$

$$\frac{\partial L}{\partial o_1} = \frac{1}{2}[2(o_1 - t_1) + 0] = (o_1 - t_1)$$

$$\frac{\partial o_1}{\partial s_3} = \frac{v-u}{v^2} = \frac{1+e^{-s_3}-1}{(1+e^{-s_3})^2} = o_1(1 - o_1)$$

$$\frac{\partial s_3}{\partial w_5} = h_1 + 0 + 0 = h_1$$

### Layer 1

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial s_1} \frac{\partial s_1}{\partial w_1}$$

$$= \left(\frac{\partial L}{\partial s_3} \frac{\partial s_3}{\partial h_1} + \frac{\partial L}{\partial s_4} \frac{\partial s_4}{\partial h_1}\right) \frac{\partial h_1}{\partial s_1} \frac{\partial s_1}{\partial w_1}$$

$$\frac{\partial L}{\partial s_3} = \frac{\partial L}{\partial o_1} \frac{\partial o_1}{\partial s_3} \qquad \frac{\partial L}{\partial s_4} = \frac{\partial L}{\partial o_2} \frac{\partial o_2}{\partial s_4}$$

## Word2vec

$$P(D = 1 | w, c) = \frac{1}{1 + e^{-wc}}$$

$$P(D = 0 | w, c) = \frac{1}{1 + e^{wc}}$$

## Cosine similarity

$$\mathrm{sim}_{\cos}(u, v) = \frac{u \cdot v}{\|u\| \|v\|} = \frac{\sum_i u_i \cdot v_i}{\sqrt{\sum_i u_i^2} \sqrt{\sum_i v_i^2}}$$

## Simple RNN

$$s_i = R(s_{i-1}, x_i) = g([s_{i-1}; x_i]W + b)$$

## Self-Attention Network

$$y_i = \sum_{j=1}^{i} a_{ij} x_j, \; a_{ij} = \frac{e^{sim(x_i, x_j)}}{\sum_{k=1}^{i} e^{sim(x_i, x_k)}}$$

## Convert a CFG with No $\varepsilon$-production to CNF

### Step 1

- Add all non-unit productions to $P'$
  - $A \to B$ where $B$ is not a single non-terminal symbol
- If $A \to \cdots \to B$ and $B \to \alpha$, add $A \to \alpha$ to $P'$

### Step 2

- If $A \to X_1 X_2 \dots X_n$ where $n \geq 2$ and $X_i$ is a terminal symbol, then replace $X_i$ with $C_i$ where $C_i \to \alpha_i$

### Step 3

- For each $A \to B_1 B_2 B_3 \dots$, replace with $A \to B_1 D_1$ where $D_1 \to B_2 D_2$, $D_2 \to B_3 D_3$ and so on.

## CKY Algorithm

```
function CKY-Parse(words, grammar) returns table

for j ← from 1 to Length(words) do
  table[j – 1, j] ← { A | A → words[j]  ∈ grammar }
  for i ← from j – 2 downto 0 do
    for k ← i + 1 to j – 1 do
      table[i, j] ← table[i, j] ∪
                    { A | A → BC ∈ grammar, B ∈ table[i, k], C ∈ table[k, j] }
```

| 0 time 1 | fruit 2 | flies 3 |
|---|---|---|
| S → time 0.1<br>VP → time 0.3<br>N → time 0.4<br>V → time 0.6<br>[0,1] | NP → N N (k = 1)<br>0.4 * 0.2 * 1.0 = 0.08<br>[0,2] | S → V NP (k = 1)<br>0.6 * 0.08 * 0.4 = 0.0192<br>VP → V NP (k = 1)<br>0.6 * 0.08 * 0.5 = 0.024<br>S → NP VP (k = 2)<br>0.08 * 0.2 * 0.4 = 0.0064<br>(lower probability, not kept)<br>[0,3] |
| | N → fruit 0.2<br>[1,2] | NP → N N (k = 2)<br>0.2 * 0.4 * 1.0 = 0.08<br>[1,3] |
| | | S → flies 0.1<br>VP → flies 0.2<br>N → flies 0.4<br>V → flies 0.4<br>[2,3] |

## Evaluating Parsers

Precision $\qquad \dfrac{\#\text{correct in parser's parse}}{\#\text{correct in treebank's parse}}$

Recall $\qquad \dfrac{\#\text{correct in parser's parse}}{\#\text{correct in parser's parse}}$

A constituent is correct if there is a constituent in the parse's parse tree that spans the same words with the same non-terminal symbol. Note that the internal parse tree structure rooted at a non-terminal symbol does not matter.

## Derivative rules

### Chain rule

$$\frac{\partial}{\partial x} u(v) = \frac{\partial u}{\partial v} \cdot \frac{\partial v}{\partial x}$$

### Product rule

$$\frac{\partial}{\partial x}(uv) = u\frac{\partial v}{\partial x} + v\frac{\partial u}{\partial x}$$

### Quotient rule

$$\frac{\partial}{\partial x}\left(\frac{u}{v}\right) = \frac{v - u}{v^2}$$

| | |
|---|---|
| $cx^a$ | $acx^{a-1}$ |
| $e^{ax}$ | $ae^{ax}$ |
| $e^u$ | $e^u \cdot \frac{du}{dx}$ |
| $a^u$ | $a^u \cdot \ln(a) \cdot \frac{du}{dx}$ |
| $\ln(u)$ | $\frac{1}{u} \cdot \frac{du}{dx}$ |
| $\log_a(u)$ | $\frac{1}{u\ln(a)} \cdot \frac{du}{dx}$ |
| $\sin(u)$ | $cos(x) \cdot \frac{du}{dx}$ |
| $\cos(u)$ | $-sin(x) \cdot \frac{du}{dx}$ |
| $\tan(u)$ | $\sec^2(u) \cdot \frac{du}{dx}$ |
| $\sin^{-1}(u)$ | $\frac{1}{\sqrt{(1-u^2)}} \cdot \frac{du}{dx}$ |
| $\cos^{-1}(u)$ | $\frac{-1}{\sqrt{(1-u^2)}} \cdot \frac{du}{dx}$ |
| $\tan^{-1}(u)$ | $\frac{1}{1+u^2} \cdot \frac{du}{dx}$ |