IN4080 – 2020 FALL

NATURAL LANGUAGE PROCESSING

Jan Tore Lønning

Neural LMs, Recurrent networks, Sequence labeling

Lecture 12, 2 Nov.

Today

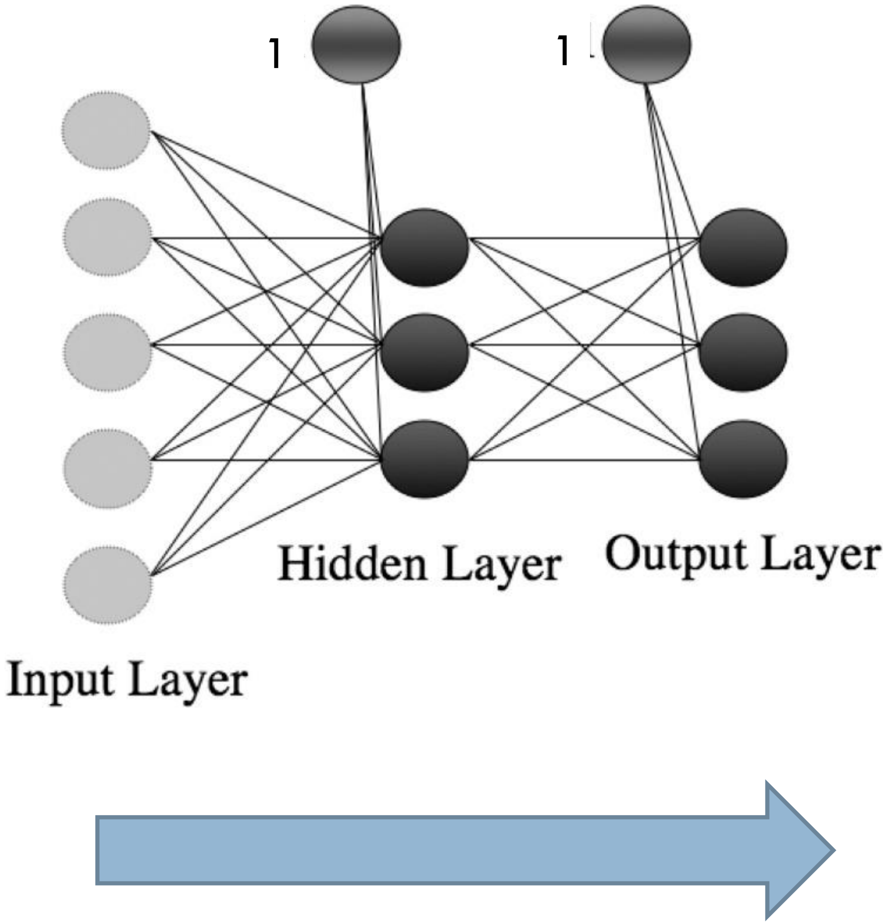
- Feedforward neural networks (partly recap)
- Recurrent networks
- Information Extraction

Today

- Feedforward neural networks (partly recap)
  - Model
  - Training
  - Computational graphs
  - Neural Language Models
- Recurrent networks
- Information Extraction

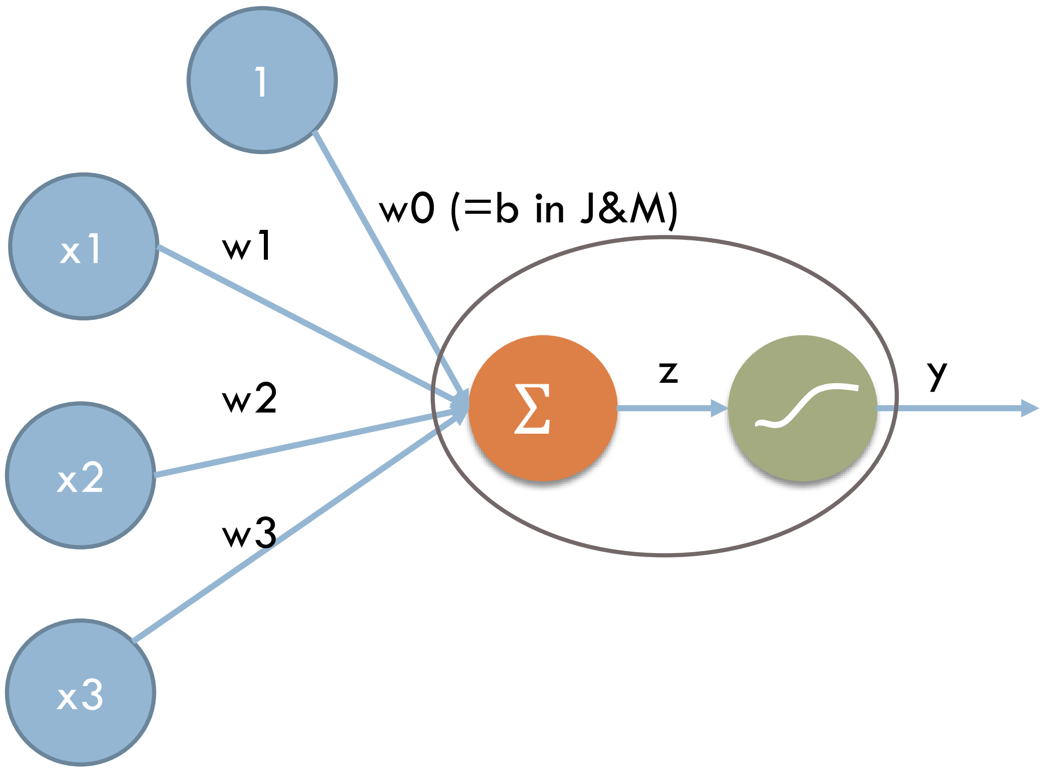
Feed forward network

- An input layer
- An output layer: the predictions
- One or more hidden layers
- Connections from one layer to the next (from left to right)
- A weight at each connection



The hidden nodes

- Each hidden node is like a small logistic regression:
  - First sum of weighted inputs :
  - $z = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$
    - where $x = 1$ and $w_0 = b$, bias
    - alternatively, $z = \sum_{i=1}^{m} w_i x_i + b$
  - Then the result is run through an activation function, e.g. σ
  - $y = \sigma(z) = \frac{1}{1+e^{-\vec{w}\cdot\vec{x}}}$

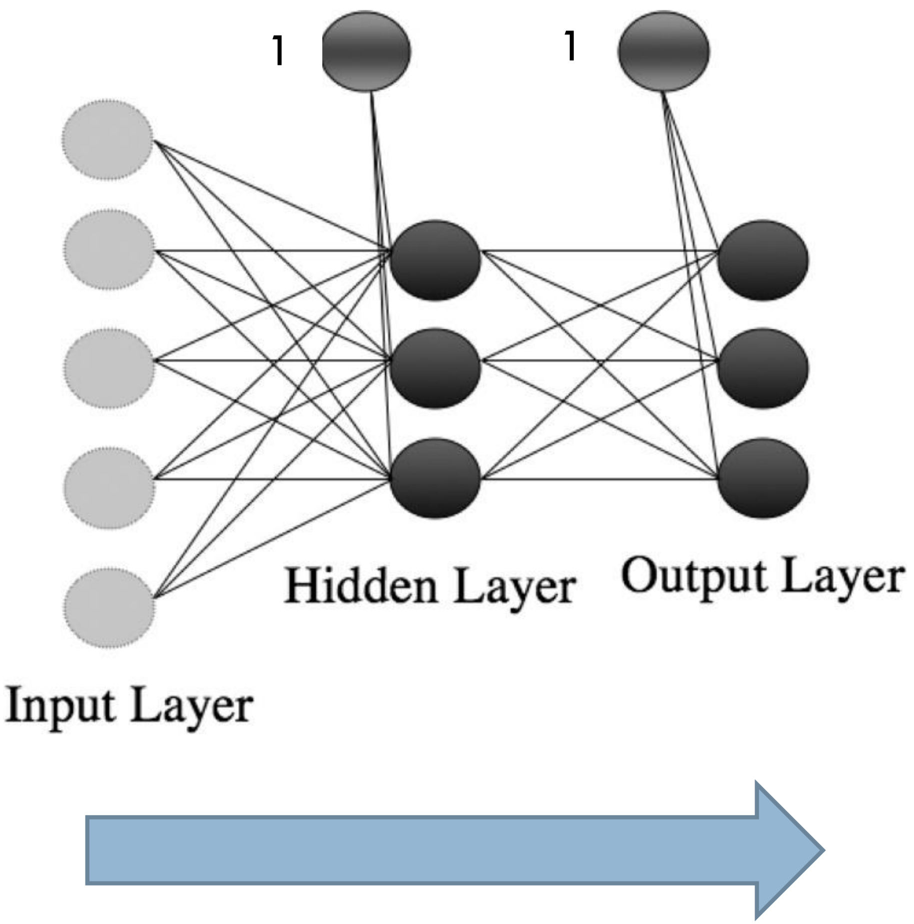


It is the non-linearity of the activation function which makes it possible for MLP to predict non-linear decision boundaries

The output layer

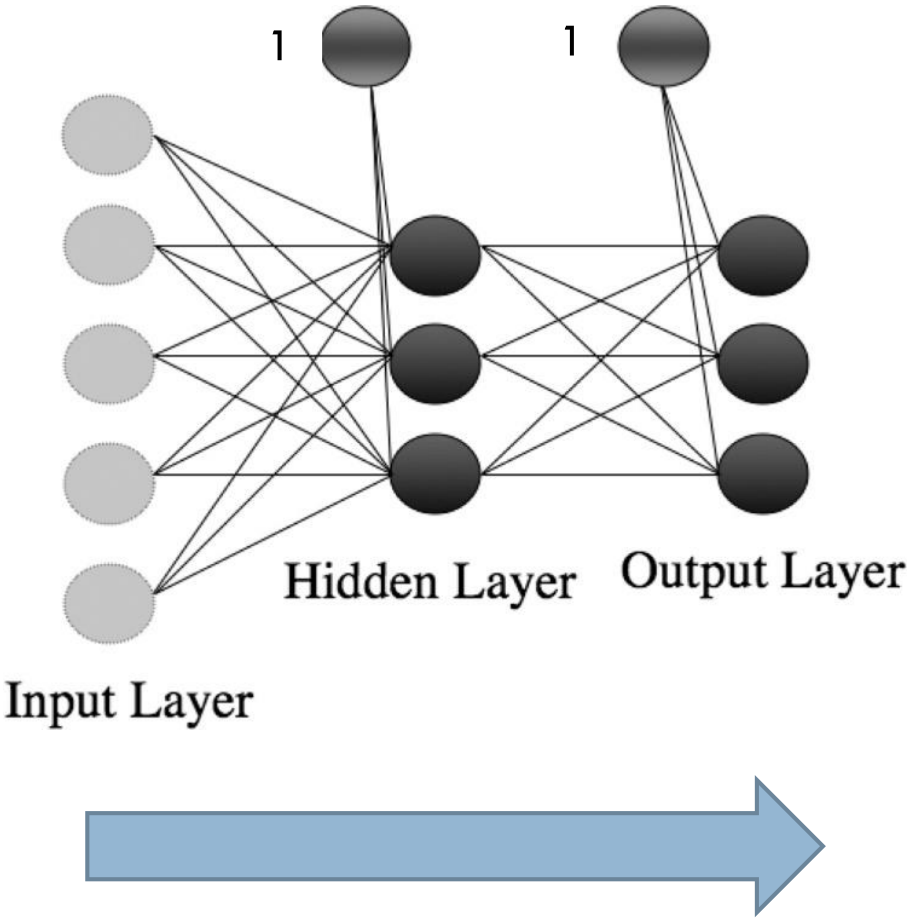
Alternatives

- Regression:
  - One node
  - No activation function
- Binary classifier:
  - One node
  - Logistic activation function
- Multinomial classifier
  - Several nodes
  - Softmax
- + more alternatives



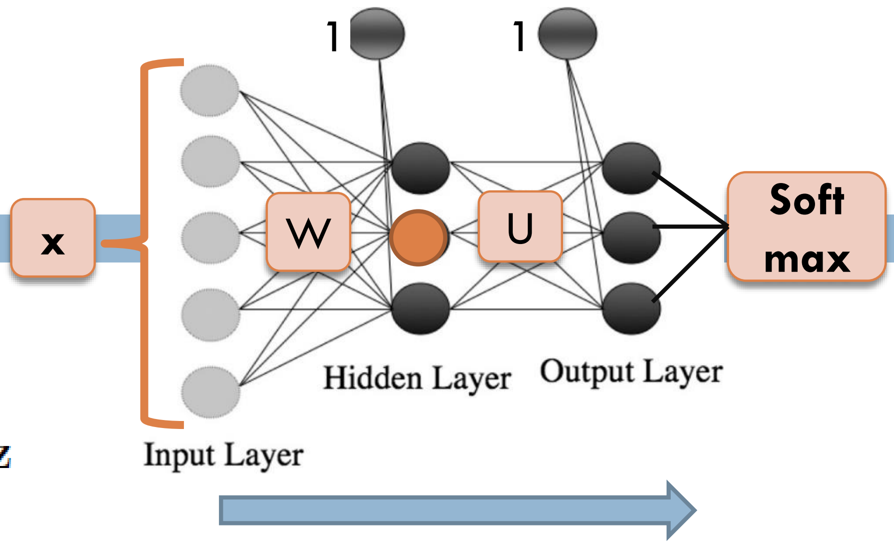
Forward

- Applying the network:
  - Start with the input vector
  - Run it step-by-step through the network



Forward

$$W\mathbf{x} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = \mathbf{z}$$



- Each layer can be considered a vector
- The connections between the layers: a matrix
- Running it through the connections: matrix multiplication

Example network:

- $\boldsymbol{h} = \sigma(W\boldsymbol{x} + b)$
- $\boldsymbol{z} = U\boldsymbol{h}$
- $\boldsymbol{y} = \mathrm{softmax}(\boldsymbol{z})$

Beware: Jurafsky and Martin uses $w_{i,j}$ where Marsland, IN3050, uses $w_{j,i}$
Marsland, and Goldberg (IN5550): $\boldsymbol{h} = \sigma(\boldsymbol{x}W + b)$, where $\boldsymbol{x}$ is a row vector

Today

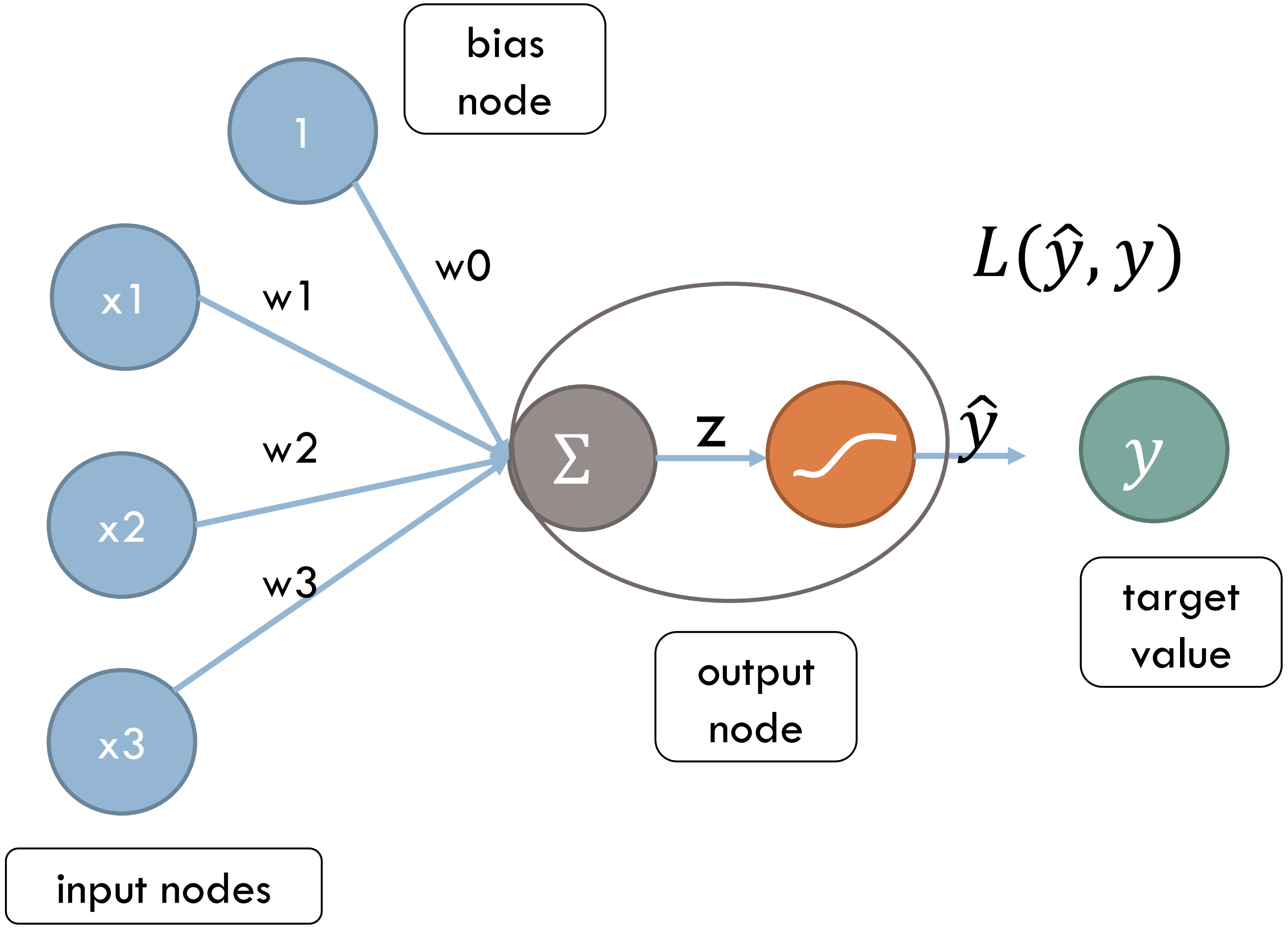
- Feedforward neural networks (partly recap)
  - Model
  - Training
  - Computational graphs
  - Neural Language Models
- Recurrent networks
- Information Extraction

Learning in neural networks

- Introduce a loss function: $L(\hat{\boldsymbol{y}}\,,\boldsymbol{y})$
  - tells something about the difference between $\hat{\boldsymbol{y}}$ and$\boldsymbol{y}$
- Update w_i according to how much it contributes to the loss
  - w_i: $w_i \leftarrow w_i - \eta \frac{\partial}{\partial w_i} L(\hat{\boldsymbol{y}}\,,\boldsymbol{y})$
- Calculate the partial derivatives using the chain rule $\frac{\partial}{\partial w_i} L(\hat{\boldsymbol{y}}\,,\boldsymbol{y})$
  - "Follow the network backwards collecting partial derivaties along the path"

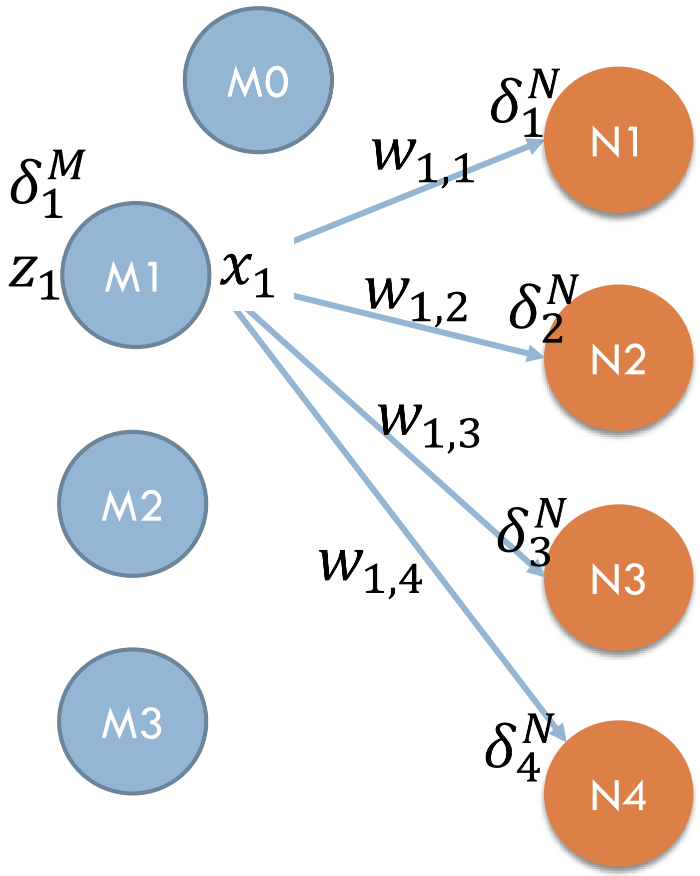
Example: Logistic regression as a network

- $z = \sum_{i=0}^{m} w_i x_i = \boldsymbol{w} \cdot \boldsymbol{x}$
- $\hat{y} = \sigma(z) = \frac{1}{1+e^{-z}}$
- $\frac{\partial}{\partial \widehat{w_i}} L_{CE} = \frac{\partial}{\partial \hat{y}} L_{CE} \times \frac{\partial \hat{y}}{\partial z} \times \frac{\partial z}{\partial w_i}$



Learning in multi-layer networks

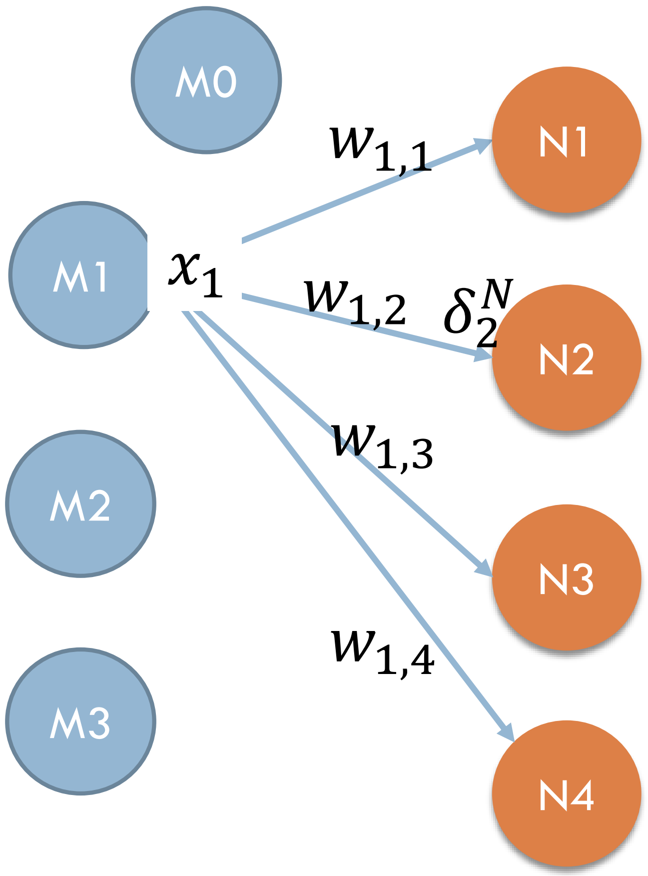
- If N is the output layer, calculate the error terms δ_j^N as before from the loss and the activation function at each node N_j
- If M is a hidden layer: Calculate the error term at the nodes combining
  - A weighted sum of the error terms at layer N
  - The derivative of the activation function
  - $\delta_i^M = \left(\sum_{j=1}^{n} w_{i,j}\delta_j^N\right)\frac{dx_i}{dz_i}$
    - where $x_i = \sigma(z_i)$, where $z_i = \sum(\dots)$



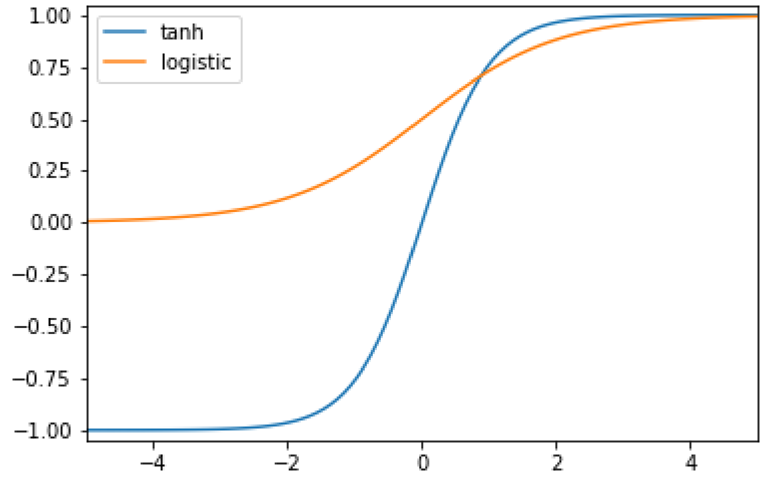
Learning in multi-layer networks

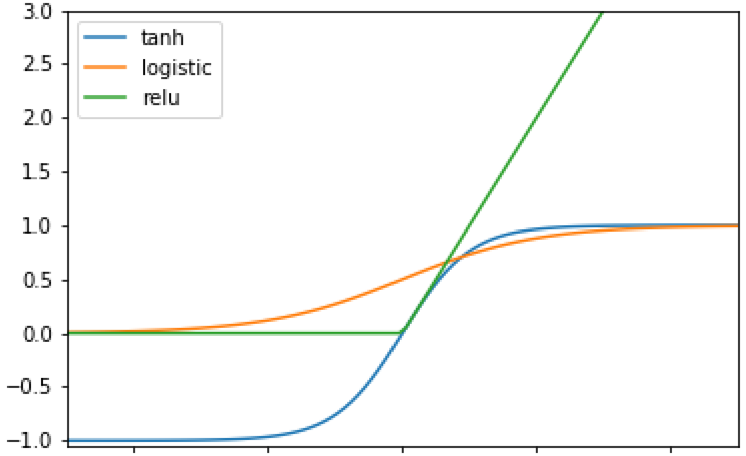
- By repeating the process, we get error terms at all nodes in all the hidden layers.
- The update of the weights between the layers can be done as before:
- $w_{i,j} = w_{i,j} - x_i \delta_j^N$
  - where x_i is the value going out of node M_i

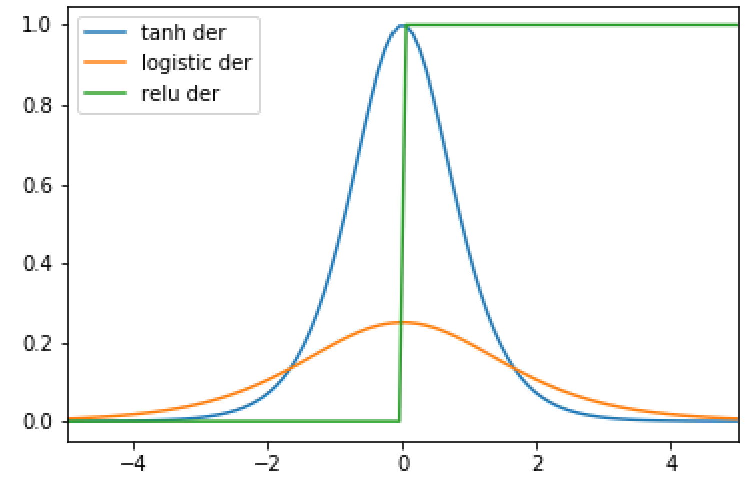
Beware: We have here used $w_{i,j}$ for the weight connecting node i and node j, while Jurafsky and Martin uses $w_{j,i}$ for this edge.



Alternative activation functions







- There are alternative activation functions
- $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- $ReLU(x) = \max(x, 0)$
- ReLU is the preferred method in hidden layers in deep networks

Footnote

Equation (5.35) is wrong. It should have been something like

$$\frac{\partial L_{CE}}{\partial w_{k,i}} = -(1\{y = k\} - \frac{e^{W_{[k,:]}\cdot\mathbf{x}+b_k}}{\sum_{j=1}^{K} e^{W_{[j,:]}\cdot\mathbf{x}+b_j}})x_i$$

where $w_{k,i}$ is the weight on the edge from input node i to output node k, and $W_{[k,:]}$ is row k in the weight matrix (written in numpy style, there might be better notations). We are assuming a similar representation as in chapter 9, where vectors are represented as column matrices and the result of sending X through the weights are written WX.

The same equation also appears in chapter 7 as (7.17)

Today

- Feedforward neural networks (partly recap)
  - Model
  - Training
  - Computational graphs
  - Neural Language Models
- Recurrent networks
- Information Extraction

Computational graphs

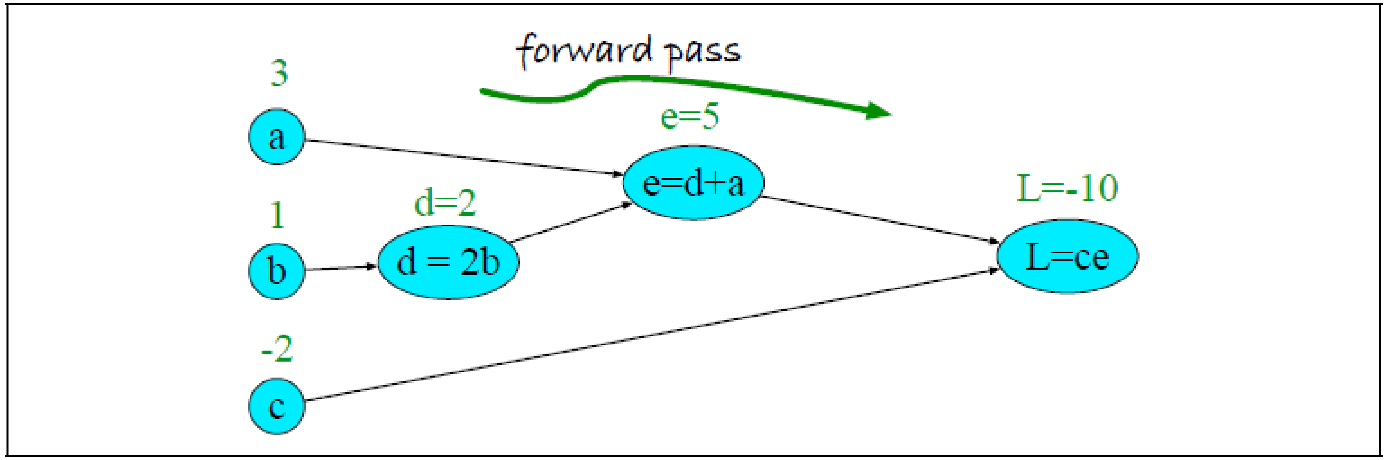


Figure 7.9 Computation graph for the function $L(a,b,c) = c(a+2b)$, with values for input nodes $a = 3$, $b = 1$, $c = -2$, showing the forward pass computation of L.

From J&M, 3.ed., 2019

- A convenient tool for describing composite functions
- And follow the partial derivatives backwards
- There are tools that let us specify the computations at an high-level as graphs
- In particular useful for "hiding" vectors, matrices, tensors

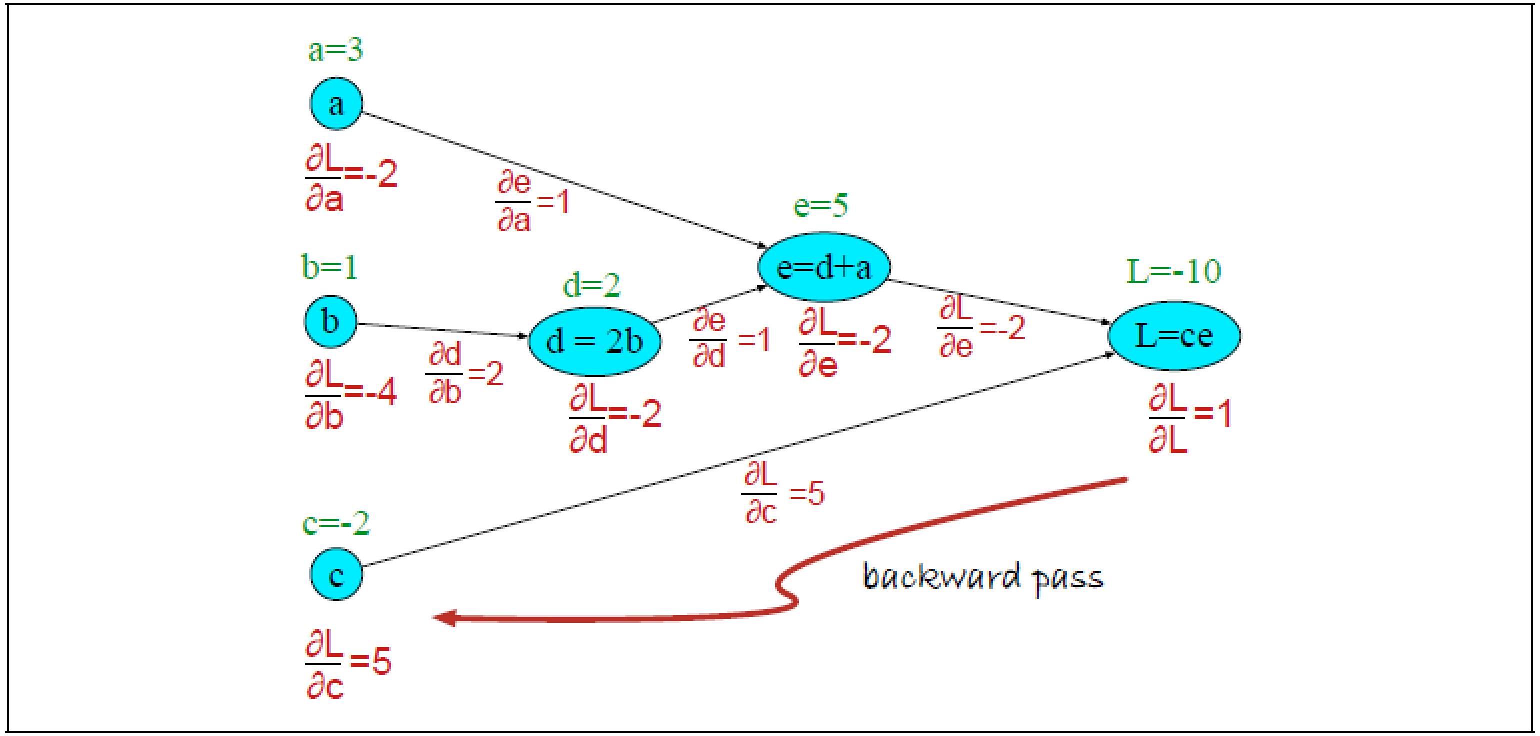


Figure 7.10 Computation graph for the function $L(a,b,c) = c(a+2b)$, showing the backward pass computation of $\frac{\partial L}{\partial a}$, $\frac{\partial L}{\partial b}$, and $\frac{\partial L}{\partial c}$.

From J&M, 3.ed., 2019

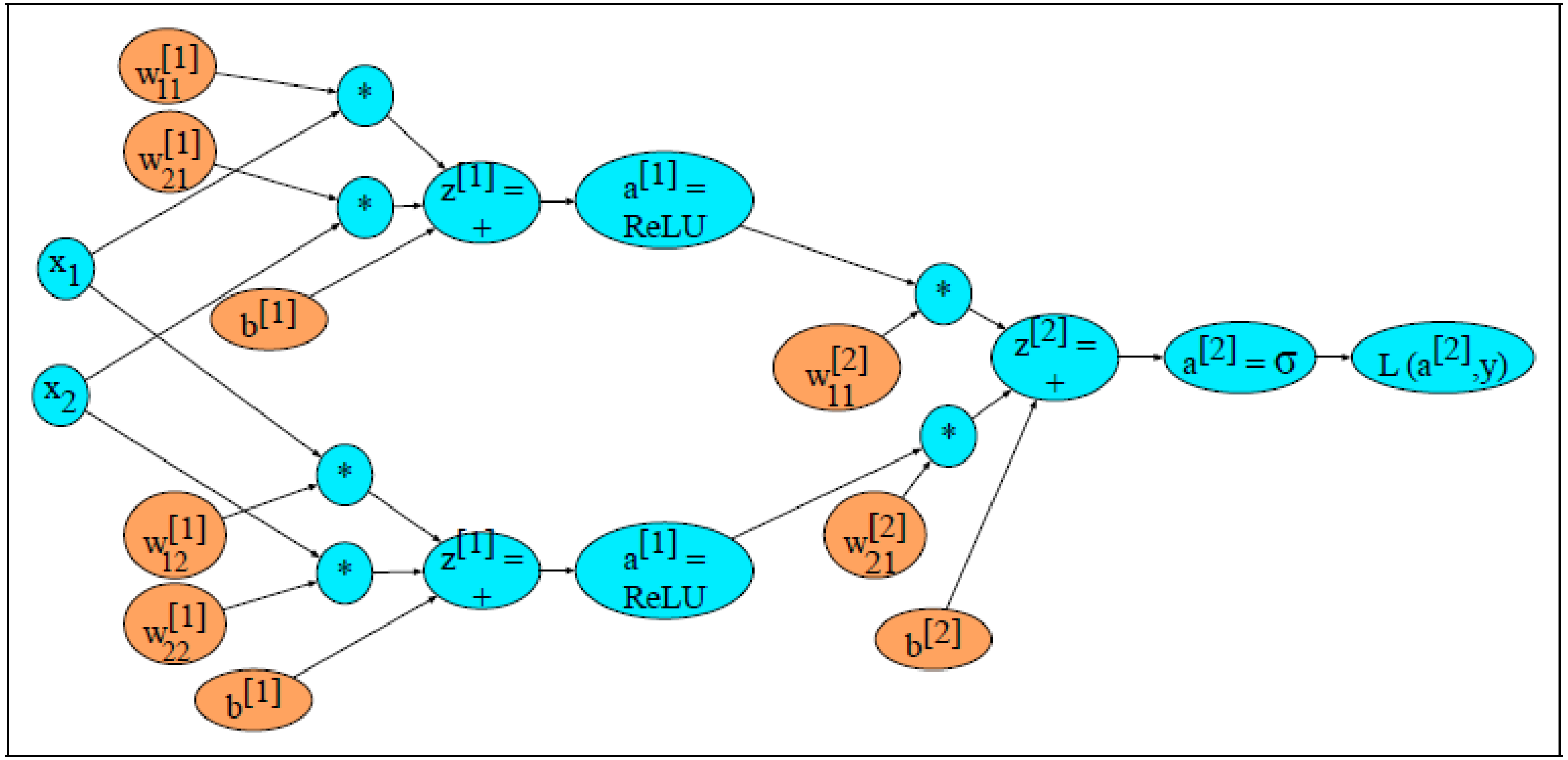


Figure 7.11 Sample computation graph for a simple 2-layer neural net (= 1 hidden layer) with two input dimensions and 2 hidden dimensions.

From J&M, 3.ed., 2019

Unfortunately: Many mistakes in the indices in the drawing

How would you draw this if x has dim 100,000 and there are 3 million parameters (weights)?

Using vector notation

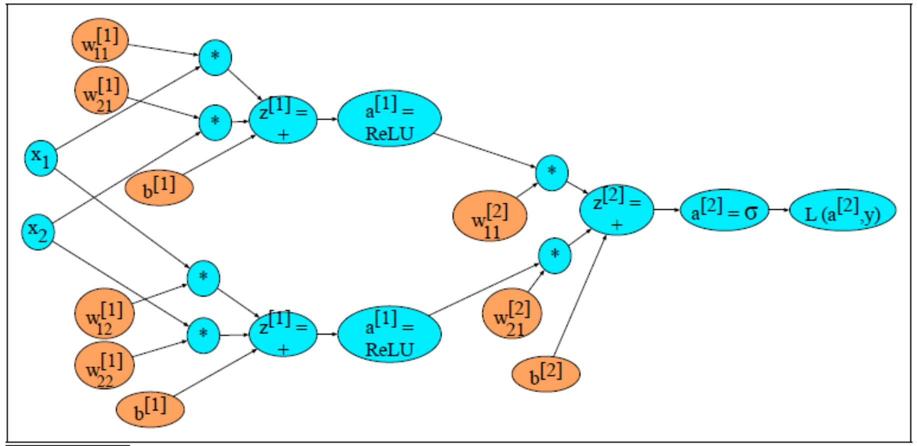
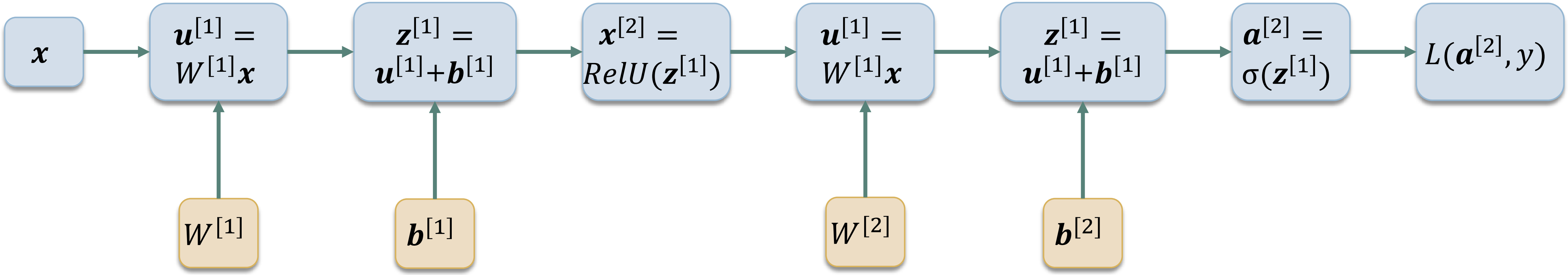


Figure 7.11 Sample computation graph for a simple 2-layer neural net (= 1 hidden layer) with two input dimensions and 2 hidden dimensions.



Or even simpler notation

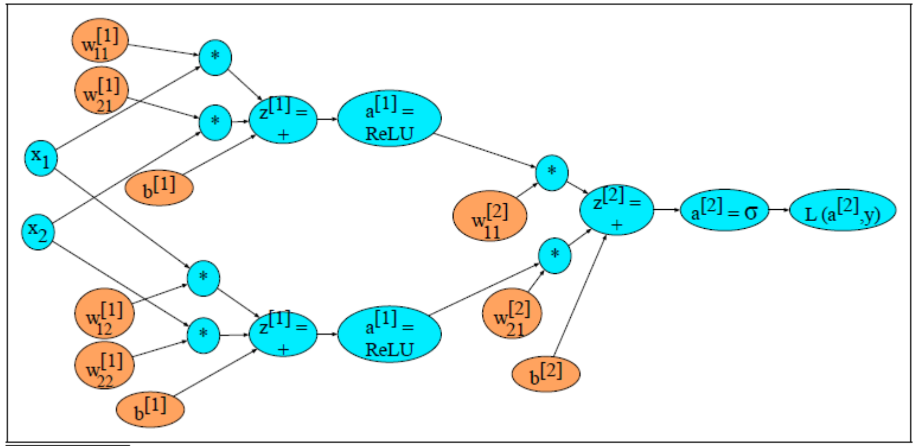
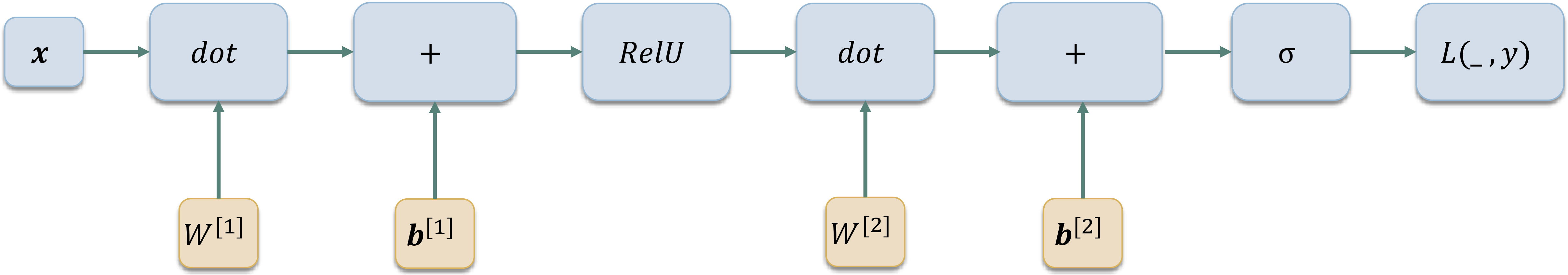


Figure 7.11 Sample computation graph for a simple 2-layer neural net (= 1 hidden layer) with two input dimensions and 2 hidden dimensions.



Details on training

- First round
  - Start with random weights.
  - Train the network.
  - Test on dev data
- Repeat:
  - You get a different result
  - Why?
- Solution:
  - Run several rounds
  - Repeat
  - Report mean and st.dev.
- There are many hyper-parameters that may be tuned
  - Example: embeddings
    - Context window size
    - Dimensions
    - "Drop-out"
- Drop-out
  - A way of regularization
  - Disregard som features during training
  - Different features for each round of training

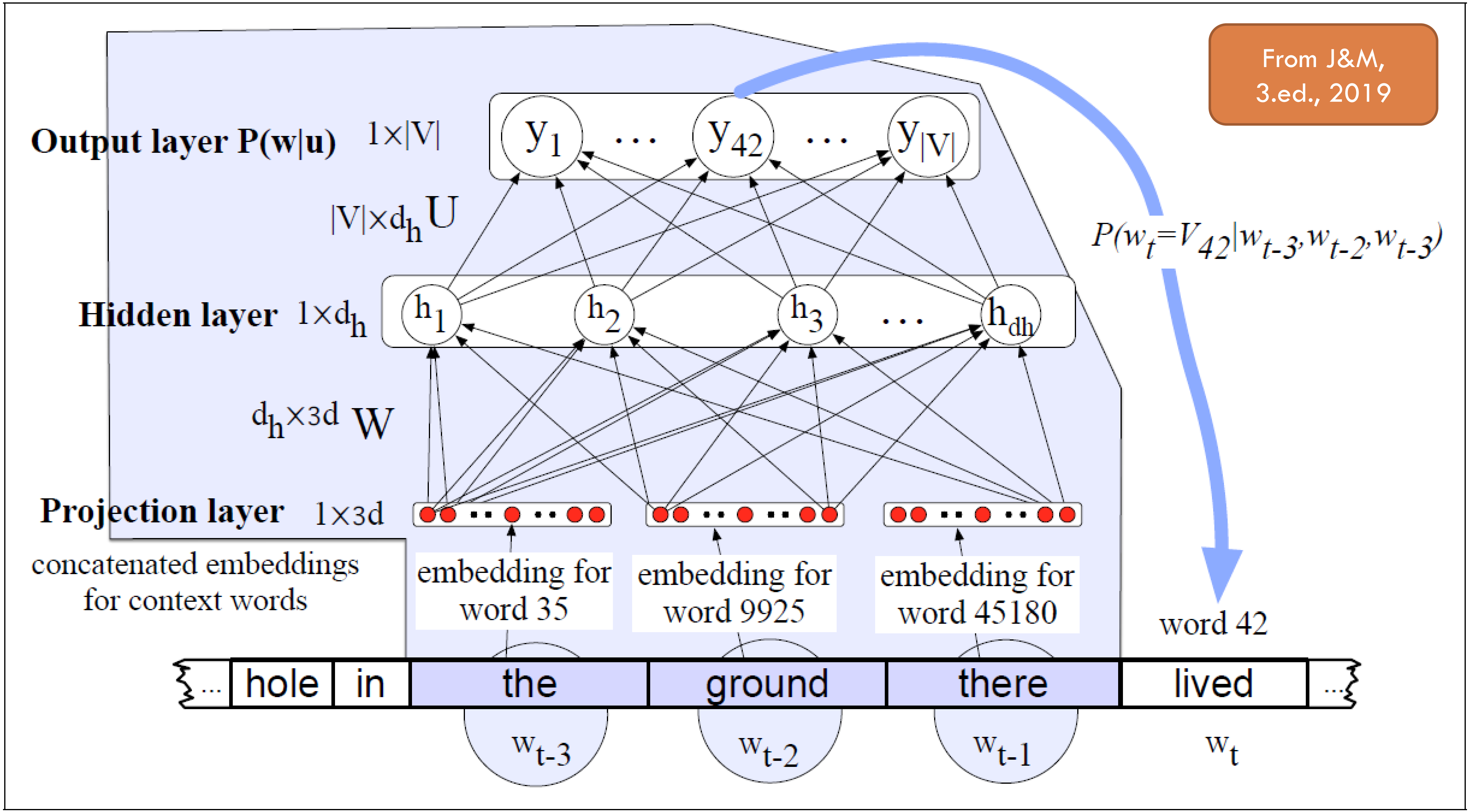
Today

- Feedforward neural networks (partly recap)
  - Model
  - Training
  - Computational graphs
  - Neural Language Models
- Recurrent networks
- Information Extraction

Neural NLP

- (Multi-layered) neural networks
- Using embeddings as word representations

- Example: Neural language model (k-gram)
  - $P\left(w_i \mid w_{i-k}^{i-1}\right)$
- Use embeddings for representing the w_i-s
- Use neural network for estimating $P\left(w_i \mid w_{i-k}^{i-1}\right)$

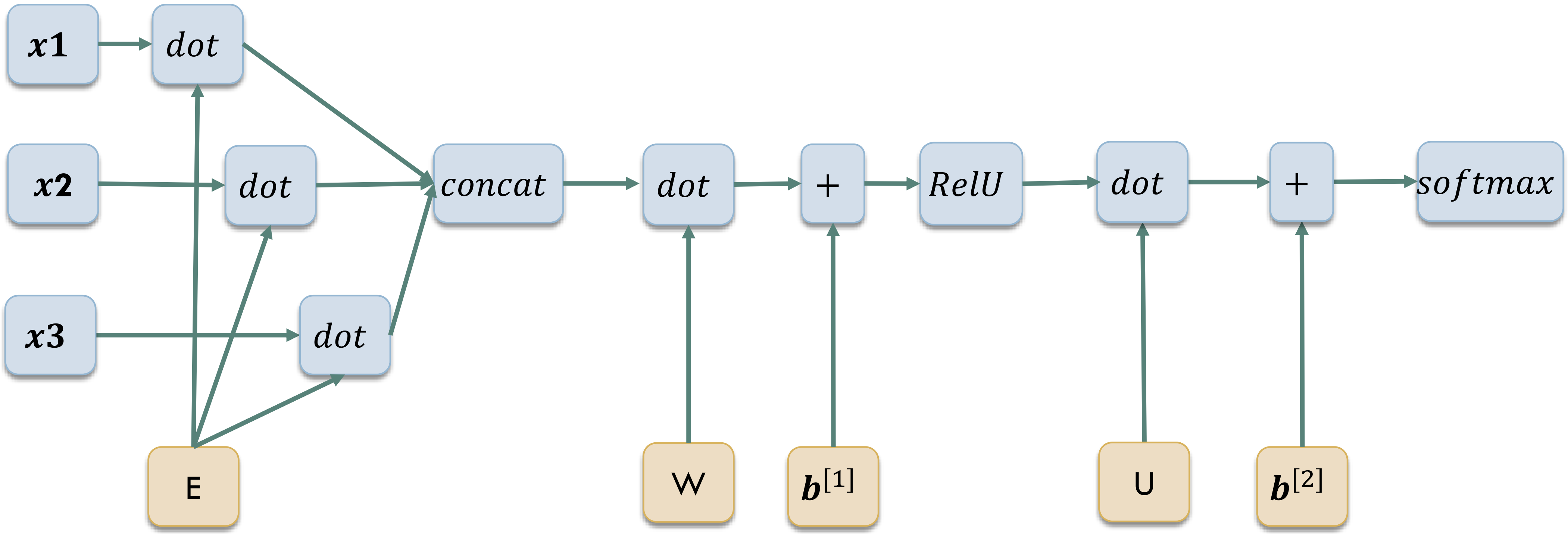


From J&M, 3.ed., 2019
Output layer P(w|u)
1×|V|
y_1 … y_{42} … $y_{|V|}$
$|V|\times d_h$ U
$P(w_t=V_{42}|w_{t-3},w_{t-2},w_{t-3})$
Hidden layer $1\times d_h$
h_1 h_2 h_3 … h_{dh}
$d_h\times 3d$ W
Projection layer 1×3d
concatenated embeddings for context words
embedding for word 35
embedding for word 9925
embedding for word 45180
word 42
… hole in the ground there lived …
w_{t-3} w_{t-2} w_{t-1} w_t

Pretrained embeddings

- The last slide uses pretrained embeddings
  - Trained with some method, SkipGram, CBOW, Glove, …
  - On some specific corpus
  - Can be downloaded from the web
- Pretrained embeddings can aslo be the input to other tasks, e.g. text classification
- The task of neural language modeling was also the basis for training the embeddings

Or simpler notation



Training the embeddings

- Alternatively we may start with one-hot representations of words and train the embeddings as the first layer in our models (=the way we trained the embeddings)
- If the goal is a task different from language modeling, this may result in embeddings better suited for the specific tasks.
- We may even use two set of embeddings for each word – one pretrained and one which is trained during the task.

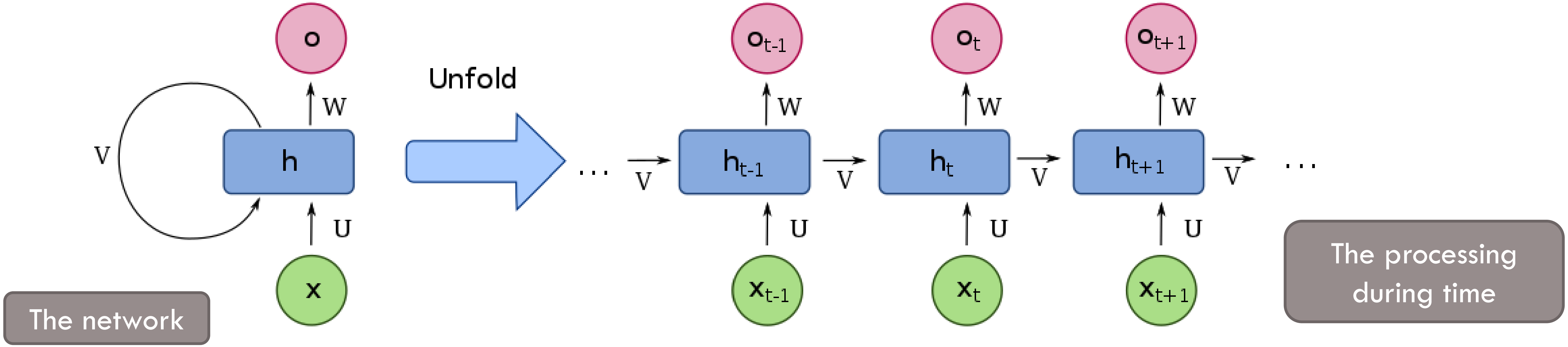
Recurrent networks

Today

- Feedforward neural networks
- Recurrent networks
  - Model
  - Language Model
  - Sequence Labeling
- Information Extraction

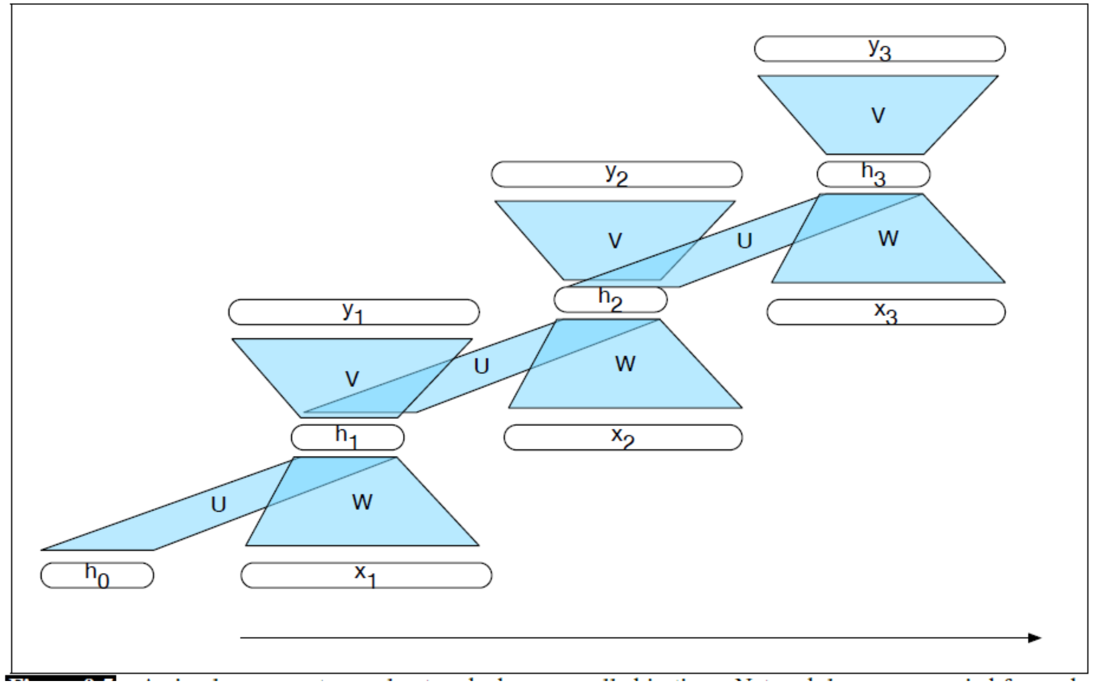
Recurrent neural nets

- Model sequences/temporal phenomena
- A cell may send a signal back to itself – at the next moment in time



https://en.wikipedia.org/wiki/Recurrent_neural_network

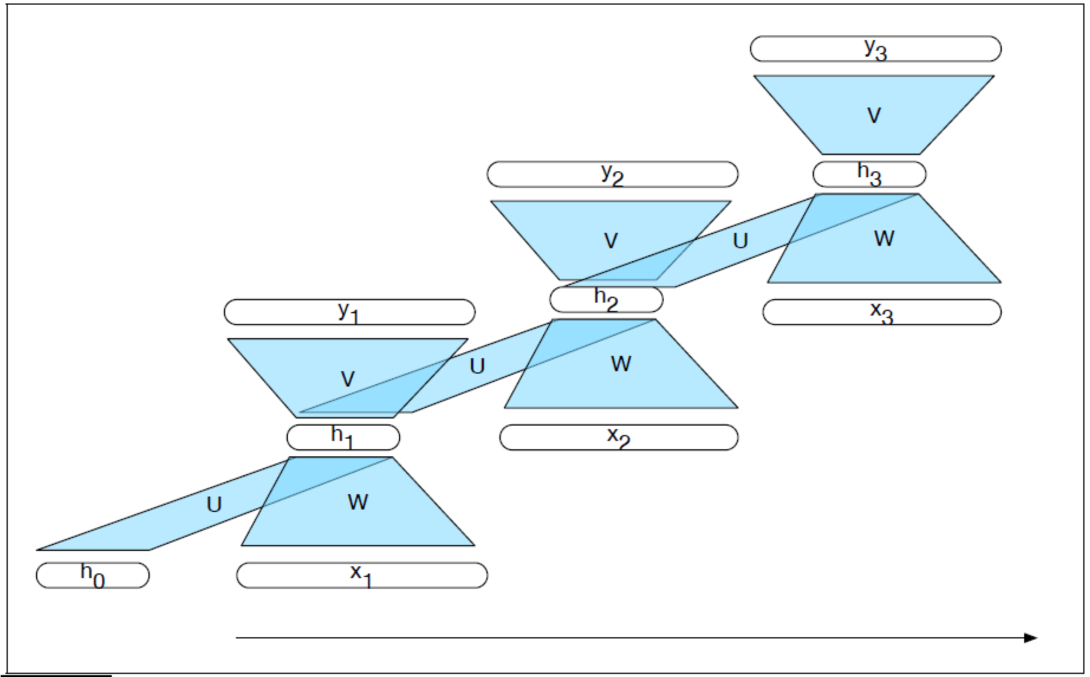
Forward



From J&M, 3.ed., 2019

- Each U, V and W are edges with weights
- $x_1, x_2, \dots, x_n$ is the input sequence
- Forward:
  1. Calculate h_1 from h_0 and .
  2. Calvculate y_1 from h_1.
  3. Calculate h_i from h_{i-1} and x_i, and y_i from i, for $i = 1, \dots, n$

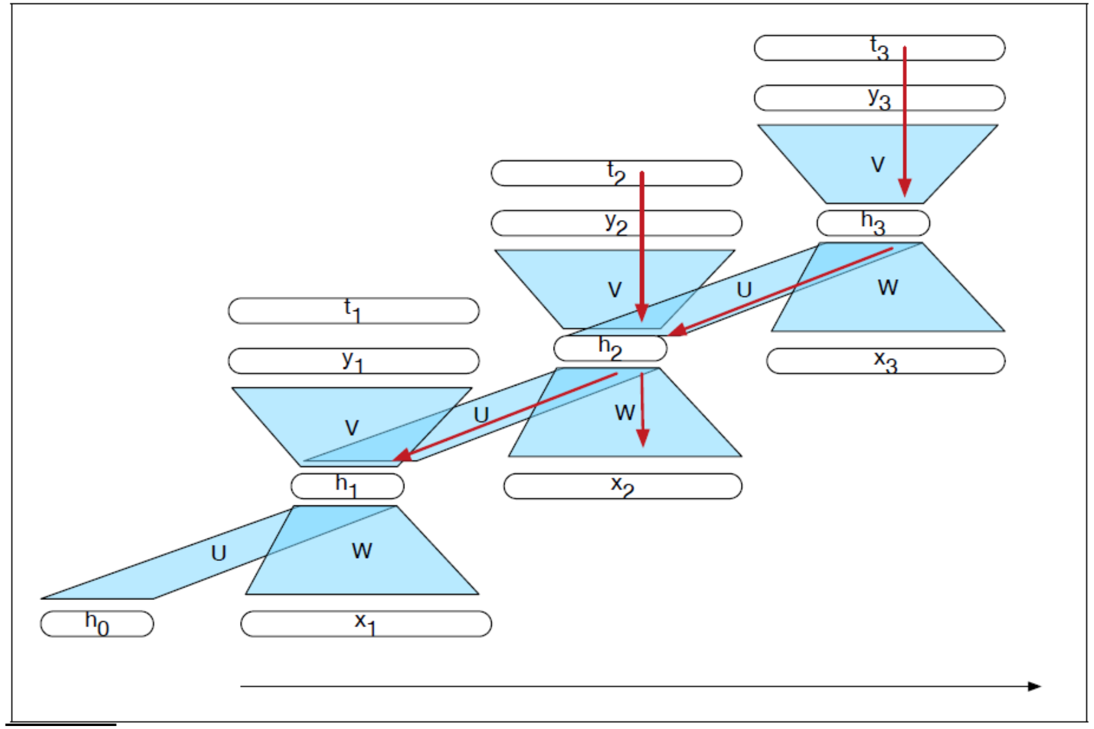
Forward



From J&M, 3.ed., 2019

- $\boldsymbol{h}_t = g(U\boldsymbol{h}_{t-1} + W\boldsymbol{x}_t)$
- $\boldsymbol{y}_t = f(V\boldsymbol{h}_t)$

Training



From J&M, 3.ed., 2019

- At each output node:
  - Calculate the loss and the
  - δ-term
- Backpropagate the error, e.g.
  - the δ-term at $h_2$is calculated
    - from the δ-term at h_3 by U and
    - the δ-term at y_2 by V
- Update
  - V from the δ-terms at the y_i-s and
  - U and W from the δ-terms at the h_i-s

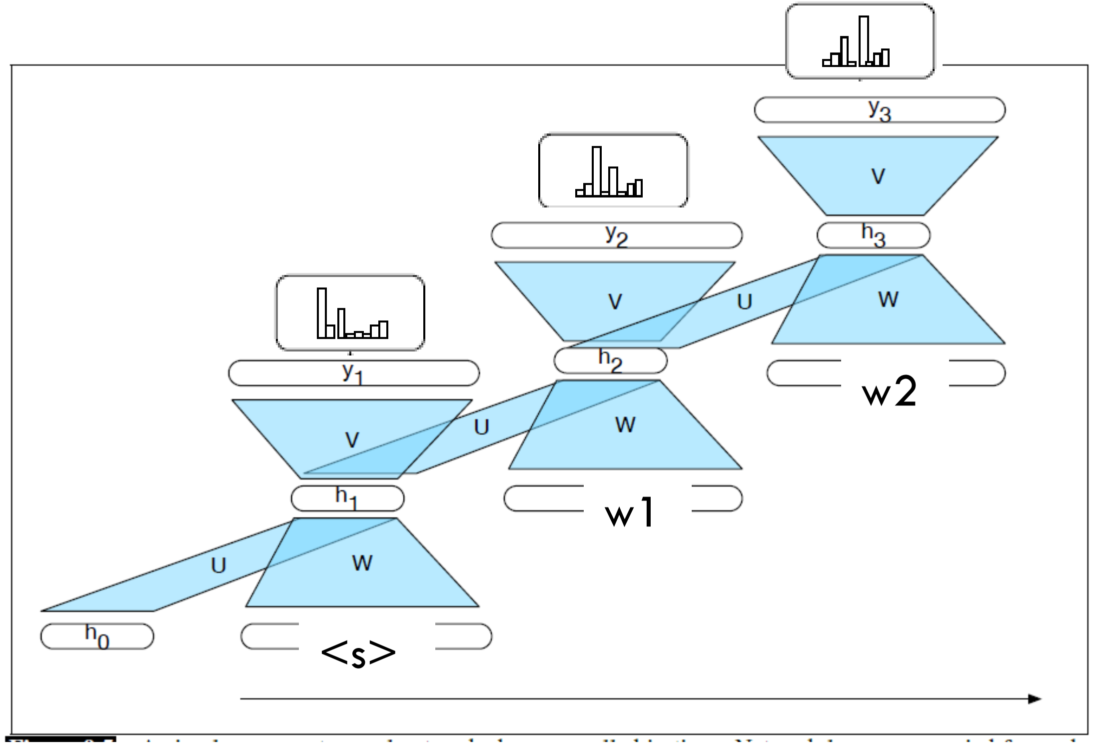
Remark

- J&M, 3. ed., 2019, sec 9.1.2 explain this at a high-level using vectors and matrices, OK
- The formulas, however, are not correct:
  - Describing derivatives of matrices and vectors demand a little more care, e.g. one has to transpose matrices
- It is beyond this course to explain how this can be done in detail
- But you should be able to do the actual calculations if you stick to the entries of the vectors and matrices, as we did above (ch. 7).

Today

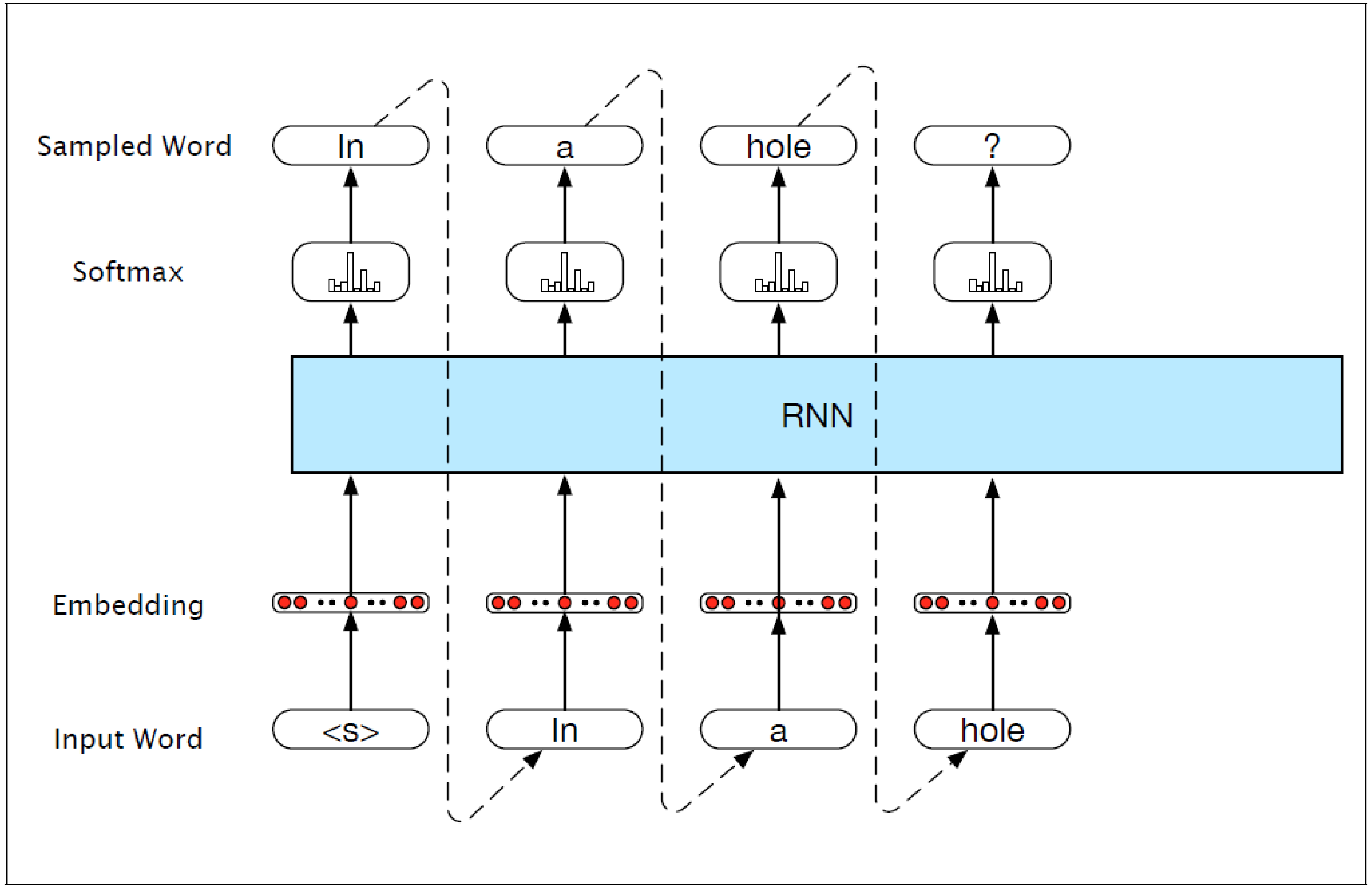
- Feedforward neural networks
- Recurrent networks
  - Model
  - Language Model
  - Sequence Labeling
- Information Extraction

RNN Language model



- $\hat{y} = P(w_n | w_1^{n-1}) = softmax(V\boldsymbol{h}_n)$
- In principle:
  - unlimited history
  - a word depends on all preceding words
- The word w_i is represented by an embedding
  - or a one-hot and the embedding is made by the LM

Autoregressive generation



From J&M, 3.ed., 2019

- Generated by probabilities:
  - Choose word in accordance with prob.distribution
- Part of more complex models
  - Encoder-decoder models
    - Translation

Today

- Feedforward neural networks
- Recurrent networks
  - Model
  - Language Model
  - Sequence Labeling
- Information Extraction

Neural sequence labeling: tagging

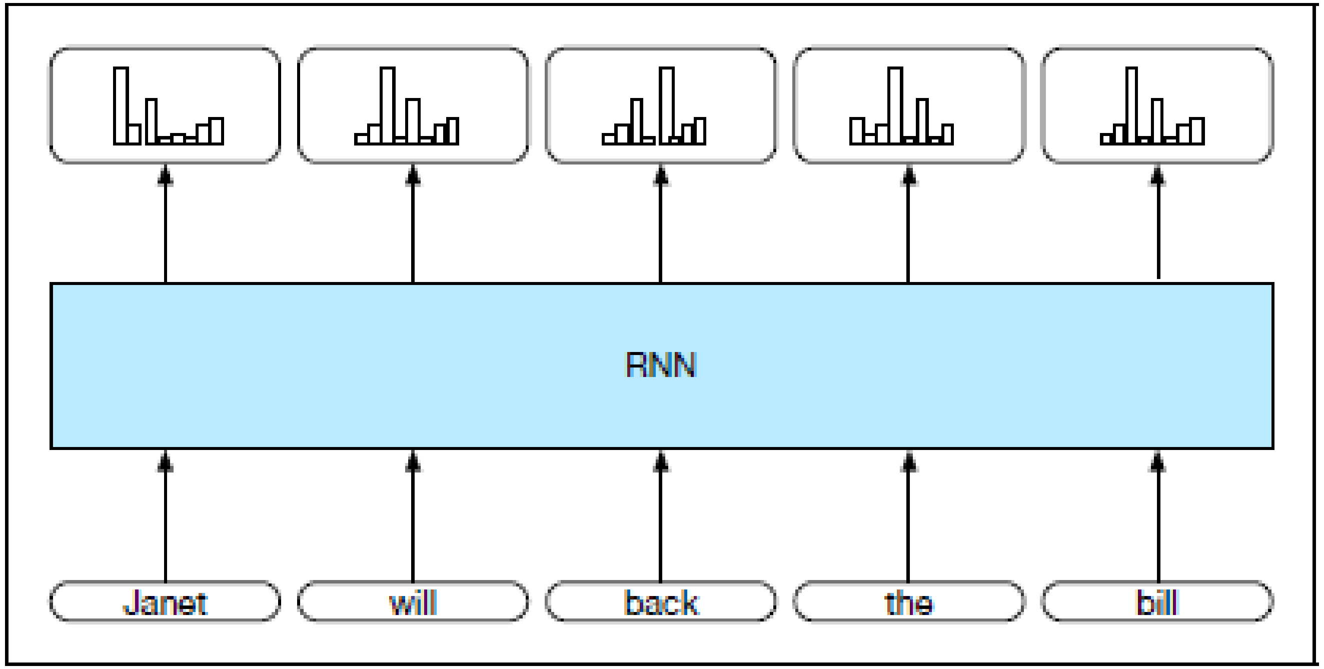


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

From J&M, 3.ed., 2019

Sequence labeling

- Actual models for sequence labeling, e.g. tagging, are more complex
- For example, that it may take words after the tag into consideration.

Information extraction

Today

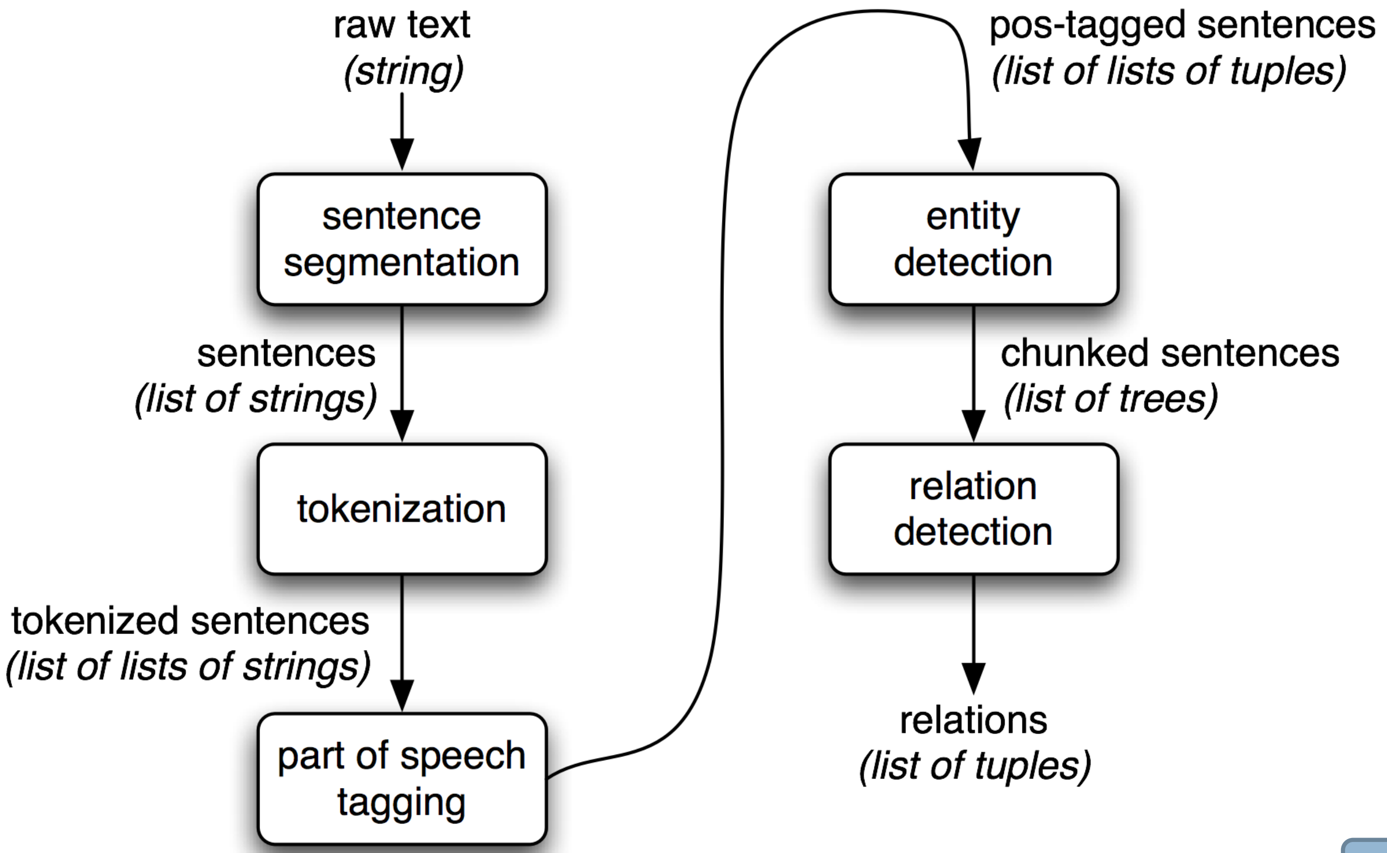
- Feedforward neural networks (partly recap)
- Recurrent networks
- Information extraction, IE
  - Chunking
  - Named entity recognition
  - Next week: Relation extraction

IE basics

Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. (Wikipedia)

- Bottom-Up approach
- Start with unrestricted texts, and do the best you can
- The approach was in particular developed by the Message Understanding Conferences (MUC) in the 1990s
- Select a particular domain and task

Steps



(Some approaches do these steps in a different order – or simultaneously)

From NLTK

Some example systems

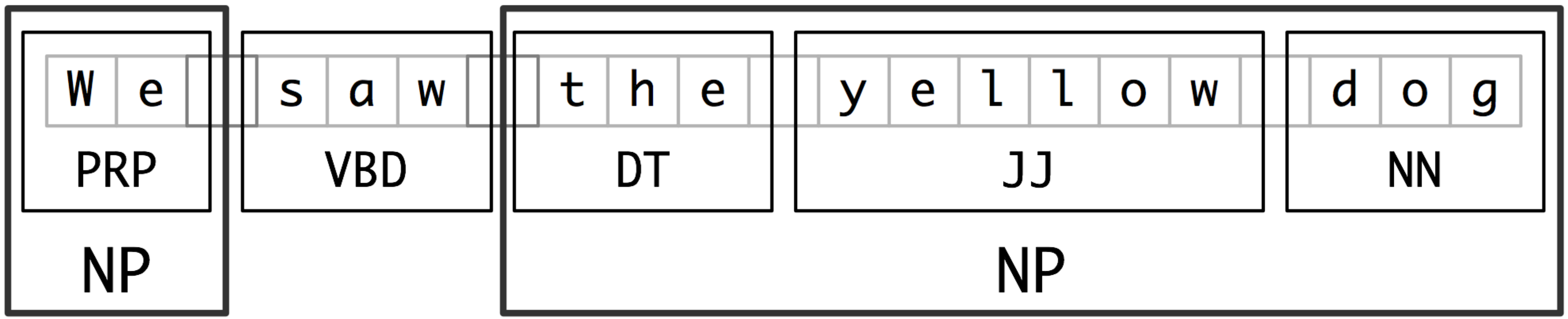
- Stanford core nlp: http://corenlp.run/
- SpaCy (Python): https://spacy.io/docs/api/
- OpenNLP (Java): https://opennlp.apache.org/docs/
- GATE (Java): https://gate.ac.uk/

- UDPipe: http://ufal.mff.cuni.cz/udpipe
  - Online demo: http://lindat.mff.cuni.cz/services/udpipe/

Today

- Feedforward neural networks (partly recap)
- Recurrent networks
- Information extraction, IE
  - Chunking
  - Named entity recognition
  - Next week: Relation extraction

Next steps



- Chunk together words to phrases

NP-chunks

[The/DT market/NN] for/IN
[system-management/NN software/NN] for/IN
[Digital/NNP]
['s/POS hardware/NN] is/VBZ fragmented/JJ enough/RB that/IN
[a/DT giant/NN] such/JJ as/IN
[Computer/NNP Associates/NNPS] should/MD do/VB well/RB there/RB ./.

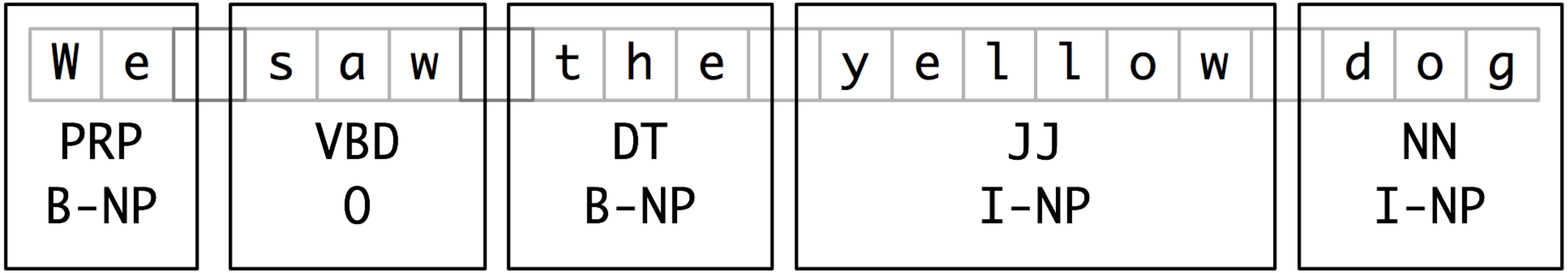
- Exactly what is an NP-chunk?
- It is an NP
- But not all NPs are chunks
- Flat structure: no NP-chunk is part of another NP chunk
- Maximally large
- Opposing restrictions

Regular Expression Chunker

- Input POS-tagged sentences
- Use a regular expression over POS to identify NP-chunks
- NLTK example:
- It inserts parentheses

```
grammar = r"""
  NP: {<DT|PP\$>?<JJ>*<NN>}
      {<NNP>+}
"""
```

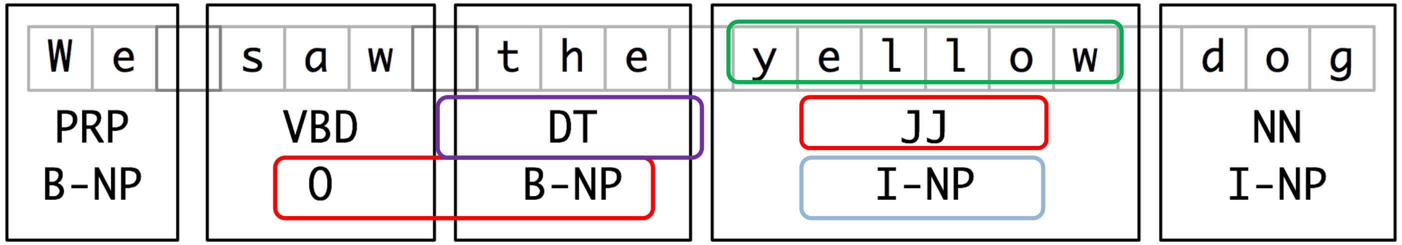
IOB-tags



- B-NP: First word in NP
- I-NP: Part of NP, not first word
- O: Not part of NP (phrase)

- Properties
  - One tag per token
  - Unambiguous
  - Does not insert anything in the text itself

Assigning IOB-tags



- The process can be considered a form for tagging
  - POS-tagging: Word to POS-tag
  - IOB-tagging: POS-tag to IOB-tag
- But one may in addition use additional features, e.g. words
- Can use various types of classifiers
  - NLTK uses a MaxEnt Classifier (=LogReg, but the implementation is slow)
  - We can modify along the lines of mandatory assignment 2, using scikit-learn

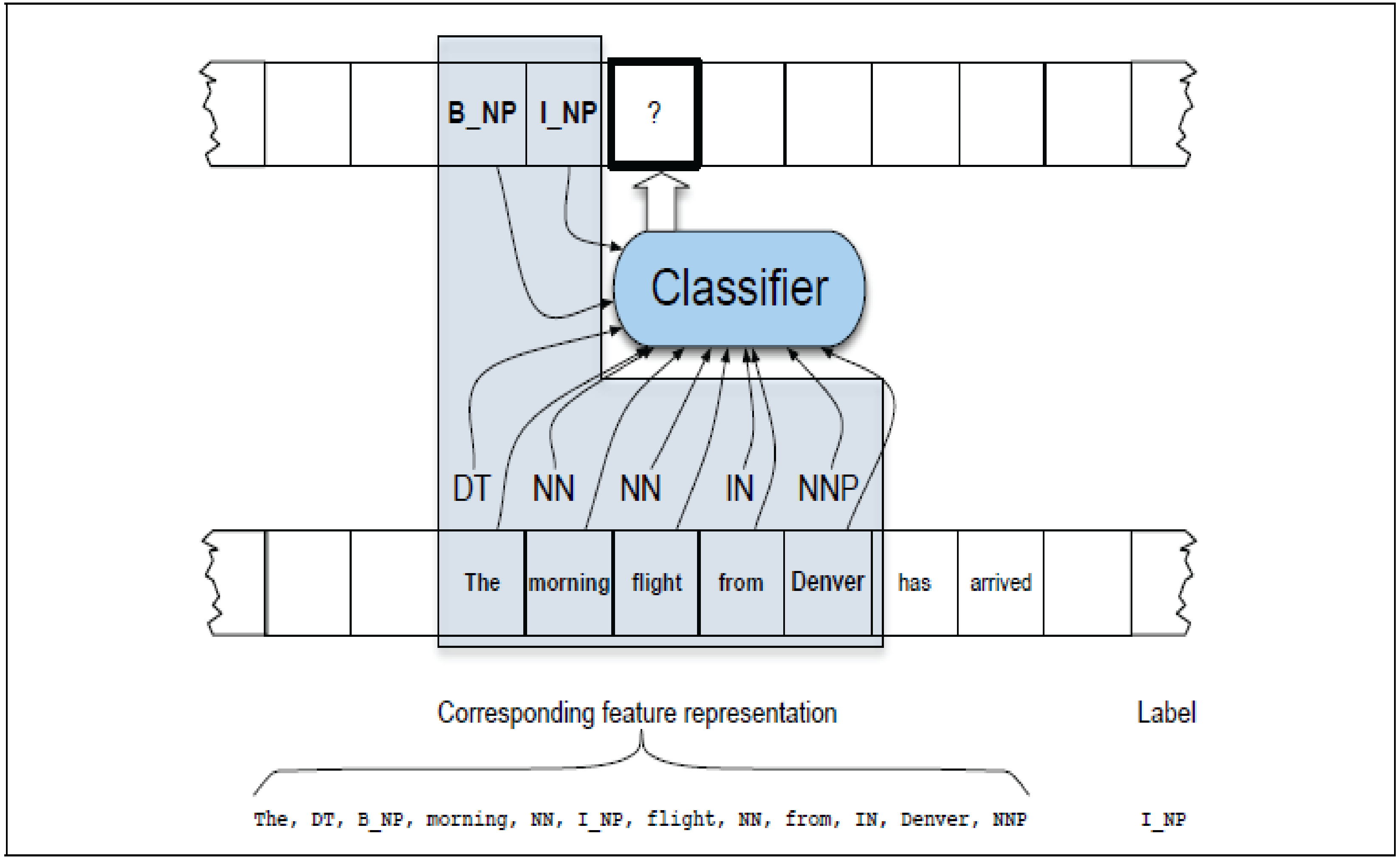


Figure 11.8 A sequence model for chunking. The chunker slides a context window over the sentence, classifying words as it proceeds. At this point, the classifier is attempting to label *flight*, using features like words, embeddings, part-of-speech tags and previously assigned chunk tags.

J&M, 3. ed.

Evaluating (IOB-)chunkers

- cp = nltk.RegexpParser("")
- test_sents = conll ('test', chunks=['NP'])
- IOB Accuracy: 43.4%
- Precision: 0.0%
- Recall: 0.0%
- F-Measure: 0.0%

- What do we evaluate?
  - IOB-tags? or
  - Whole chunks?
  - Yields different results
- For IOB-tags:
  - Baseline:
    - majority class O,
    - yields > 33%
- Whole chunks:
  - Which chunks did we find?
  - Harder
  - Lower numbers

Evaluating (IOB-)chunkers

- cp = nltk.RegexpParser("")
- test_sents = conll ('test', chunks=['NP'])
- IOB Accuracy: 43.4%
- Precision: 0.0%
- Recall: 0.0%
- F-Measure: 0.0%

>> cp = nltk.RegexpParser(r"NP: {<[CDJNP].*>+}")

- IOB Accuracy: 87.7%
- Precision: 70.6%
- Recall: 67.8%
- F-Measure: 69.2%

Today

- Feedforward neural networks (partly recap)
- Recurrent networks
- Information extraction, IE
  - Chunking
  - Named entity recognition
  - Next week: Relation extraction, 5 different ways

Named entities

Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY $6] per round trip on flights to some cities also served by lower-cost carriers. [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said. [ORG United], a unit of [ORG UAL Corp.], said the increase took effect [TIME Thursday] and applies to most routes where it competes against discount carriers, such as [LOC Chicago] to [LOC Dallas] and [LOC Denver] to [LOC San Francisco].

- Named entity:
  - Anything you can refer to by a proper name
  - i.e. not all NP (chunks):
    - *high fuel prices*
  - Maybe longer NP than just chunk:
    - *Bank of America*
- Find the phrases
- Classify them

Types of NE

Type	Tag	Sample Categories
People	PER	Individuals, fictional characters, small groups
Organization	ORG	Companies, agencies, political parties, religious groups, sports teams
Location	LOC	Physical extents, mountains, lakes, seas
Geo-Political Entity	GPE	Countries, states, provinces, counties
Facility	FAC	Bridges, buildings, airports
Vehicles	VEH	Planes, trains, and automobiles

- The set of types vary between different systems
- Which classes are useful depend on application

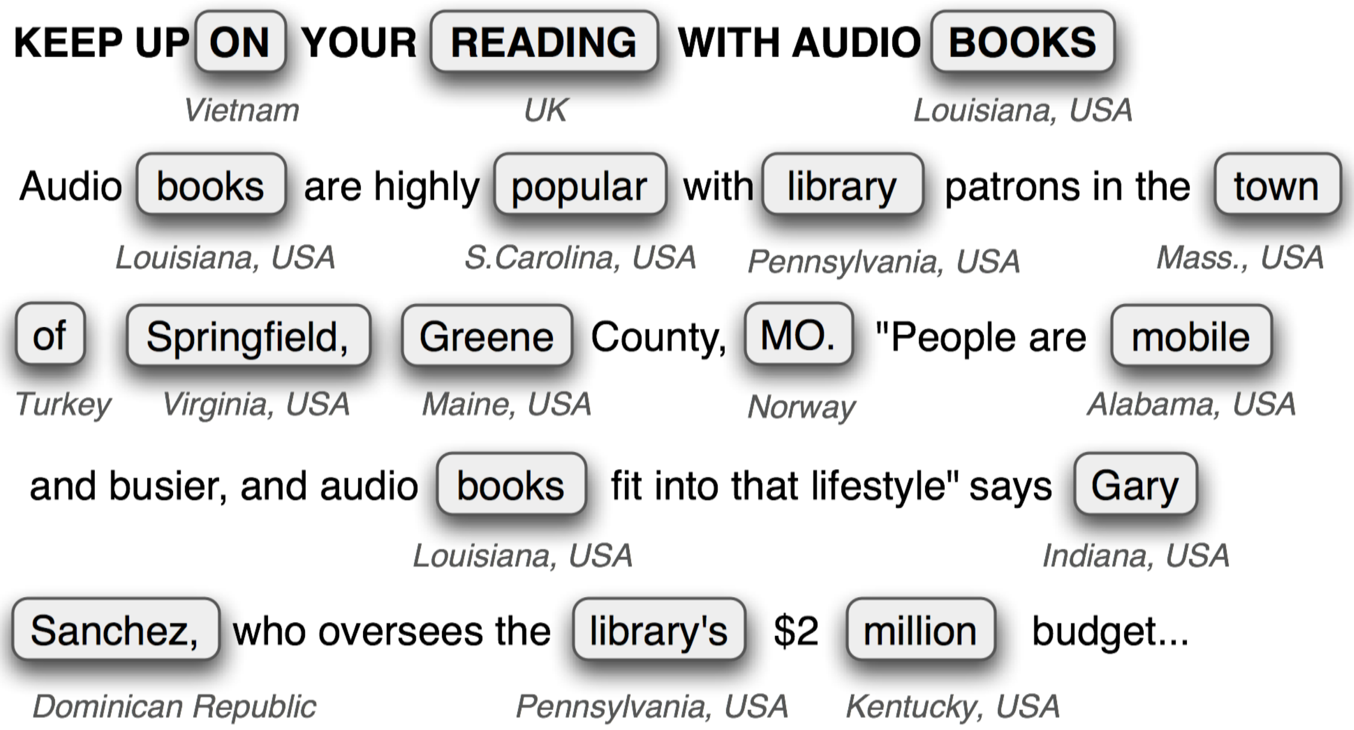
Ambiguities

Name	Possible Categories
Washington	Person, Location, Political Entity, Organization, Facility
Downing St.	Location, Organization
IRA	Person, Organization, Monetary Instrument
Louis Vuitton	Person, Organization, Commercial Product

[$_{PERS}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{ORG}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{LOC}$ Washington] for what may well be his last state visit.
In June, [$_{GPE}$ Washington] passed a primary seatbelt law.
The [$_{FAC}$ Washington] had proved to be a leaky ship, every passage I made...

Gazetteer

- Useful: List of names, e.g.
  - Gazetteer: list of geographical names
- But does not remove all ambiguities
  - cf. example

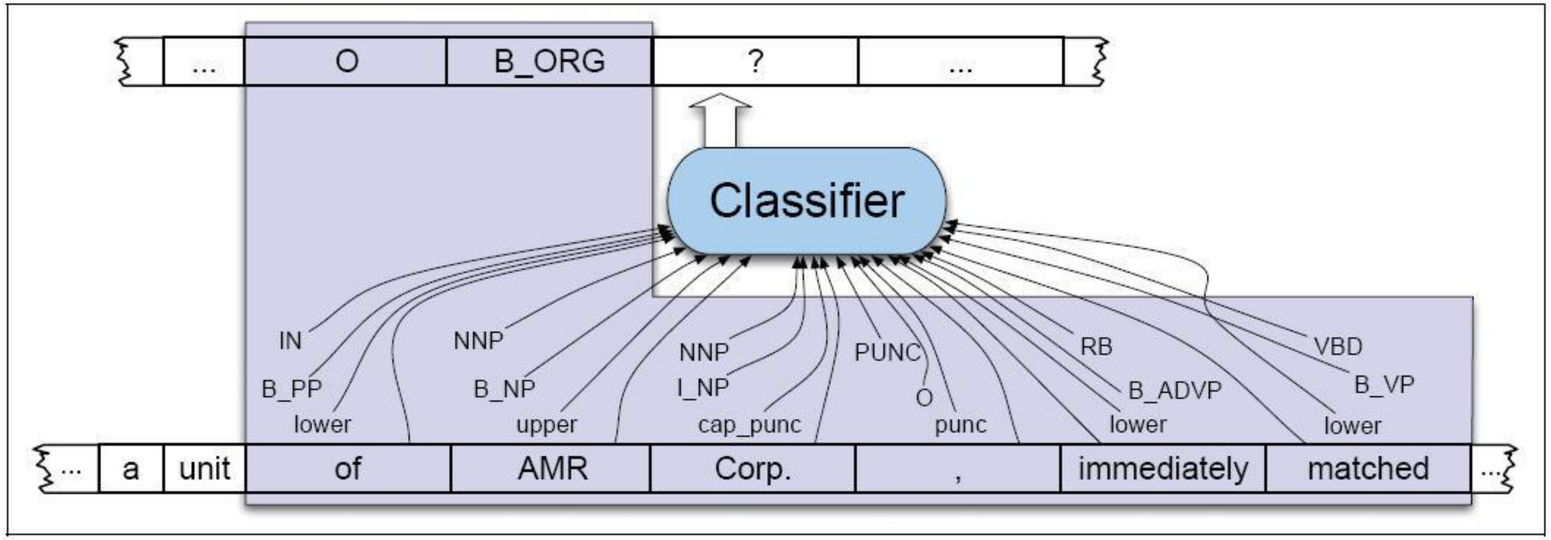


Representation (IOB)

Words	IOB Label	IO Label
American	B-ORG	I-ORG
Airlines	I-ORG	I-ORG
,	O	O
a	O	O
unit	O	O
of	O	O
AMR	B-ORG	I-ORG
Corp.	I-ORG	I-ORG
,	O	O
immediately	O	O
matched	O	O
the	O	O
move	O	O
,	O	O
spokesman	O	O
Tim	B-PER	I-PER
Wagner	I-PER	I-PER
said	O	O
.	O	O

Figure 17.4 Named entity tagging as a sequence model, showing IOB and IO encodings.

Feature-based NER



- Similar to tagging and chunking
- You will need features from several layers
- Features may include
  - Words, POS-tags, Chunk-tags, Graphical prop.
  - and more (See J&M, 3.ed)

Neural sequence labeling: NER

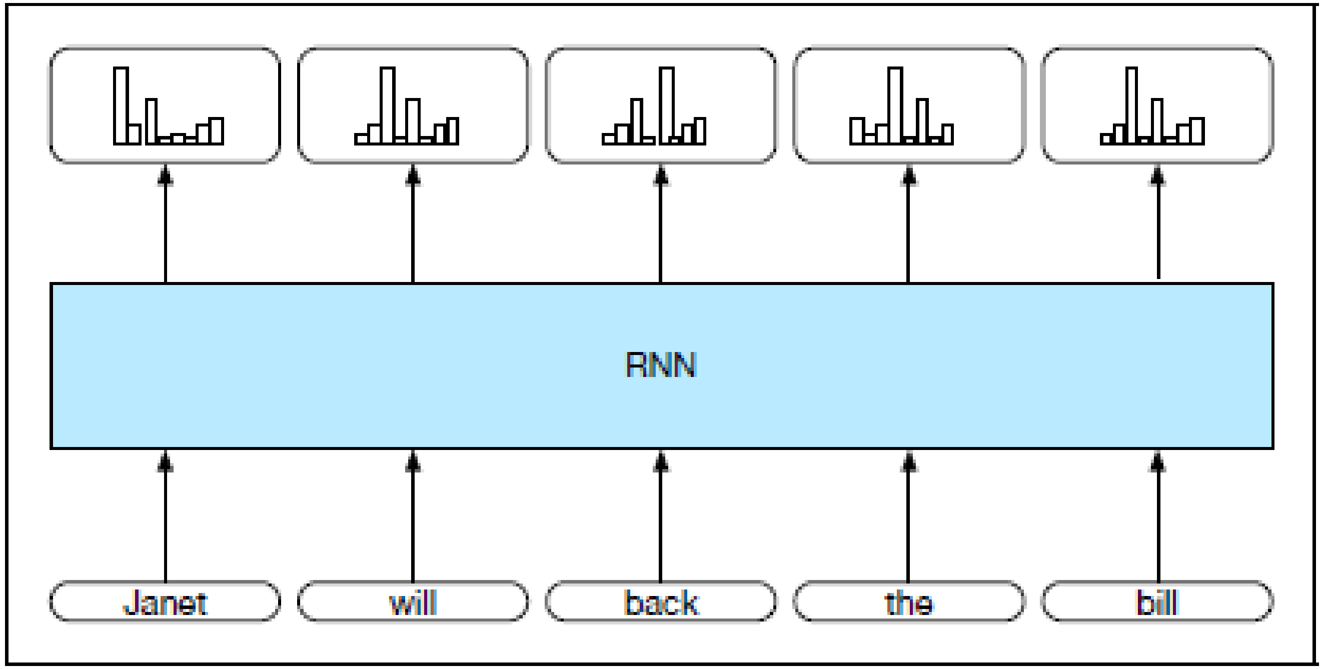


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

From J&M, 3.ed., 2019

- We can use IOB-tags
- RNN
- Similarly to POS-tagging

Evaluation

- Have we found the correct NERs?
  - Evaluate precision and recall as for chunking
- For the correctly identified NERs, have we labelled them correctly?

To be continued