VL Deep Learning for Natural Language Processing

11. Recurrent Neural Networks I

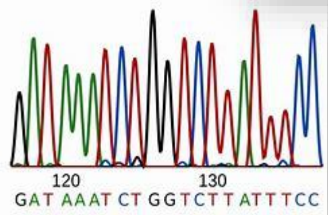
Prof. Dr. Ralf Krestel

AG Information Profiling and Retrieval

Sequencial Data

- So far: feed forward nets
  - Fully-conected layers
  - Every input data point independent
    - Input, e.g. a whole move review
  - No storing of states accross training samples
  - A sequence can only be processed as a whole not one-by-one

- Task with sequencial data
  - Speech recognition
  - Music generation
  - Sentiment classification
  - DNA analysis
  - Machine translation
  - Scene description

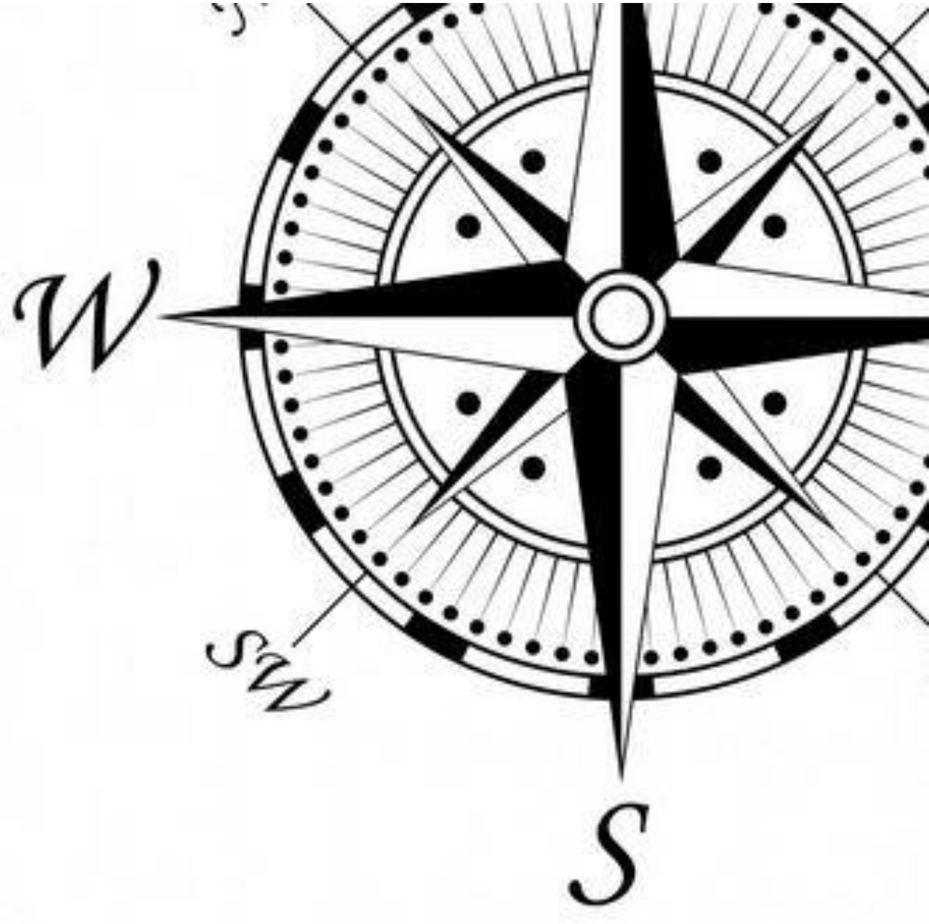


Lerning Goals for this Chapter

- Understand the difference between feed-forward-nets and recurrent network architectures
- Know application areas
- Adapt task description so that RNNs can be used to solve the problem
- Explain different kinds of RNNs and how they work
- Implement and evaluate a simple RNN model

- Relevant chapters:
  – P6.2, S5+S6 (2021)

Topics Today



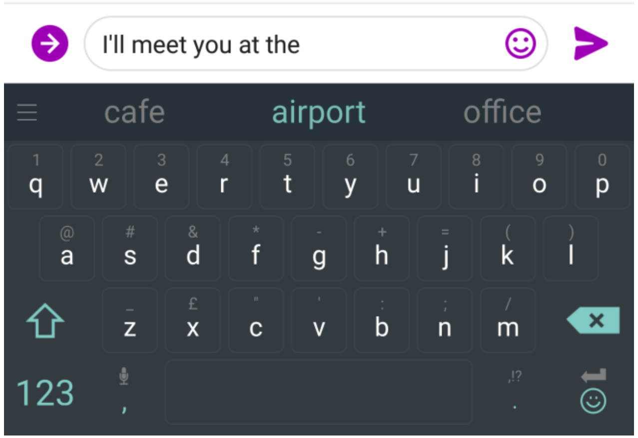
1. **Language Models**
2. A RNN Language Model
3. Recurrent Neural Networks (RNN)
4. Different RNN Types
5. A Simple RNN

Language Model Definition

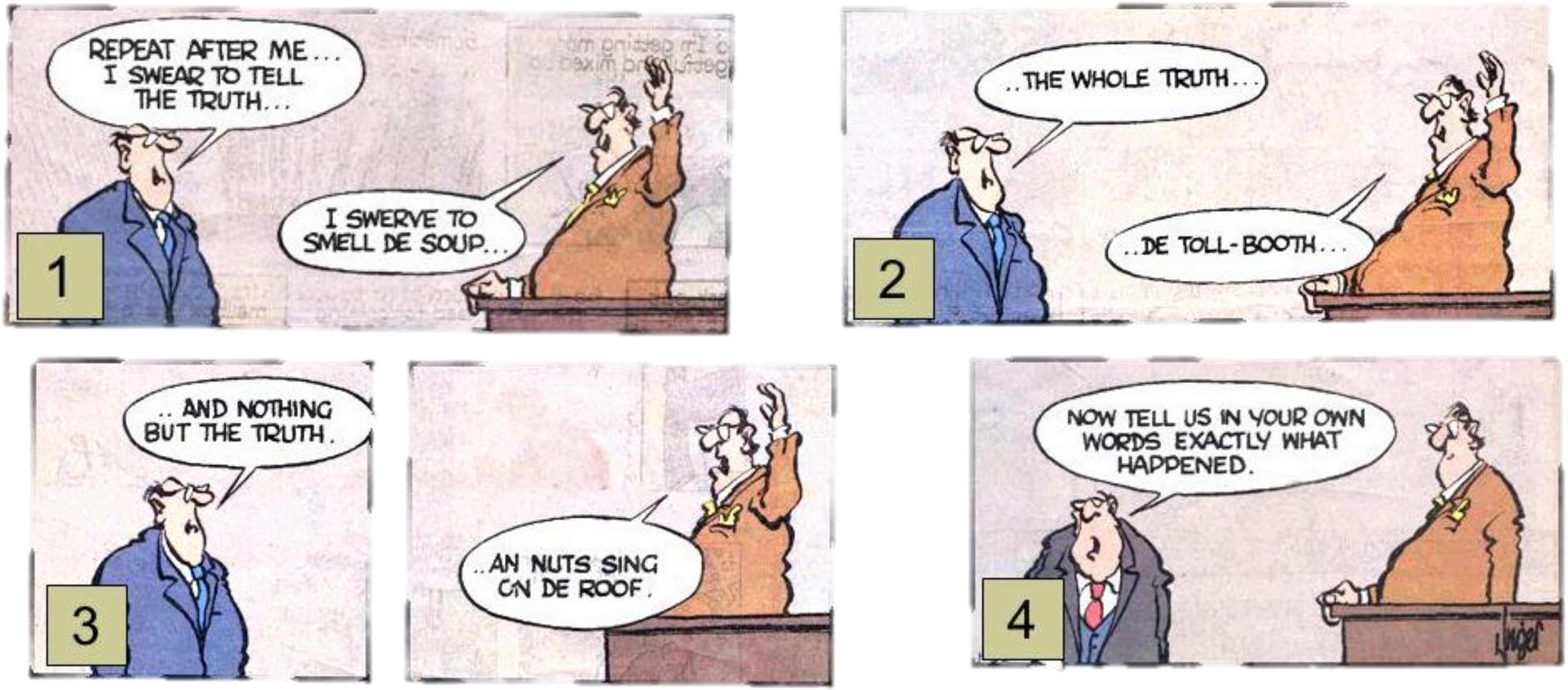
- The goal of **language modeling** is to model a language, i.e., build a model of a language.
- With a good model you can make predictions:
  - „In five minutes, I will go ________“
    - ❒ home
    - ❒ Berlin
    - ❒ out of town
    - ❒ supermarket
- Formal: Given a sequence of words $x^{(1)}, x^{(2)}, \dots, x^{(t)}$, compute a probability distribution over the next word $x^{(t+1)}$: $P\left(x^{(t+1)} = w_j \middle| x^{(t)}, \dots, x^{(1)}\right)$
  - where w_j is a word from vocabulary $V = \{w_1, \dots, w_{|V|}\}$
- „Language modeling“ denotes the task.
- A system which solves this task is called **a language model**.

Examples


Google
Deutschland
warum
Erweiterte Suche
Sprachoptionen
warum ist der himmel blau
warum liegt hier stroh
warum
warum ist die banane krumm
warum soll ein längeres starkes gefälle nicht mit getretener kupplung durchfahren werden
warum bin ich so fröhlich
warum kann das befahren dieser ungleichmäßig beleuchteten straße gefährlich werden
warum musste robert das bb haus verlassen
warum will thoas iphigenie heiraten
warum ist scharfes anfahren zu vermeiden
Google-Suche
Auf gut Glück!


I'll meet you at the
cafe
airport
office
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123

A Bad Language Model



N-Gram Language Model I

„In five minutes, I will go _______“

- How to learn a laguage model?
  - By counting **n-grams**!
- An n-gram denotes consequtive words
  - n=1: unigrams: „In“, „five“, „minutes“, „I“, „will“, „go“
  - n=2: bigrams: „In five“, „five minutes“, „minutes I“, …
  - n=3: trigrams: „In five minutes“, „five minutes I“, …
  - n=4: fourgrams: „In five minutes I“, „five minutes I will“, …

- Side note: There are also character n-grams
  - n=2: „_I“, „In“, „n “, „ f“, „fi“, „iv“, „ve“, „e “, …

N-Gram Language Model II

- Asumption:
$$P\left(x^{(t+1)} = w_j \middle| x^{(t)}, \dots, x^{(1)}\right) = P\left(x^{(t+1)} = w_j \middle| x^{(t)}, \dots, x^{(t-n+2)}\right)$$

- Conditional Probability:
$$P\left(x^{(t+1)} = w_j \middle| x^{(t)}, \dots, x^{(t-n+2)}\right) = \frac{P(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{P(x^{(t)}, \dots, x^{(t-n+2)})}$$

- Estimate the probabilities:
  - Large, represantative corpus
$$P\left(x^{(t+1)} = w_j \middle| x^{(t)}, \dots, x^{(1)}\right) \approx \frac{count(x^{(t+1)}, x^{(t)}, \dots, x^{(t-n+2)})}{count(x^{(t)}, \dots, x^{(t-n+2)})}$$

Example Four-Gram Language Model

„~~It is very cold outside~~, I will go ________“

$$P(w_j|i\ will\ go) = \frac{count(i\ will\ go\ w_j)}{count(i\ will\ go)}$$

- In the copus:
  - „i will go“ occurs 1000 times.
  - „i will go home“ occurs 400 times.
    - $P(home|i\ will\ go) = 0.4$
  - „I will go indoors“ occurs 10 times.
    - $P(indoors|i\ will\ go) = 0.01$
- Problem:
  - Context too small
    - But for any n>5 too sparse
    - Memory need increases exponentially with n ($O(\exp(n))$)

Smoothing: E.g. Laplace

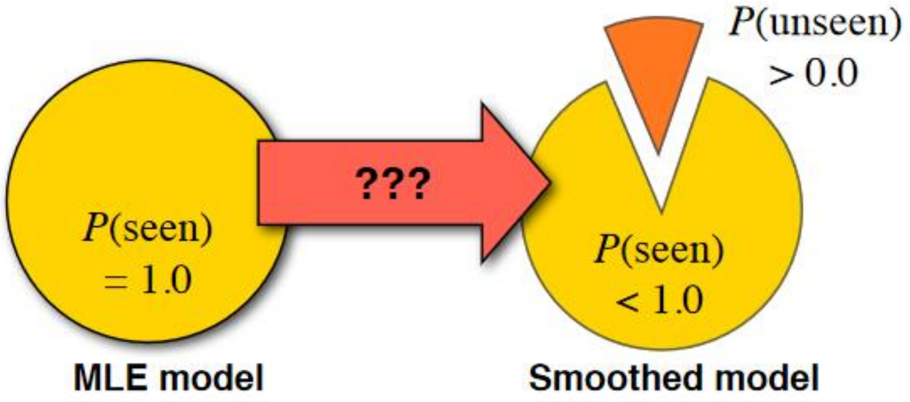
Backoff: E.g. Katz

Smoothing

- Maximum Likelihood Estimate (MLE)

$$\mathrm{P}(\mathrm{w_i}) = \frac{C(w_i)}{\sum_j C(w_j)} = \frac{C(w_i)}{N}$$

  - with N=count of all tokens
  - P(seen)=1



- Smoothing
  - Assign some probability to unseen n-grams
  - Laplace (add-1) smoothing

$$\mathrm{P}(\mathrm{w_i}) = \frac{C(w_i) + 1}{N + V}$$

Katz‘s Back-Off Model

- Idea
  - If count for n-gram is zero, take shorter n-gram instead
- Non-linear method
- The estimate for an n-gram is allowed to back off through progressively shorter histories.
- The most detailed model that can provide sufficiently reliable information about the current context is used.
- Trigram version (simplified):
  - if $C(w', w'', w) > 0$ $P^*(w \mid w', w'') = P(w \mid w', w'')$
  - else if $C(w'', w) > 0$ $P^*(w \mid w', w'') = P(w \mid w'')$
  - else if $C(w) > 0$ $P^*(w \mid w', w'') = P(w)$
  - else $P^*(w \mid w', w'') = 1 / \#words$

Katz‘s Back-Off Model Exmaple

- Smoothing of Conditional Probabilities

  P(Angeles | to, Los)

- If *„to Los Angeles“* is not in the training corpus, the smoothed probability P(Angeles | to, Los) is identical to P(York | to, Los).

- However, the actual probability is probably close to the bigram probability P(Angeles | Los).

Generative Language Model I

```
from nltk.corpus import reuters
from collections import Counter
counts = Counter(reuters.words())
total_count = len(reuters.words())
print counts.most_common(n=20)
# [(u'.', 94687), (u',', 72360), (u'the',
58251), (u'of', 35979), (u'to', 34035),
(u'in', 26478), (u'said', 25224), (u'and',
25043), (u'a', 23492), (u'mln', 18037),
(u'vs', 14120), (u'-', 13705), (u'for',
12785), (u'dlrs', 11730), (u"'", 11272),
(u'The', 10968), (u'000', 10277), (u'1',
9977), (u's', 9298), (u'pct', 9093)]
for word in counts:
    counts[word] /= float(total_count)
print sum(counts.values())
# 1.0
```

https://nlpforhackers.io/language-models/

```
import random
text = []
for _ in range(100):
    r = random.random()
    accumulator = .0
    for word, freq in counts.iteritems():
        accumulator += freq
        if accumulator >= r:
            text.append(word)
            break
print ' '.join(text)
# tax been its and industrial and vote "
decision rates elimination and 2 . base Ltd one
merger half three division trading it to company
before CES mln may to . . , and U is - exclusive
affiliate - biggest its Association […]
from operator import mul
print reduce(mul, [counts[w] for w in text],
1.0)
# 3.0290546883e-32
```

Generate a text with 100 words (=unigram language model)

Probability of generated text

Generative Language Model II

```
from nltk.corpus import reuters
from nltk import bigrams, trigrams
from collections import Counter, defaultdict
first_sentence = reuters.sents()[0]
print first_sentence                # [u'ASIAN', u'EXPORTERS', u'FEAR', u'DAMAGE', u'FROM`…
print list(bigrams(first_sentence)) # [(u'ASIAN', u'EXPORTERS'), (u'EXPORTERS', u'FEAR'),…
print list(bigrams(first_sentence, pad_left=True, pad_right=True))
print list(trigrams(first_sentence, pad_left=True, pad_right=True))
model = defaultdict(lambda: defaultdict(lambda: 0))
for sentence in reuters.sents():
    for w1, w2, w3 in trigrams(sentence, pad_right=True, pad_left=True):
        model[(w1, w2)][w3] += 1
print model["what", "the"]["economists"]   # "economists" follows "what the" 2 times
print model["what", "the"]["nonexistingword"] # 0 times
print model[None, None]["The"]             # 8839 sentences start with "The"
for w1_w2 in model:
    total_count = float(sum(model[w1_w2].values()))
    for w3 in model[w1_w2]:
        model[w1_w2][w3] /= total_count
```

Language Model

- What is the probability of the following sentence „*the weather is nice*" under a bigram language model learnt from the corpus below?

the weather was bad yesterday
today the weather will be better
it is nice today

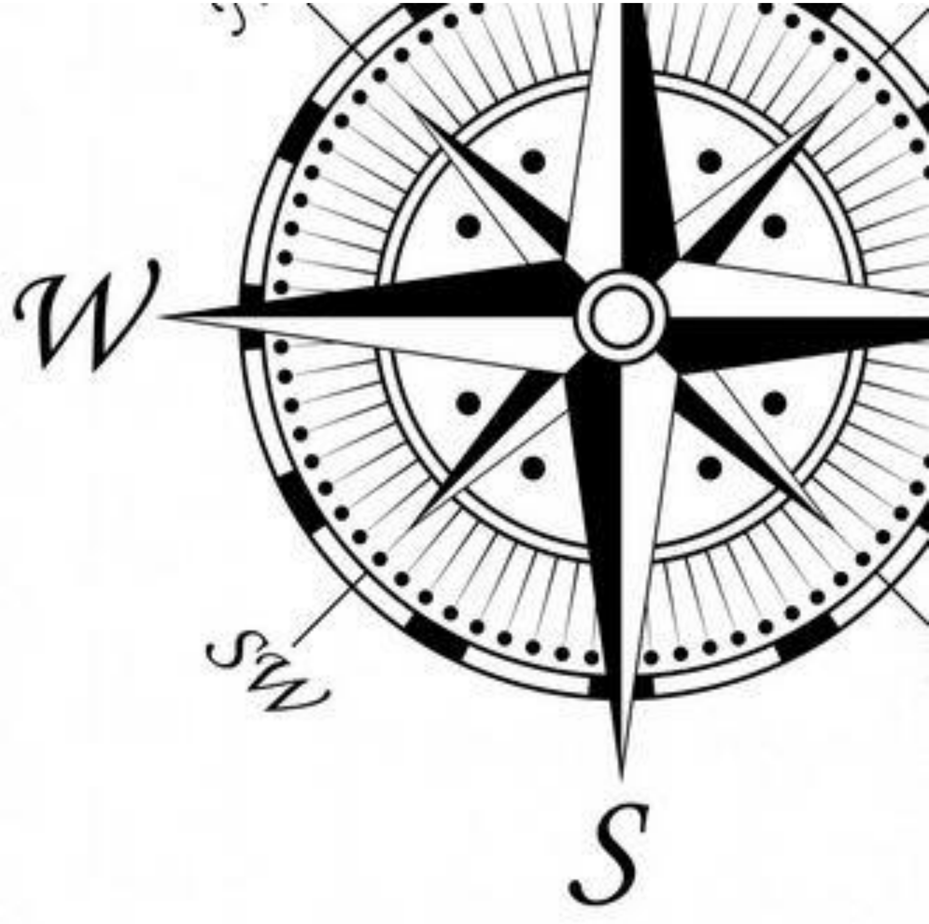
  – Use Laplace smoothing!
  – Laplace-smoothed bigrams:

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

Start 5 4 3 2 1 End

Topics Today

1. Language Models
2. **A RNN Language Model**
3. Recurrent Neural Networks (RNN)
4. Different RNN Types
5. A Simple RNN



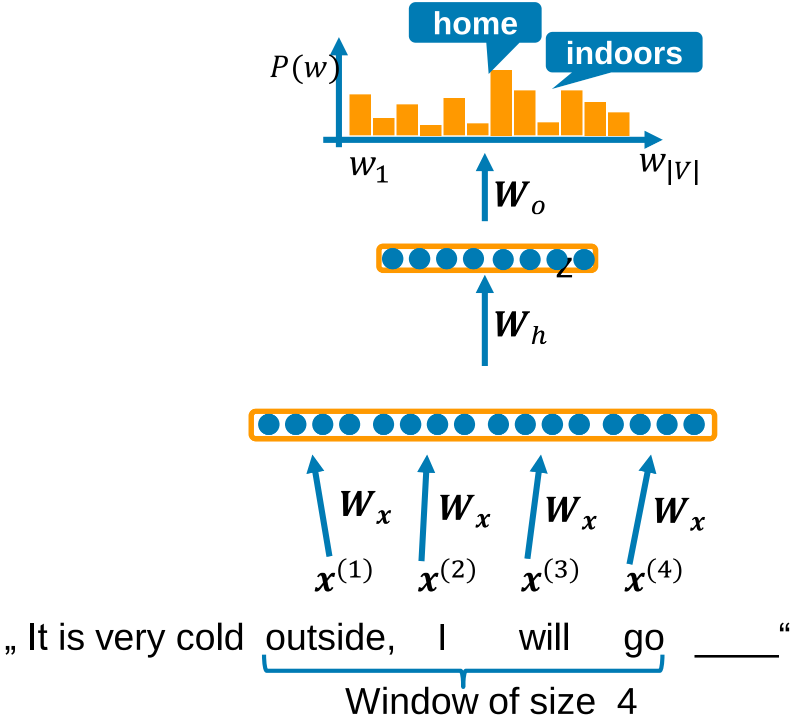
Neural Language Model With Fixed Window Size I

- Output=probability distribution

  $$\hat{\boldsymbol{y}} = softmax(\boldsymbol{W}_o\boldsymbol{h} + \boldsymbol{b}_o) \in \mathbb{R}^{|V|}$$

- Hidden layer
  - $\boldsymbol{h} = f(\boldsymbol{W}_h\boldsymbol{e} + \boldsymbol{b}_h)$

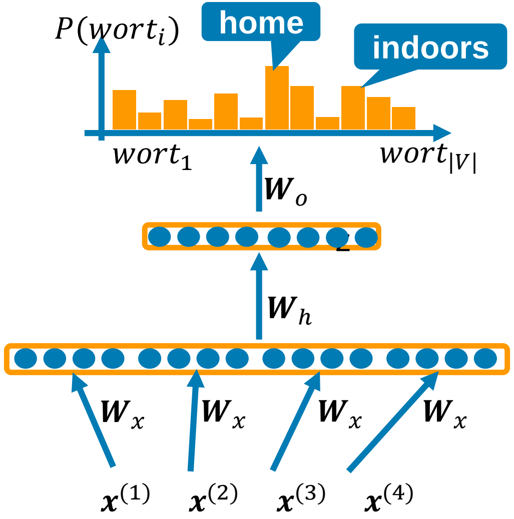
- Concatenated word embeddings
  - $\boldsymbol{e} = [\boldsymbol{e}^{(1)} = \boldsymbol{W}_x\boldsymbol{x}^{(1)}; \boldsymbol{e}^{(2)} =$



Neural Language Model With Fixed Window Size II

- Advantage over n-gram language model
  - No sparsity problem
  - Model size is in $O(n)$ not $O(\exp(n))$

- Not yet solved:
  - Fixed window size too small
  - Increasing window size increases $\boldsymbol{W}_h$
    - Window will never be large enough!
  - Weights are not shared among $x^{(i)}$



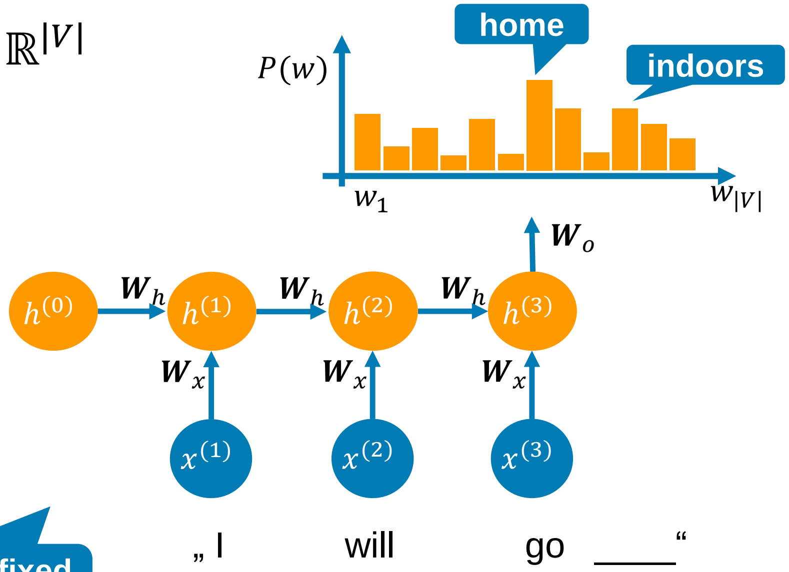
- We need a model that can process input sequences of different lengths.

RNN Language Model

- Probability distribution

$$\hat{\boldsymbol{y}} = softmax\left(\boldsymbol{W}_o \boldsymbol{h}^{(T)}\right) \in \mathbb{R}^{|V|}$$

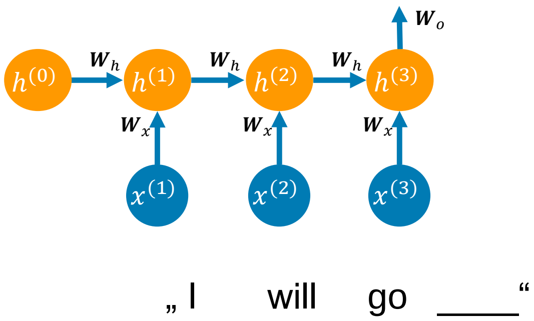
- Hidden layer
  - $\boldsymbol{h}^{(t)} = f(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{h}^{(t)})$
- Word embedding vectors
  - $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$



Not limited to fixed size anymore!

RNN Language Model

- Advantage over fixed window
  - Input data is processed sequentially.
    - Input can be of variable length.
  - Weights are shared across time steps in a state matrix.
    - (Theoretically) access to information at time step t from many time steps before



- Disadvantages of RNNs
  - Computation accross many time steps very slow
  - In practice, it is hard to access old information

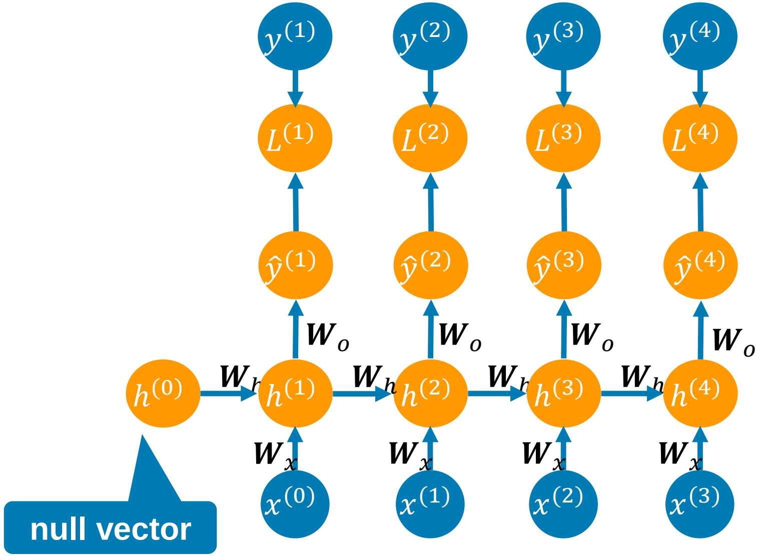
Learning the Weights

- Take a large corpus
  - Sequence of words $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(T)}$
- Compute for each word $\boldsymbol{x}^{(t)}$ a probability distribution $\widehat{\boldsymbol{y}}^{(t)}$ given all previous words
- Loss function for step t is the cross entropy between predicted distribution $\widehat{\boldsymbol{y}}^{(t)}$ and actual next word $\boldsymbol{y}^{(t)} = \boldsymbol{x}^{(t+1)}$:

$$L^{(t)}(\theta) = CE\left(\widehat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}^{(t)}\right) = -\sum_{j=1}^{|V|} y_j^{(t)} log \hat{y}_j^{(t)}$$

- Total loss is average:

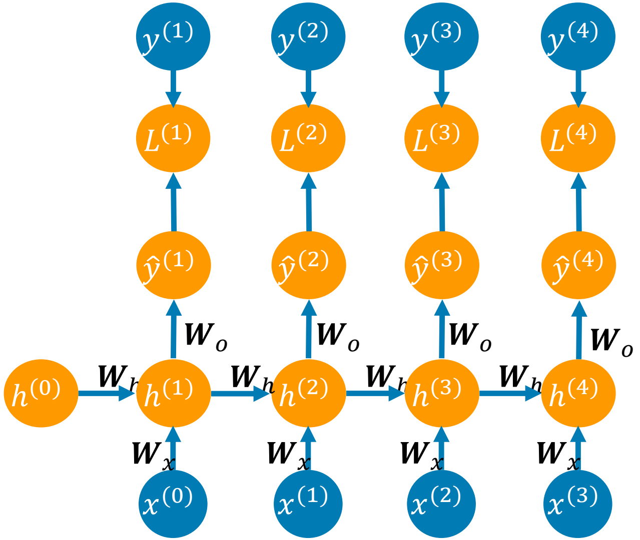
$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} L^{(t)}(\theta)$$



Train RNNs

- Computing the loss function and the gradients for the whole corpus is way too expensive!
- Stochastic gradient descent to the rescue
  - Update weights using small samples
- → computation per sentence
  - $\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(T)}$ is a sentence

$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} L^{(t)}(\theta)$$

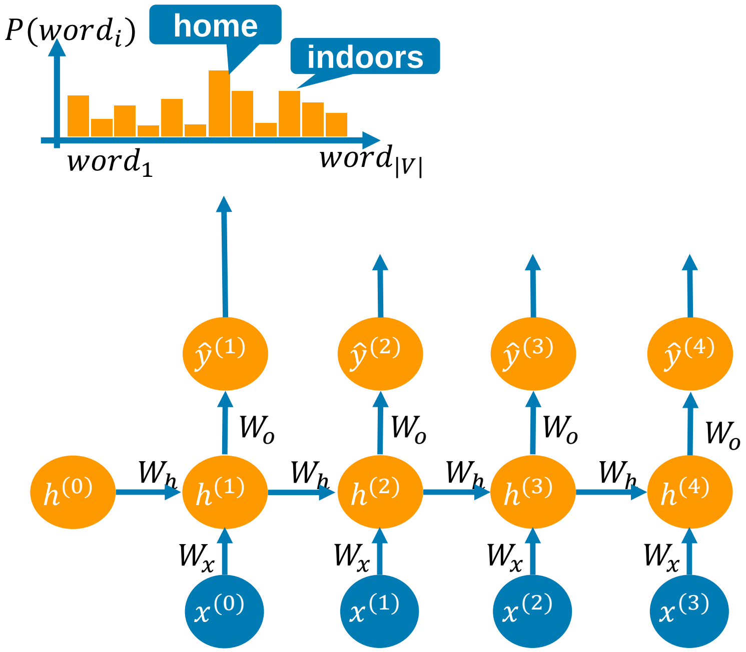


- Computation of $L(\theta)$ for one sentence:
  1. Computation of the gradients
  2. Update the weights
  3. Continue with next sentence

Generation of Sentences

- Analogous to n-gram language model
  - Repeated sampling of words
  - The sampled words in one step become the input for the next step
  - At some point there will be a <eos>-token sampled
  - In case a <unk>-token is sampled, ignore and sample again

`np.random.choice`



Examples I

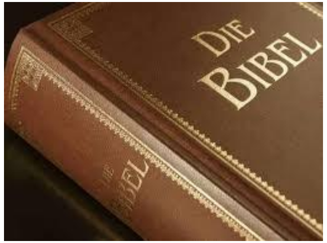
- The kind of text that is generated depends on the training data

Good morning. And as we mark the fact that they can stand with their companies that are consistent to the state of Pakistan and the United States of America.
With the financial system we can do that. And the people of the United States will not be able to continue to support the people of the greatest problem of the American people to stay in the White House. And that's why I've got to recognize the private sector that there is no doubt that we've got to continue to shape the painful realisation that we are the United States of America.

JERRY:
Well the elevator opens and wrong side of the door… I thought maybe the door's not waiting, but it said "going down" and Kramer couldn't help me move it. I just wanted to get out of it, just get out.
(He slams his hand on the door.)
KRAMER enters dancing with garbage.
KRAMER:
Hey hey hey, great idea for a big sponge: Make it so large you think it's got a fat clock in the middle.
JERRY
(takes off his bones)
Kramer, do you have a fun flashback to do?

23:2 And the vision of the breaking thereof shall be in rubbick, and they shall take away the stones out of the land.

24:11 Thus saith the LORD of hosts; Ask now this stones are for the righteous and the children of Israel.

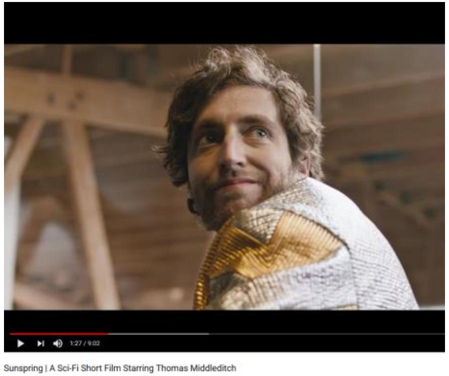


https://twitter.com/RNN_Bible
https://www.avclub.com/a-bunch-of-comedy-writers-teamed-up-with-a-computer-to-1818633242
https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0

Example II

- Movie scripts written by AI (with some human help, e.g., selection)

https://www.youtube.com/watch?v=5qPgG98_CQ8



https://www.youtube.com/watch?v=LY7x2Ihqjmc

Evaluation of Language Models

- Typically computation of **perplexity** on test set $\mathrm{W} = w_1 \ldots w_T$

$$PP(W) = P(w_1 \ldots w_T)^{-\frac{1}{T}}$$

Normalization over number of words

$$PP(W) = \left(\prod_{i=1}^{T} \frac{1}{P(w_i | w_1 \ldots w_{i-1})}\right)^{\frac{1}{T}}$$

  - Lower is better!

- Or **log-likelihood**

$$\sum_{i=1}^{T} \log P(w_i | w_1 \ldots w_{i-1})$$

  - Higher is better!

Model	Perplexity
Interpolated Kneser-Ney 5-gram (Chelba et al., 2013)	67.6
RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013)	51.3
RNN-2048 + BlackOut sampling (Ji et al., 2015)	68.3
Sparse Non-negative Matrix factorization (Shazeer et al., 2015)	52.9
LSTM-2048 (Jozefowicz et al., 2016)	43.7
2-layer LSTM-8192 (Jozefowicz et al., 2016)	30
Ours small (LSTM-2048)	43.9
Ours large (2-layer LSTM-2048)	39.8

One billion parameters

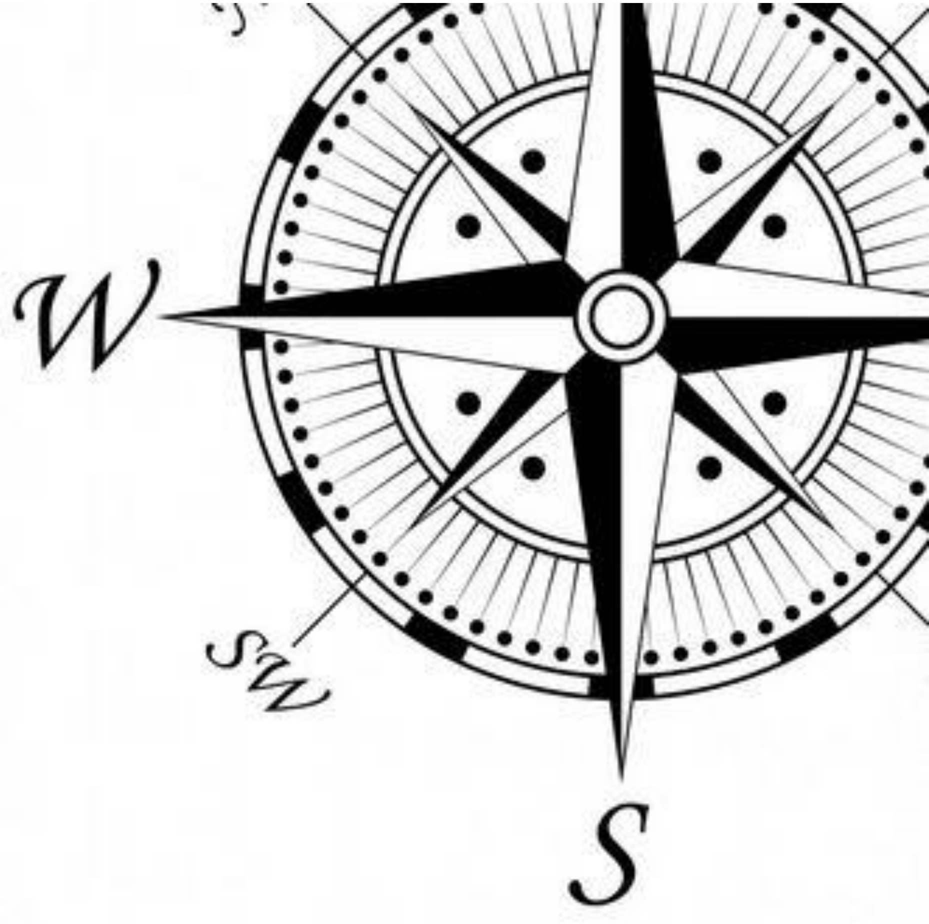
https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/

Generative Language Models

- Are automatically generated texts useful?
  - If so, in which context?
- Which are applications where language models can be used in a meaningful way?
  - (Except to generate text)
- Ethical, legal considerations?
  - Copyright, authorship, …
  - Fake news, „truth“, …

Start 5 4 3 2 1 End

Topics Today

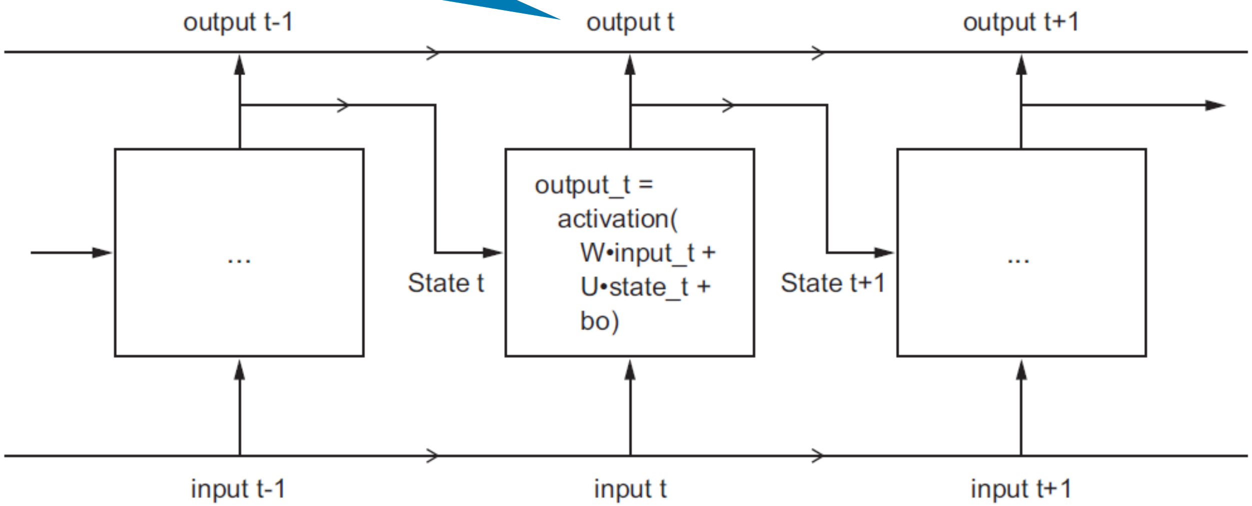


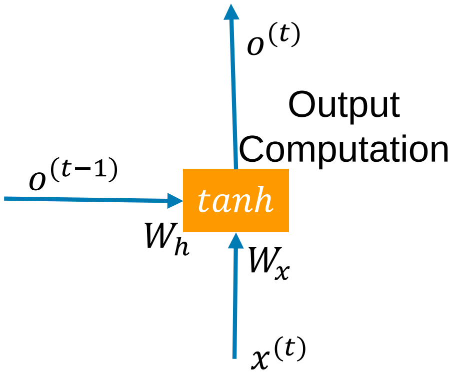
1. Language Models
2. A RNN Language Model
3. **Recurrent Neural Networks (RNN)**
4. Different RNN Types
5. A Simple RNN

Rolled-Out Layer

- Output of layer at timestep t
  - $h^{(t)} = \tanh(W_h h^{(t-1)} + W_x x^{(t)} + b_h)$

Output layer above, e.g. softmax

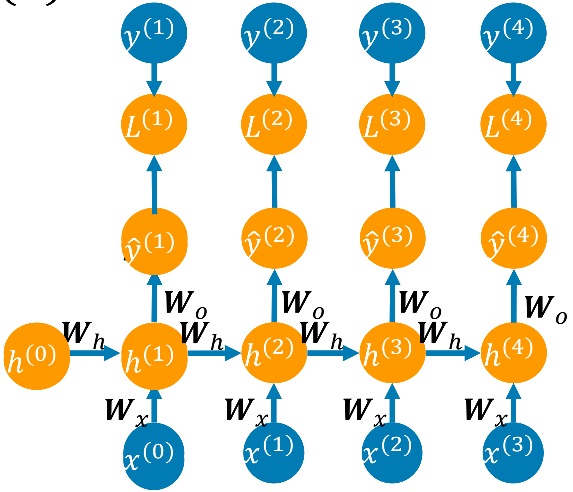




Forward Computation

- $L^{(t)}(\theta) = CE\left(\hat{\boldsymbol{y}}^{(t)}, \boldsymbol{y}^{(t)}\right) = -\sum_{j=1}^{|V|} y_j^{(t)} log \hat{y}_j^{(t)}$
- Total loss is average:

$$L(\theta) = \frac{1}{T}\sum_{t=1}^{T} L^{(t)}(\theta)$$



Forward Computation Implementation

```
import numpy as np
timesteps = 100
input_features = 32
output_features = 64
inputs = np.random.random((timesteps, input_features))
h_t = np.zeros((output_features,))
W_x = np.random.random((output_features, input_features))
W_h = np.random.random((output_features, output_features))
b_h = np.random.random((output_features,))
successive_outputs = []
for input_t in inputs:
        output_t = np.tanh(np.dot(W_x, input_t) + np.dot(W_h, h_t) + b_h)
        successive_outputs.append(output_t)
        h_t = output_t
final_output_sequence = np.concatenate(successive_outputs, axis=0)
```

Components of an input sequence

Here: Input is random noise

Initial state = 0-vector

Also the weight matrices are randomly initialized

output_t, sufficient, since it contains information about the whole sequence

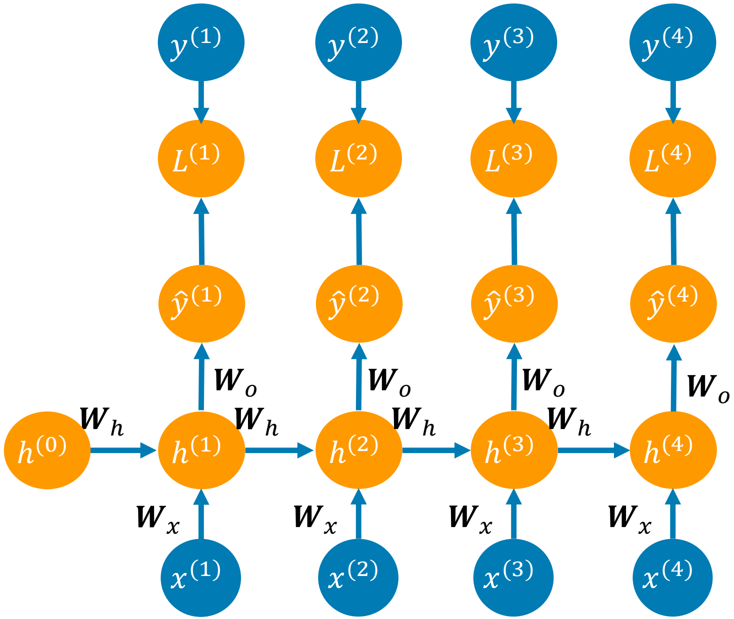
Output is a 2d-tensor of shape (timesteps, output_features)

Backward Computation

- **Backpropagation through time**
- Given: Multi-variable function $f(x,y)$ and two functions with one variable $x(t)$ and $y(t)$, then this is the multi-variable chain rule

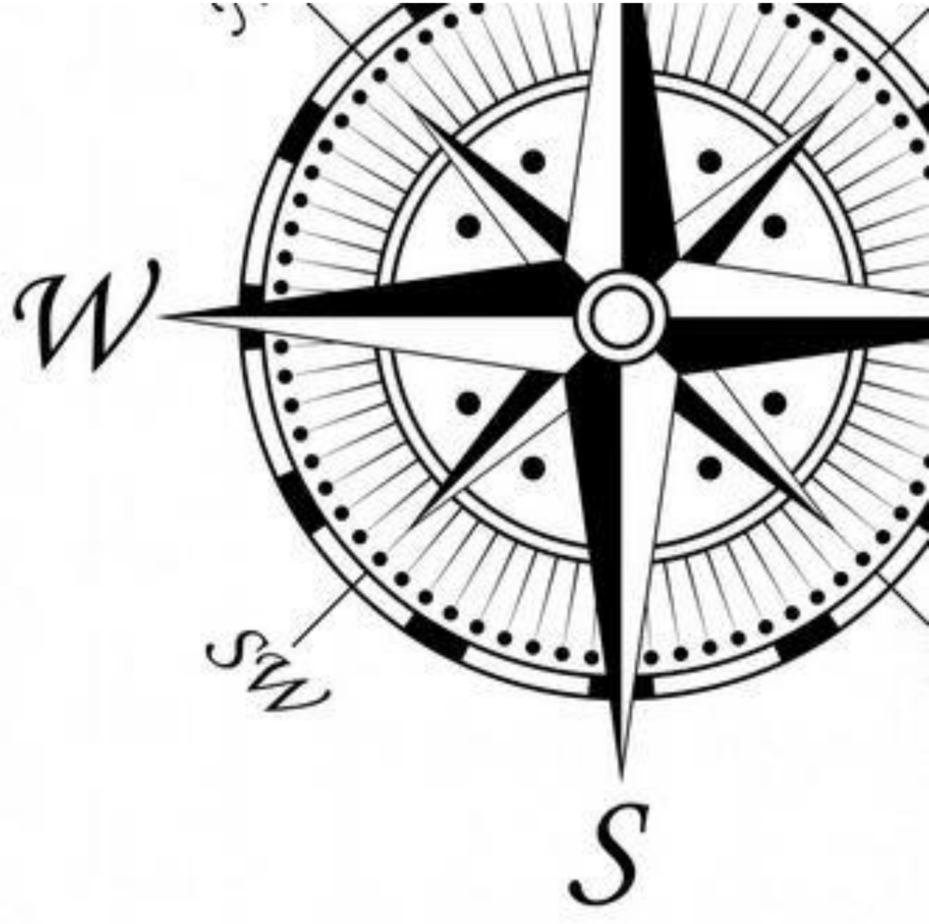
$$\frac{d}{dt} f\big(x(t), y(t)\big) = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

- Derivation of the loss fuction $L^{(t)}(\theta)$ with respect to repeating $\boldsymbol{W}_h$
  - https://www.youtube.com/watch?v=q4mVeRLitsU



Topics Today

1. Language Models
2. A RNN Language Model
3. Recurrent Neural Networks (RNN)
4. **Different RNN Types**
5. A Simple RNN

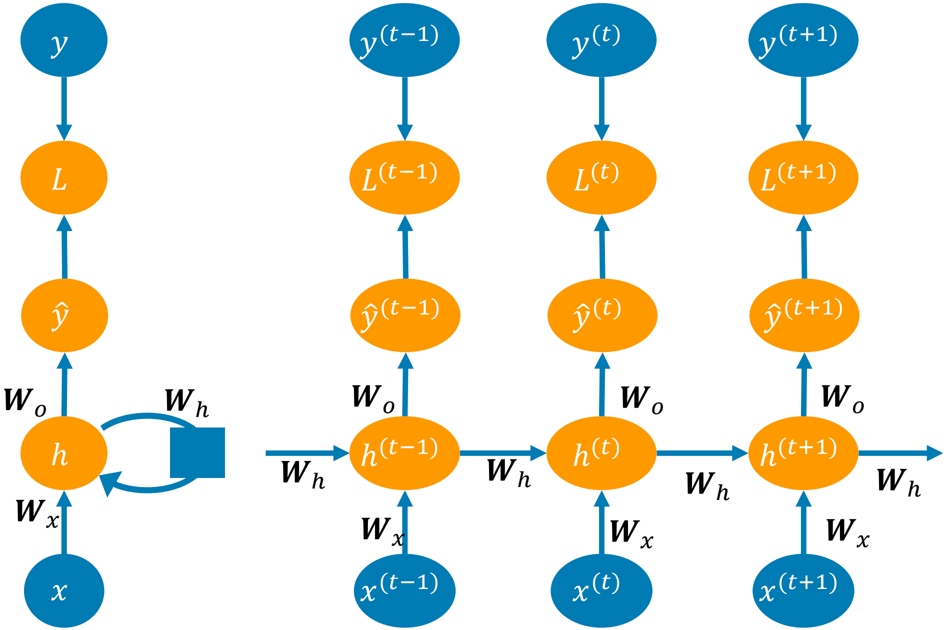


Categorization

- RNNs can be categorized based on their input and output.
  - **Many-to-many**
    - Standard RNN
    - Sequence-to-Sequence
  - **Many-to-one**
    - Summary
  - **One-to-many**
    - Generative models
  - **(One-to-one)**
    - Standard NN

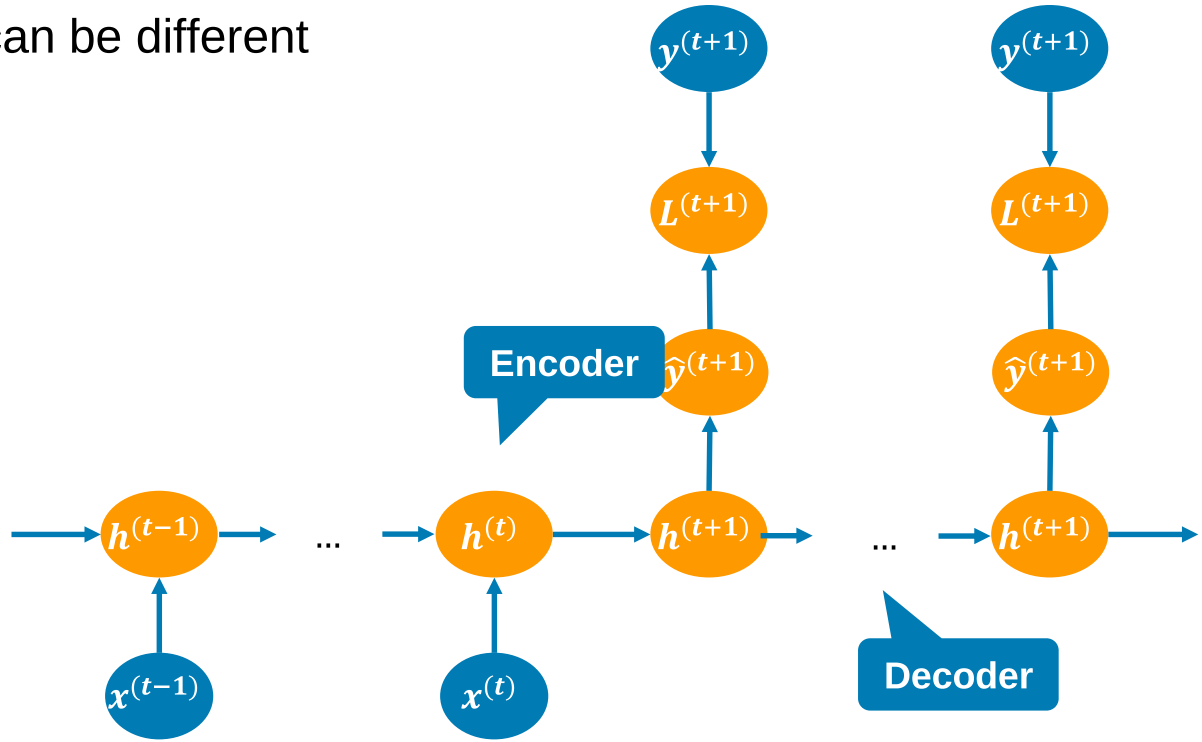
Many-to-Many: Standard RNN

- RNNs are very powerful
  - Turing-complete
- Through the hidden layer, access to all information from the past
- Input length equals output length
  - At each time step t there is an input $x^{(t)}$ and an output $y^{(t)}$



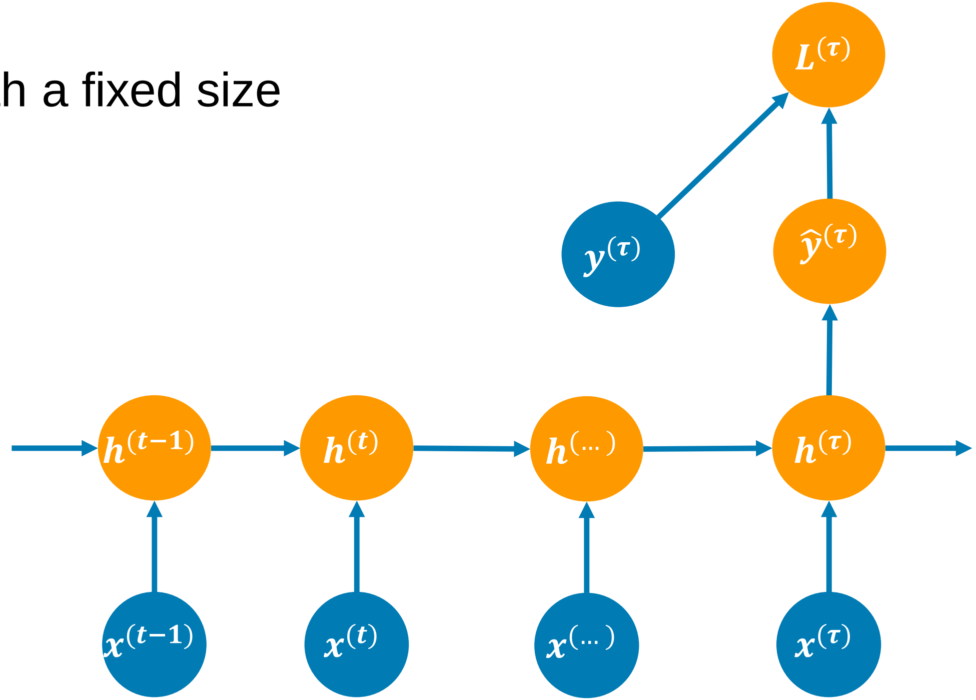
Many-to-Many: Sequence-to-Sequence

- Input and output length can be different
- E.g. machine translation



Many-to-One: Summary

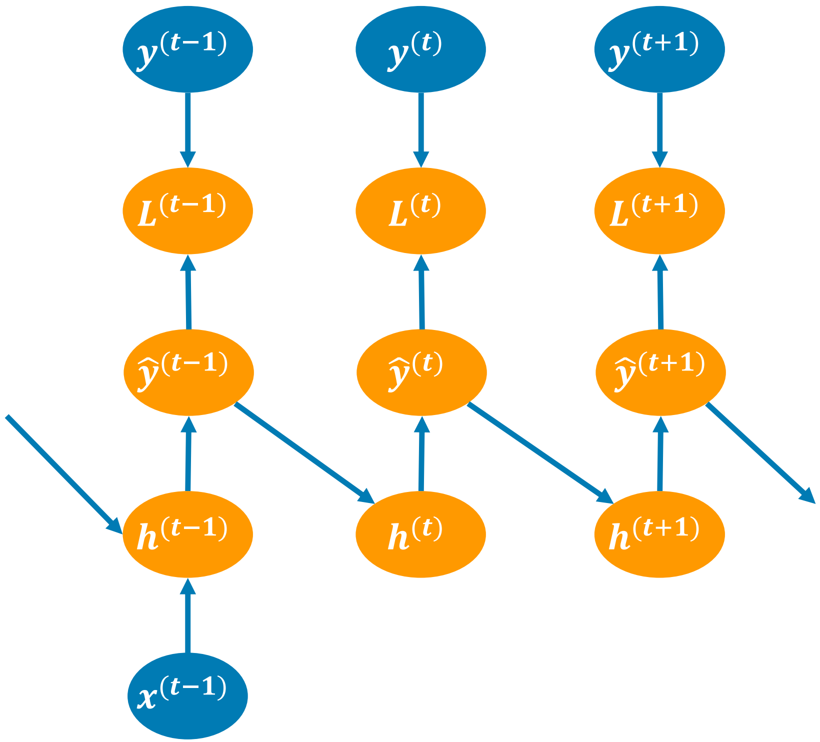
- Can summarize a sequence
  - Representation of a sequence with a fixed size
- E.g. sentiment analyse



- As input for further layers
  - Output will be learnt by backprop
- Here with ist own target value

One-to-Many: Generative Models

- E.g. music generation
- Input could be, e.g., an integer, to decide on the style
  - Input could also be empty
    $\boldsymbol{x} = \emptyset$

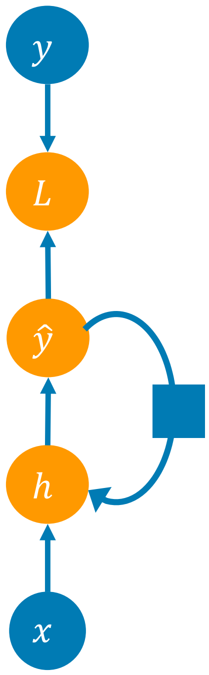


Many/One-to-Many: RNN Variants

- Less powerful
  - Only output is transmitted

- Only indirect connection of previous hidden layer to current hidden layer
  - Via output layer
    - Typically less dimenstions than hidden layer

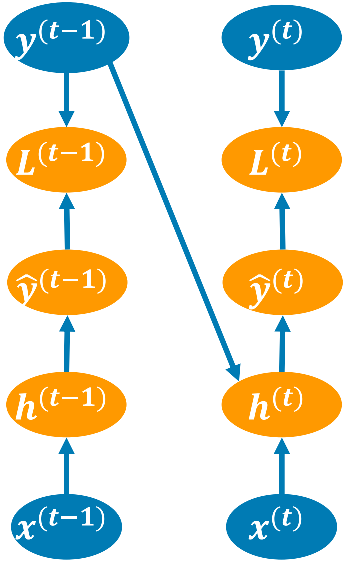
- Potentially simpler to train
  - Parallelization



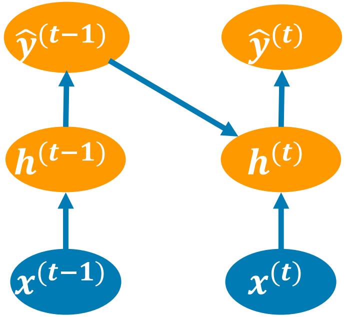
Many/One-to-Many: Teacher Forcing

- Method for RNNs to learn from ground-truth
- When the model is deployed, the ground-truth is approximated by the output

For training



For testing



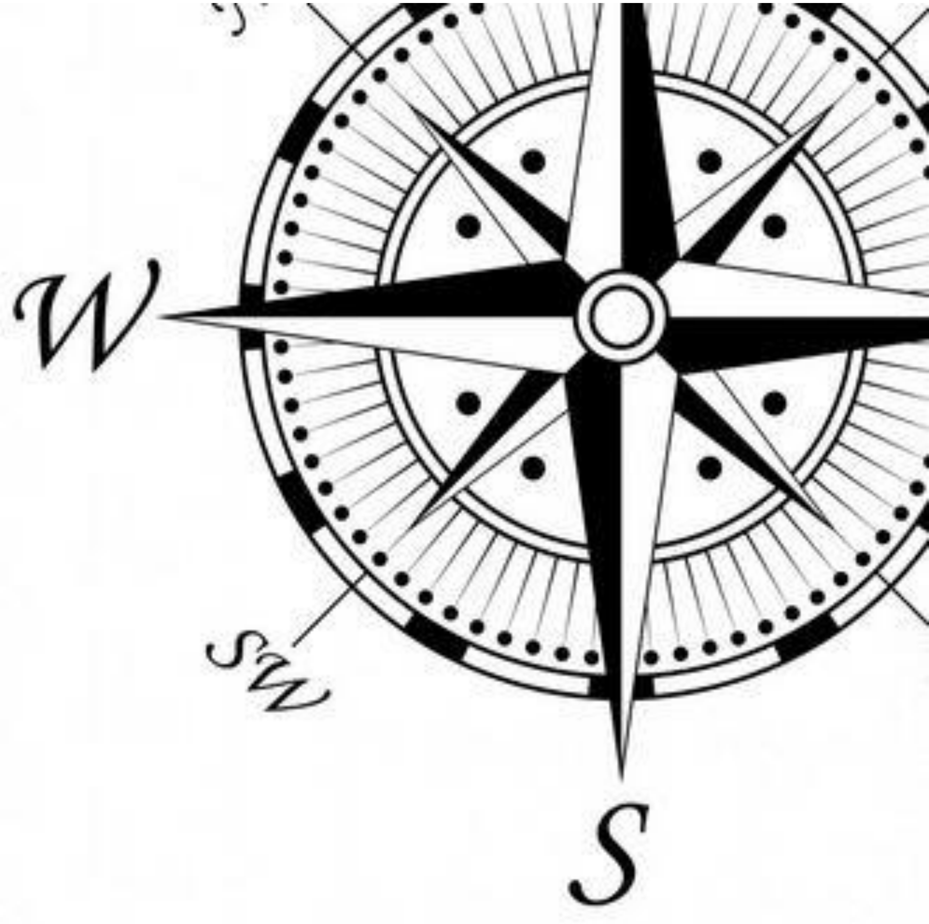
Recurrent Layer

- Think about suitable scenarios for the presented RNNs variants.
- What are advantages and disadvantages of teacher forcing?

Start 5 4 3 2 1 End

Topics Today

1. Language Models
2. A RNN Language Model
3. Recurrent Neural Networks (RNN)
4. Different RNN Types
5. **A Simple RNN**



Simple RNN in Keras

```
from keras.layers import SimpleRNN
```

- Batch processing, i.e. multiple sequences simultaneously

```
from keras.models import Sequential
from keras.layers import Embedding, SimpleRNN
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32), return_sequences=True))
model.summary()
```

```
________________________________________________________________
Layer (type)                 Output Shape              Param #
================================================================
embedding_22 (Embedding)     (None, None, 32)          320000
________________________________________________________________
simplernn_10 (SimpleRNN)     (None, 32)                2080
================================================================
Total params: 322,080
Trainable params: 322,080
Non-trainable params: 0
```

Stacked Simple RNN in Keras

```
model = Sequential()
model.add(Embedding(10000, 32))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32, return_sequences=True))
model.add(SimpleRNN(32))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_24 (Embedding)     (None, None, 32)          320000
_________________________________________________________________
simplernn_12 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simplernn_13 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simplernn_14 (SimpleRNN)     (None, None, 32)          2080
_________________________________________________________________
simplernn_15 (SimpleRNN)     (None, 32)                2080
=================================================================
Total params: 328,320
Trainable params: 328,320
Non-trainable params: 0
```

IMDB Example: Preprocessing

```
from keras.datasets import imdb
from keras.preprocessing import sequence
max_features = 10000
maxlen = 500
batch_size = 32
print('Loading data...')
(input_train, y_train), (input_test, y_test) =
imdb.load_data(num_words=max_features)
print(len(input_train), 'train sequences')
print(len(input_test), 'test sequences')
print('Pad sequences (samples x time)')
input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
input_test = sequence.pad_sequences(input_test, maxlen=maxlen)
print('input_train shape:', input_train.shape)
print('input_test shape:', input_test.shape)
```

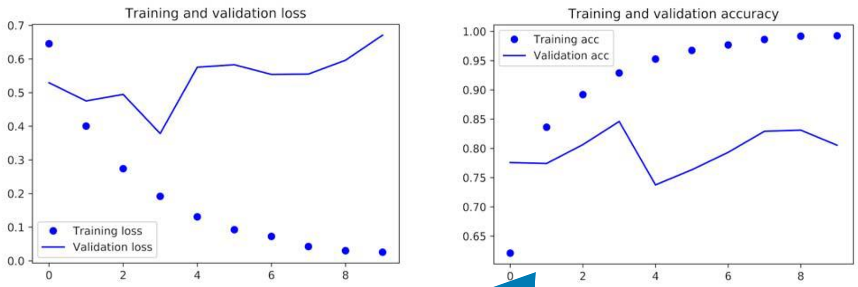
IMDB Example: Model

```
from keras.layers import Dense
model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

IMDB Example: Validation

```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```



Test accuracy without RNN: 88%;
With RNN only 85%;

Learning Goals for this Chapter

- Understand the difference between feed-forward-nets and recurrent network architectures
- Know application areas
- Adapt task description so that RNNs can be used to solve the problem
- Explain different kinds of RNNs and how they work
- Implement and evaluate a simple RNN model

- Relevant chapters:
  – P6.2, S5+S6 (2021)

Literature