NCTU Deep Learning

Week 10

Natural Language Processing

$$P(x_1, \ldots, x_\tau) = P(x_1, \ldots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t \mid x_{t-n+1}, \ldots, x_{t-1}). \qquad (12.5)$$

$$P(\texttt{THE DOG RAN AWAY}) = P_3(\texttt{THE DOG RAN}) P_3(\texttt{DOG RAN AWAY}) / P_2(\texttt{DOG RAN}). \qquad (12.7)$$

Improve with:
-Smoothing
-Backoff
-Word categories

- An important predecessor to deep NLP is the family of models based on *n*-grams:

Word Embeddings

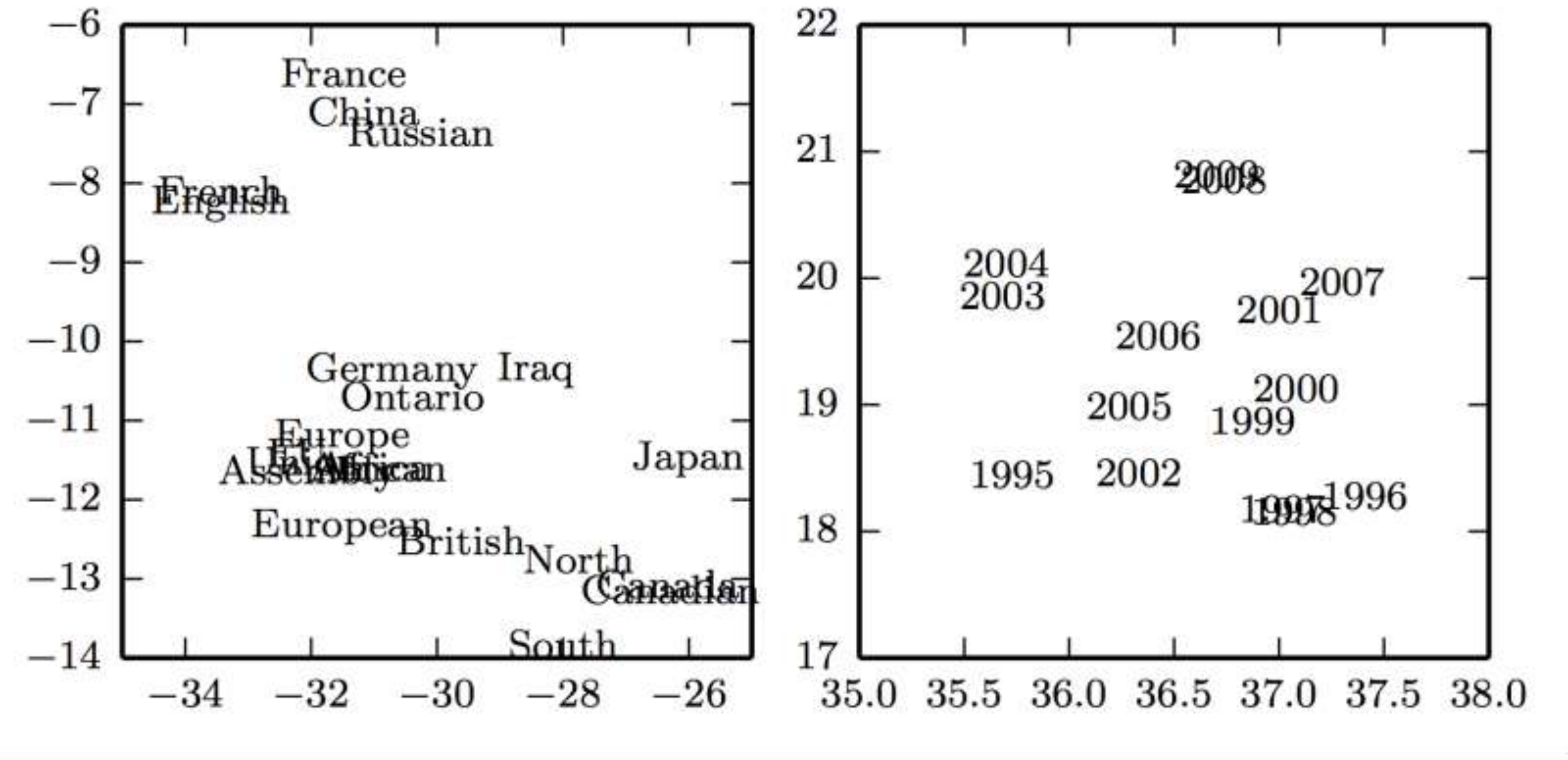


Figure 12.3

CBOW and Skip-gram

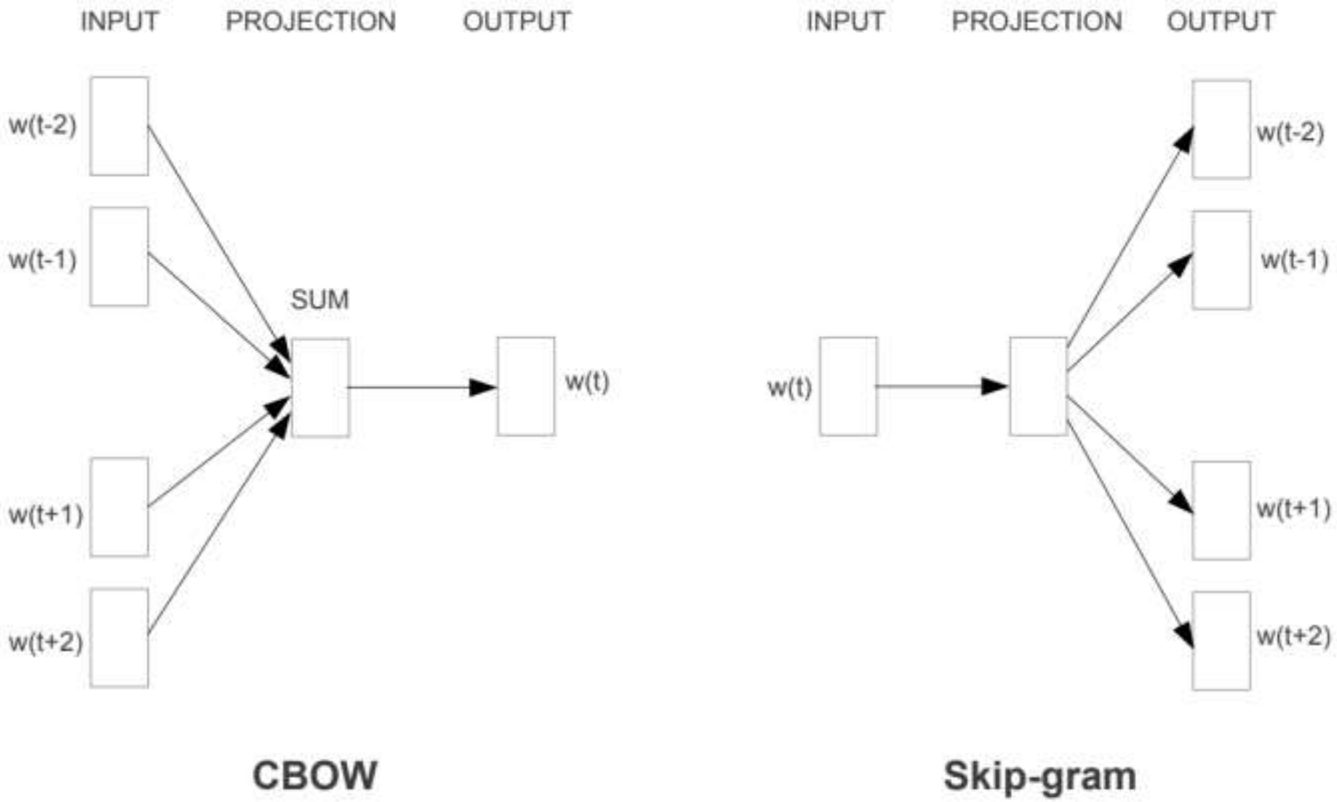
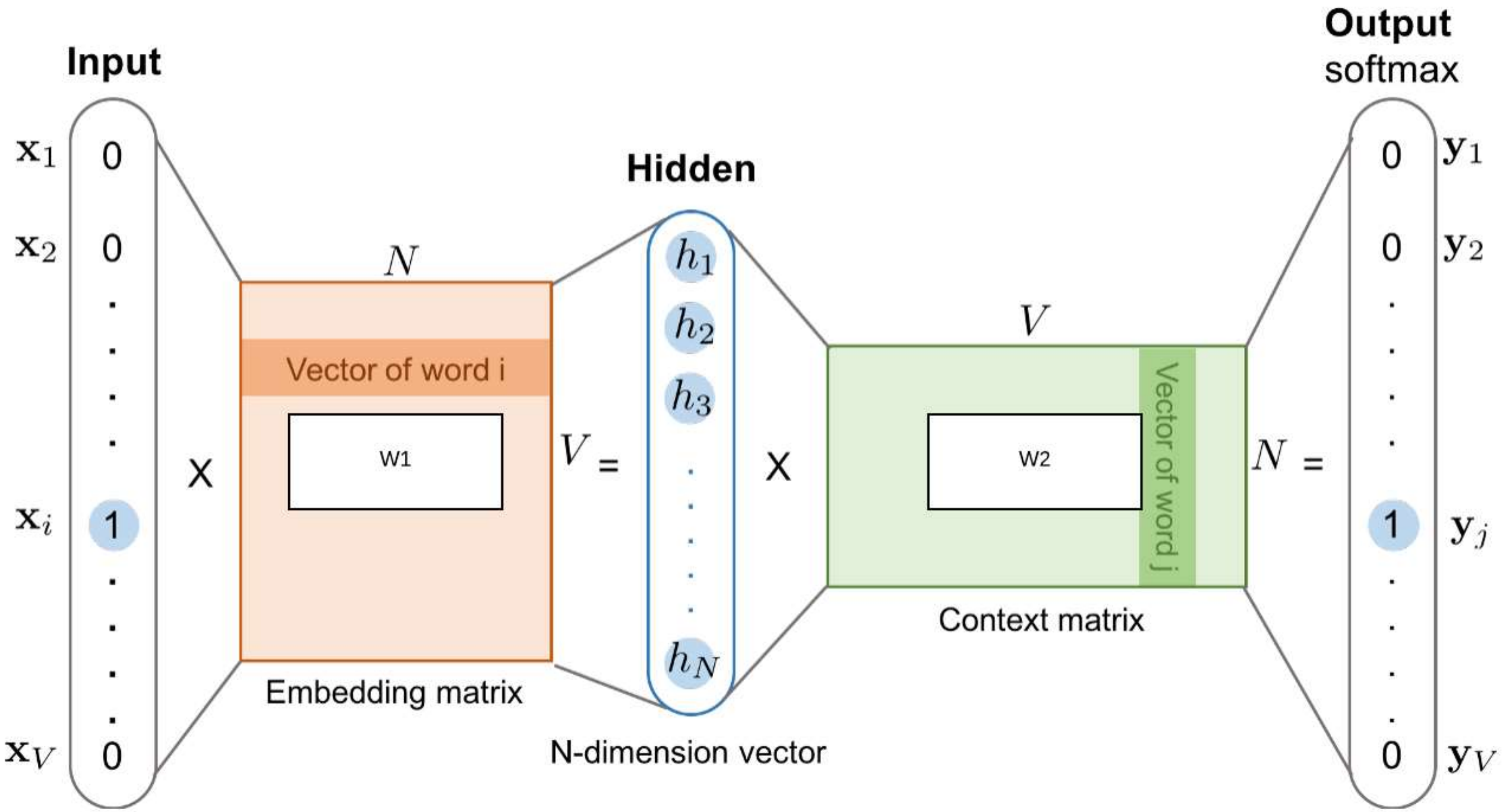


Figure 1: New model architectures. The CBOW architecture predicts the current word based on the context, and the Skip-gram predicts surrounding words given the current word.

Word2vec, skip-gram

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c,j\neq 0}\log p(w_{t+j}|w_t)$$

word2vec


Input
Output
softmax
Hidden
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_i$
$\mathbf{x}_V$
0
0
1
0
N
Vector of word i
W1
Embedding matrix
V =
X
h_1
h_2
h_3
h_N
N-dimension vector
X
V
W2
Vector of word j
Context matrix
N =
$\mathbf{y}_1$
$\mathbf{y}_2$
$\mathbf{y}_j$
$\mathbf{y}_V$

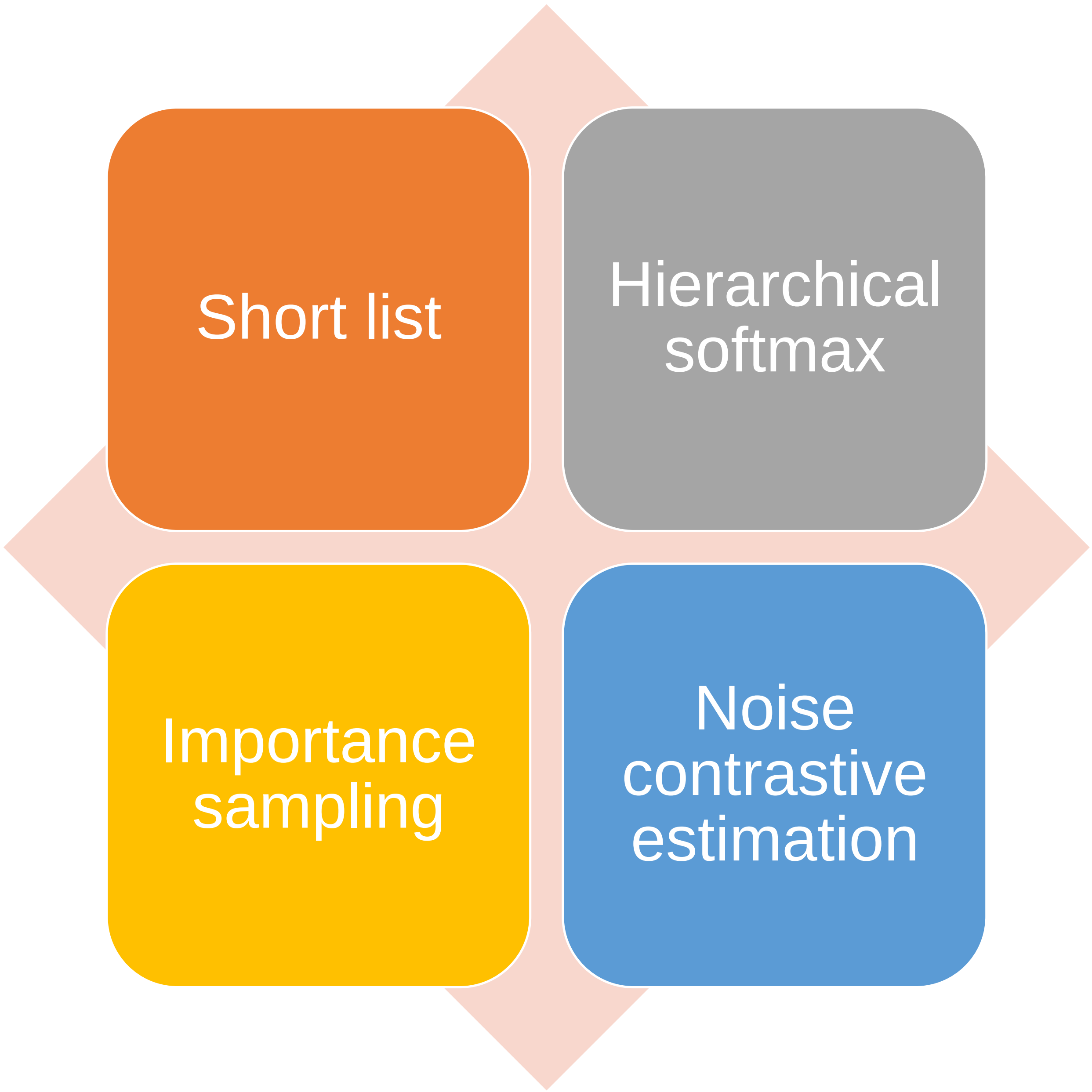
Word2vec, Subsampling of Frequent Words

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

Parametrization for word2vec

- CBOW(fast) or skip-gram(better for infrequent words)
- Hierachical softmax or negative sampling
- Dimensionality (100-1000)
- Context window

High-Dimensional Output Layers for Large Vocabularies



softmax

$$p_\theta(w \mid c) = \frac{u_\theta(w, c)}{\sum_{w' \in V} u_\theta(w', c)}$$

A Hierarchy of Words and Word Categories

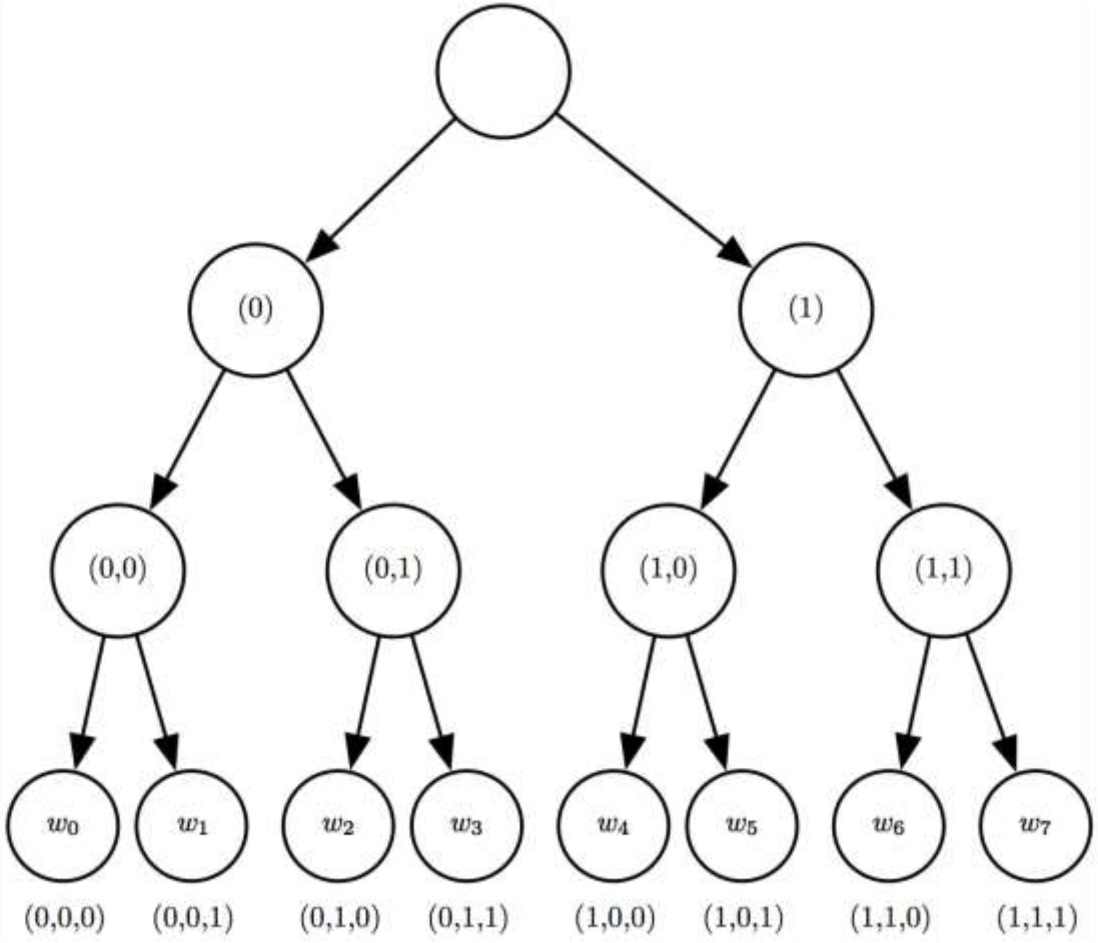
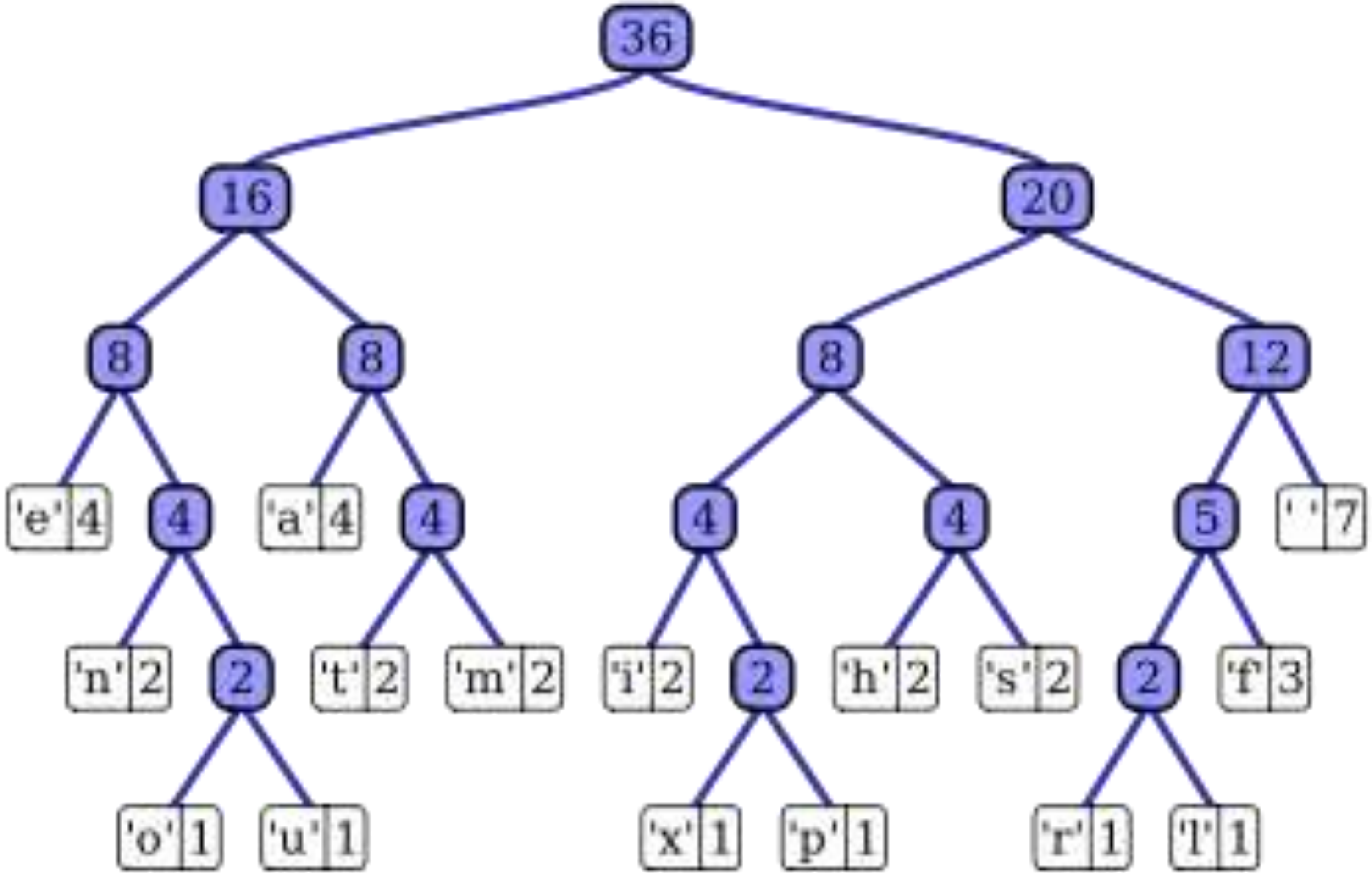


Figure 12.4

Huffman Tree



Noise Contrastive Estimation

noise distribution

$$p(d, w \mid c) = \begin{cases} \frac{k}{1+k} \times q(w) & \text{if } d = 0 \\ \frac{1}{1+k} \times \tilde{p}(w \mid c) & \text{if } d = 1 \end{cases}.$$

Noise Contrastive Estimation

$$p(D=0 \mid c,w) = \frac{\frac{k}{1+k} \times q(w)}{\frac{1}{1+k} \times \tilde{p}(w \mid c) + \frac{k}{1+k} \times q(w)}$$

$$= \frac{k \times q(w)}{\tilde{p}(w \mid c) + k \times q(w)}$$

$$p(D=1 \mid c,w) = \frac{\tilde{p}(w \mid c)}{\tilde{p}(w \mid c) + k \times q(w)}.$$

Noise Contrastive Estimation

$$p(D = 0 \mid c, w) = \frac{k \times q(w)}{u_\theta(w, c) + k \times q(w)}$$

$$p(D = 1 \mid c, w) = \frac{u_\theta(w, c)}{u_\theta(w, c) + k \times q(w)}.$$

Negative Sampling

$$\log \sigma({v'_{w_O}}^\top v_{w_I}) + \underbrace{\sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-{v'_{w_i}}^\top v_{w_I}) \right]}_{\text{K noise samples}}$$

Software for word embeddings

- Word2vec
- GloVe
- fastText
- MUSE (Multilingual Unsupervised and Supervised Embeddings)
- Gensim

NLP

NLTK

Spacy

BERT

GPT-2

ULMFiT

Jieba

CKIP

Articut

Neural Machine Translation

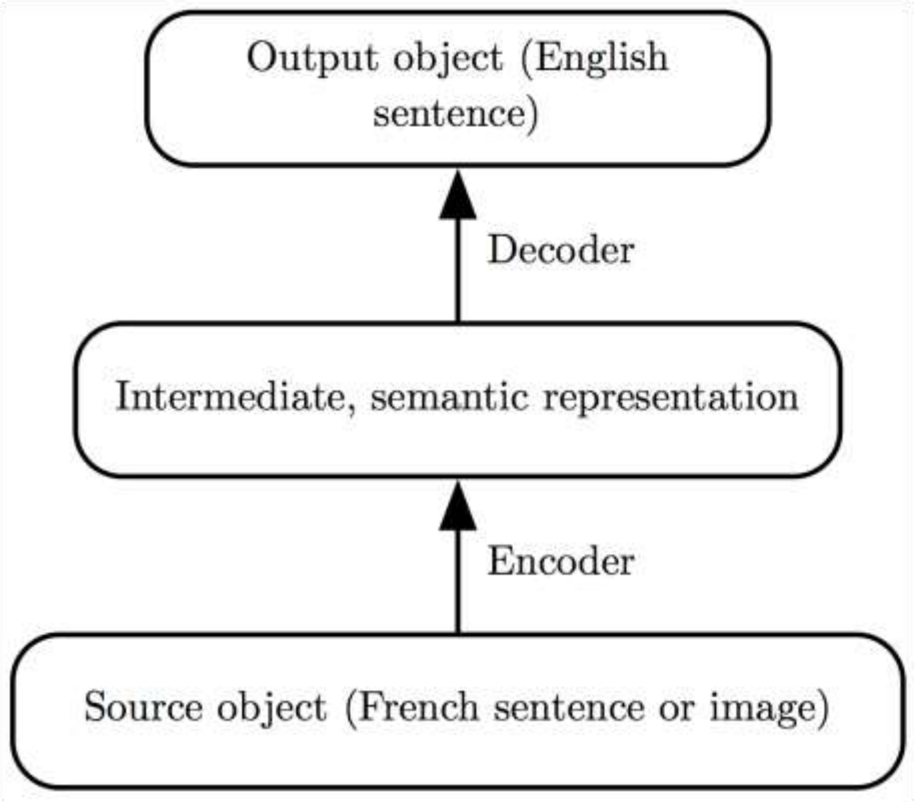
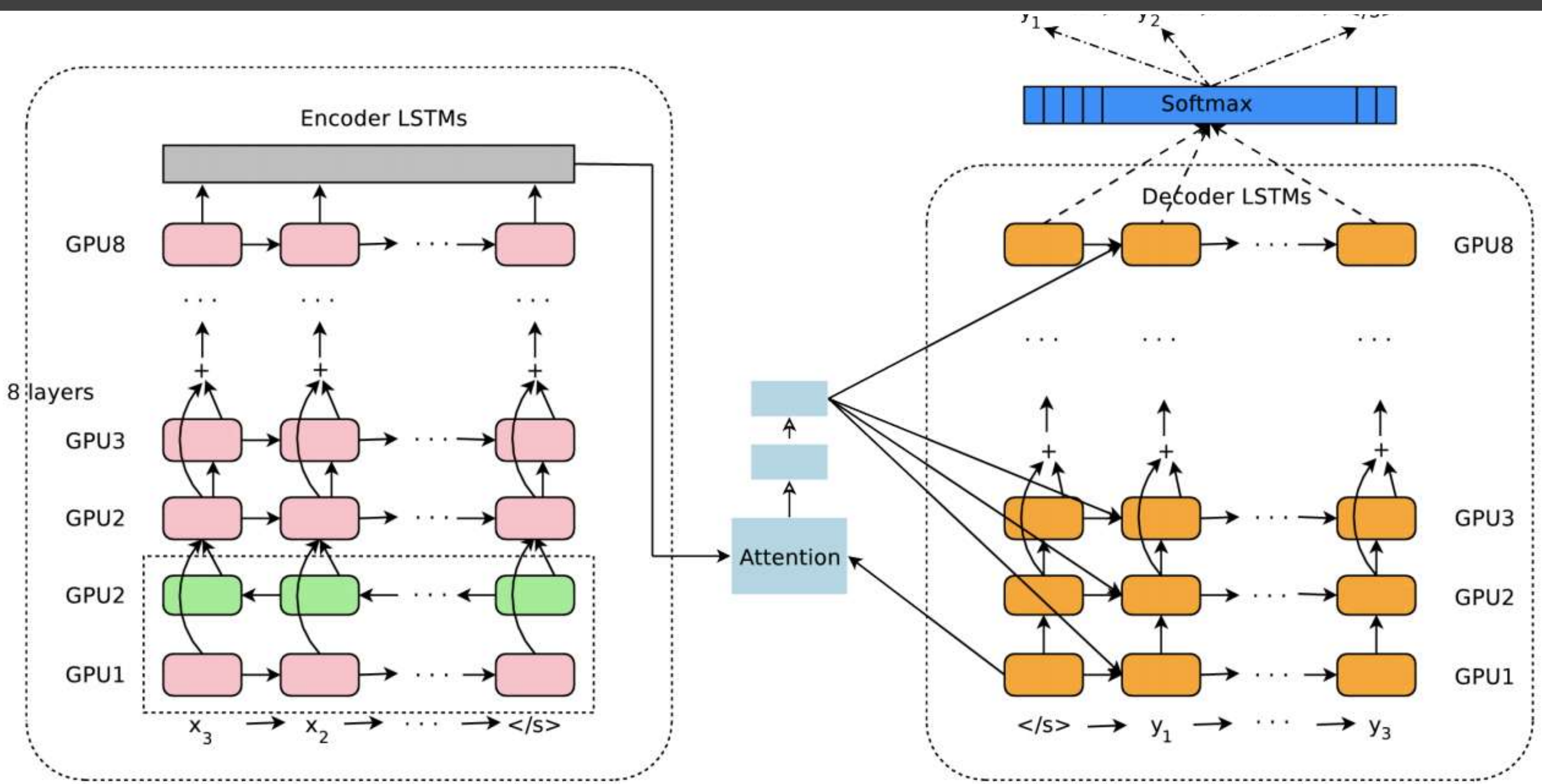


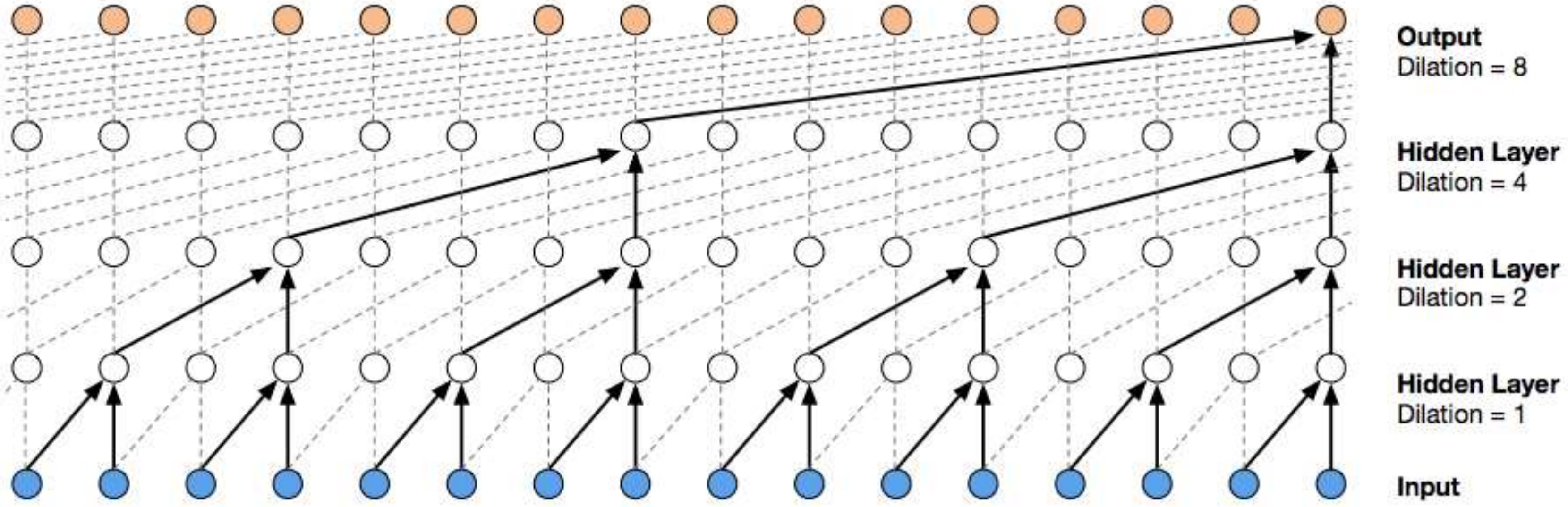
Figure 12.5

Google Neural Machine Translation



Wu et al 2016

Speech Synthesis



WaveNet

(van den Oord et al, 2016)

Speech Recognition

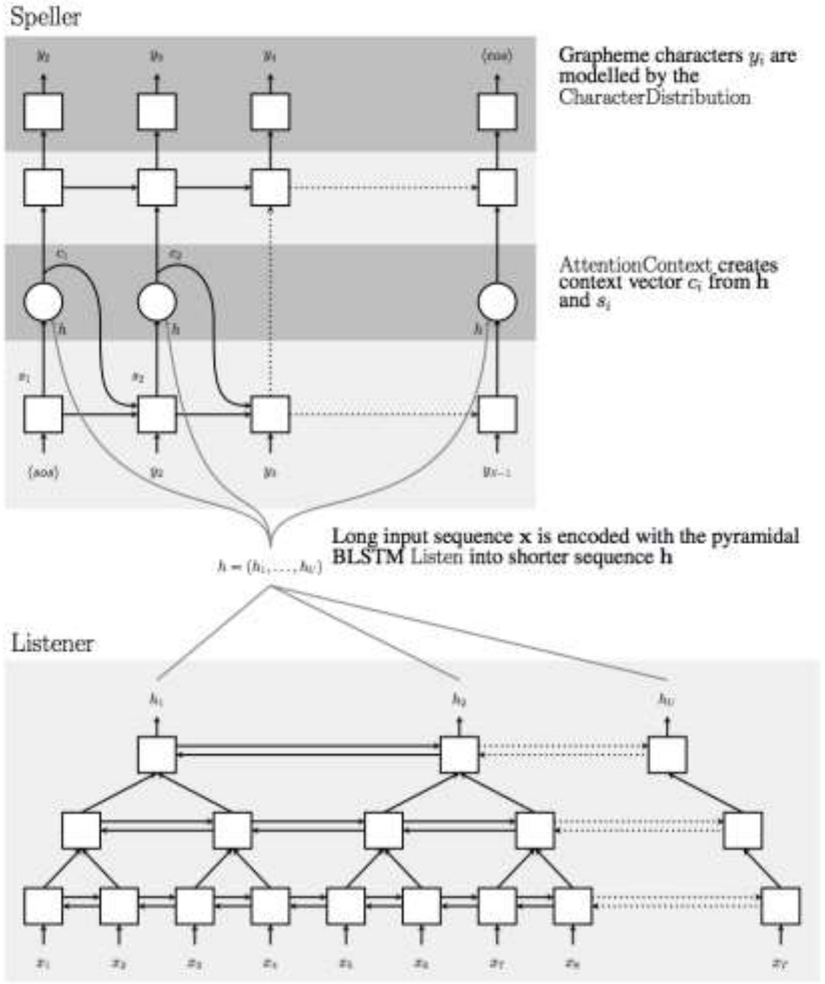


Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence x into high level features h, the speller is an attention-based decoder generating the y characters from h.

“Listen, Attend, and Spell”

Graphic from [Chan et al 2015](#)

[Current speech recognition](#) is based on seq2seq with attention

Deep RL for Atari game playing

Figure 3: The leftmost plot shows the predicted value function for a 30 frame segment of the game Seaquest. The three screenshots correspond to the frames labeled by A, B, and C respectively.

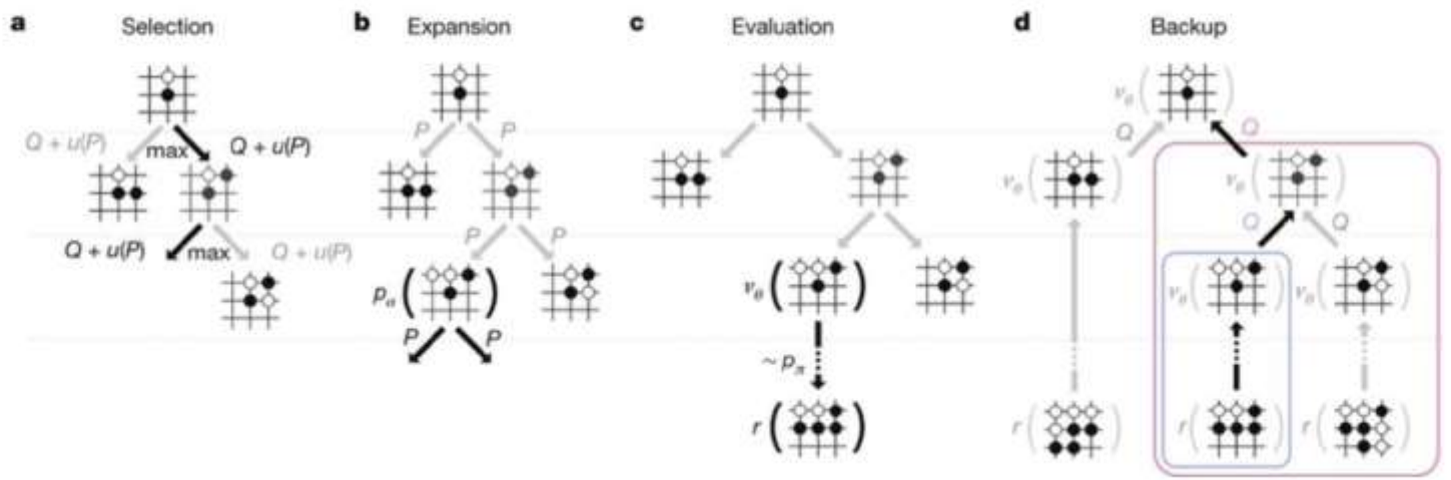
(Mnih et al 2013)

Convolutional network estimates the value function (future rewards) used to guide the game-playing agent.

(Note: deep RL didn't really exist when we started the book,
became a success while we were writing it, extremely hot topic by the time the book was printed)

Superhuman Go

Monte Carlo tree search, with convolutional networks for value function and policy



a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

(Silver et al, 2016)