

Machine Translation

Session 5: Neural Networks

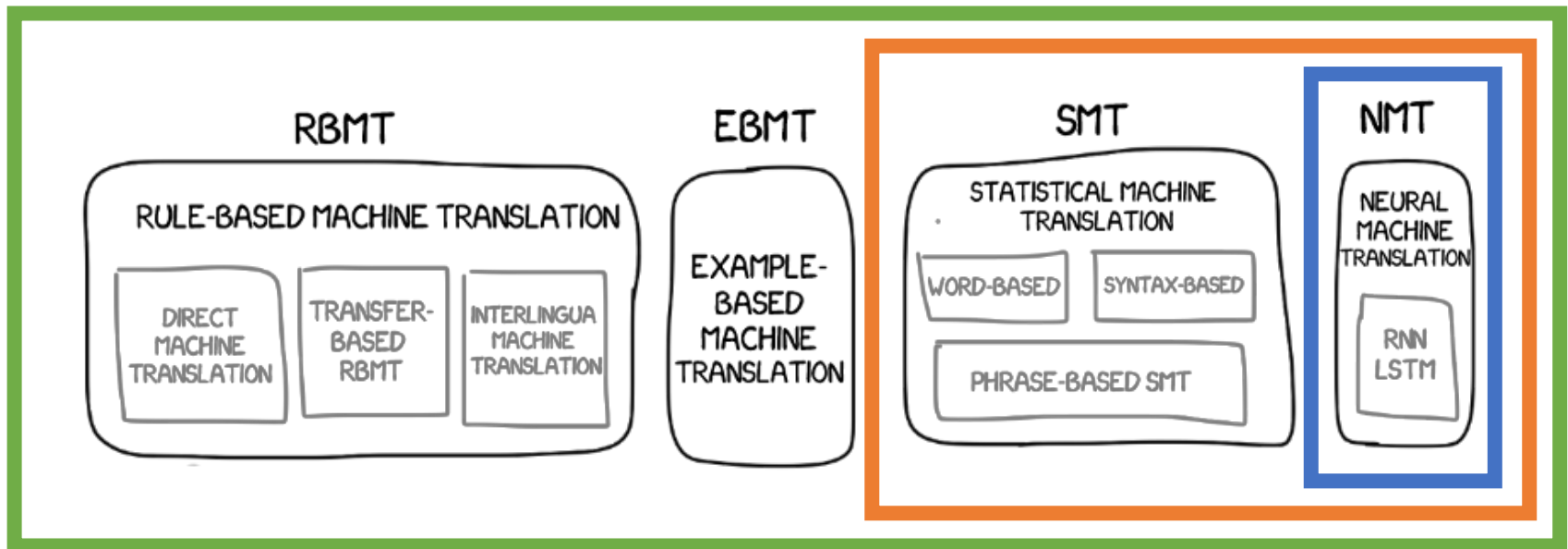
Sharid Loáiciga — May 20th 2020

Slides credits: Yves Scherrer

Evaluation

Parallel corpora

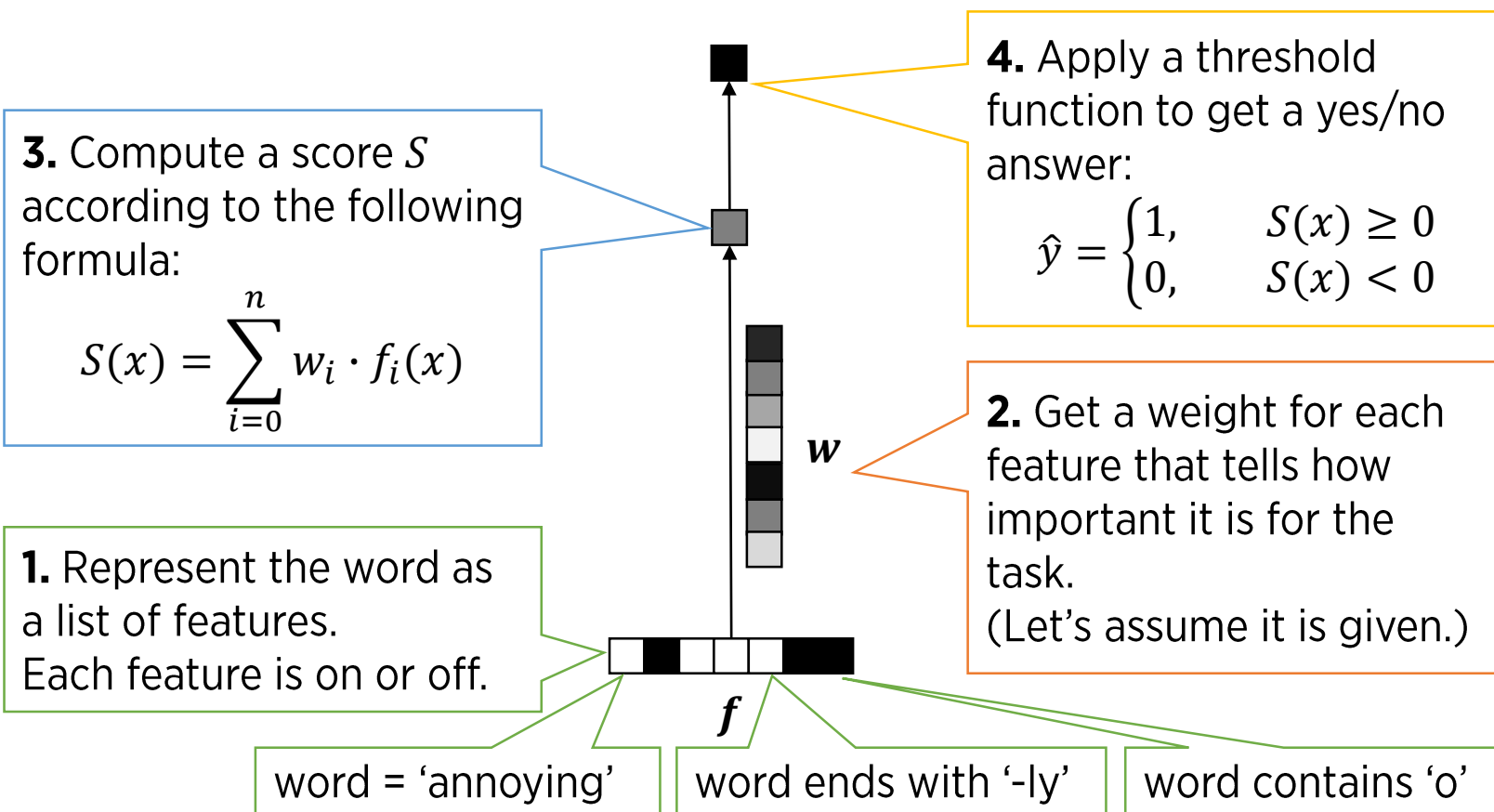
Models



Today: A crash course in neural networks...

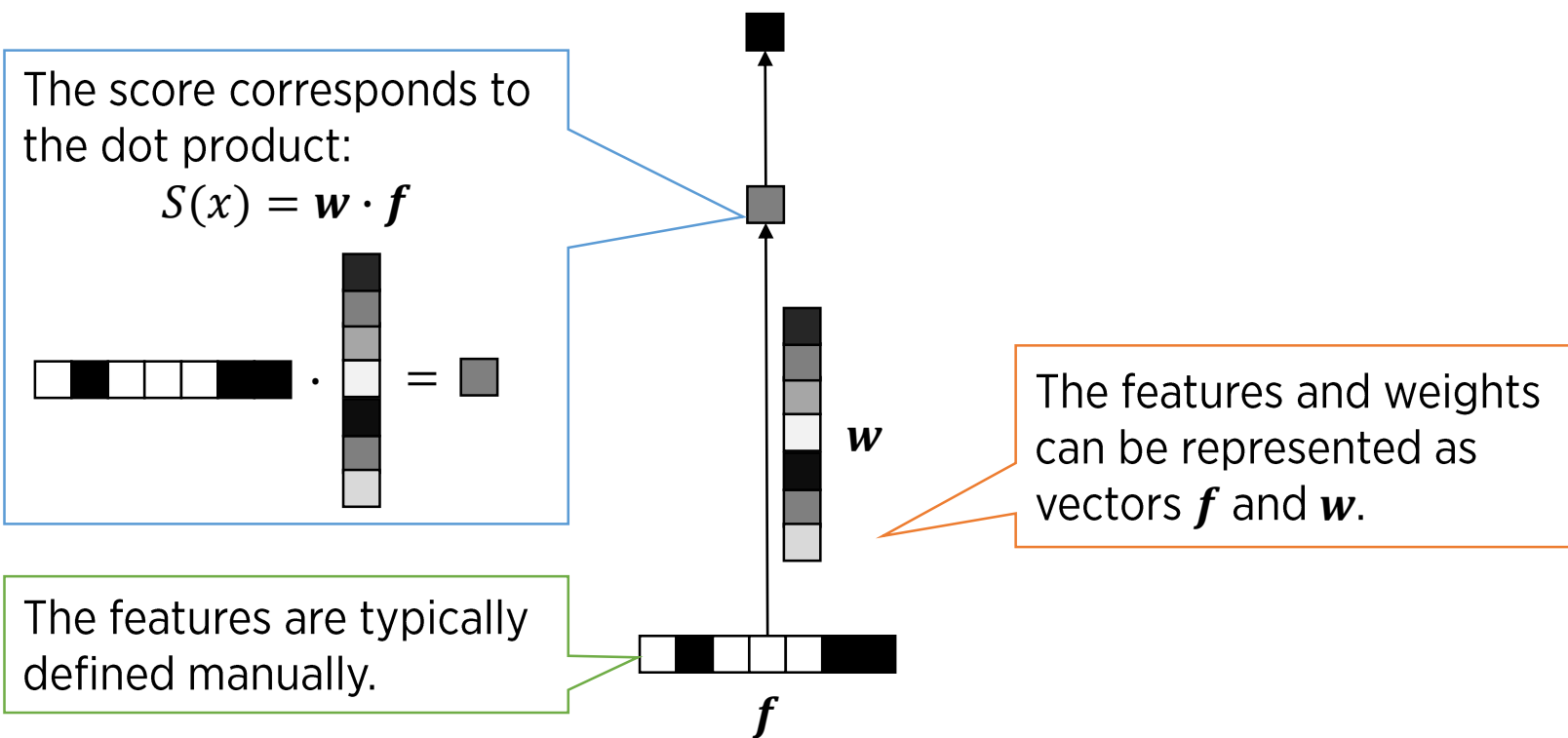
Linear models

- Example: Determine if a word has negative sentiment



Linear models

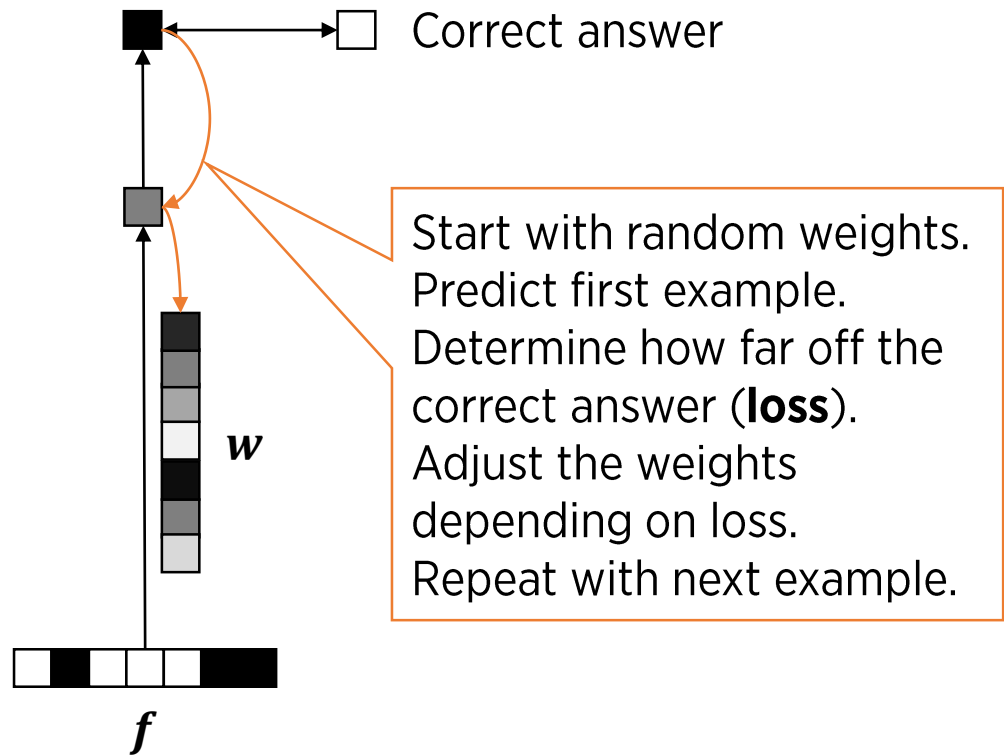
- Example: Determine if a word has negative sentiment



Linear models

Training

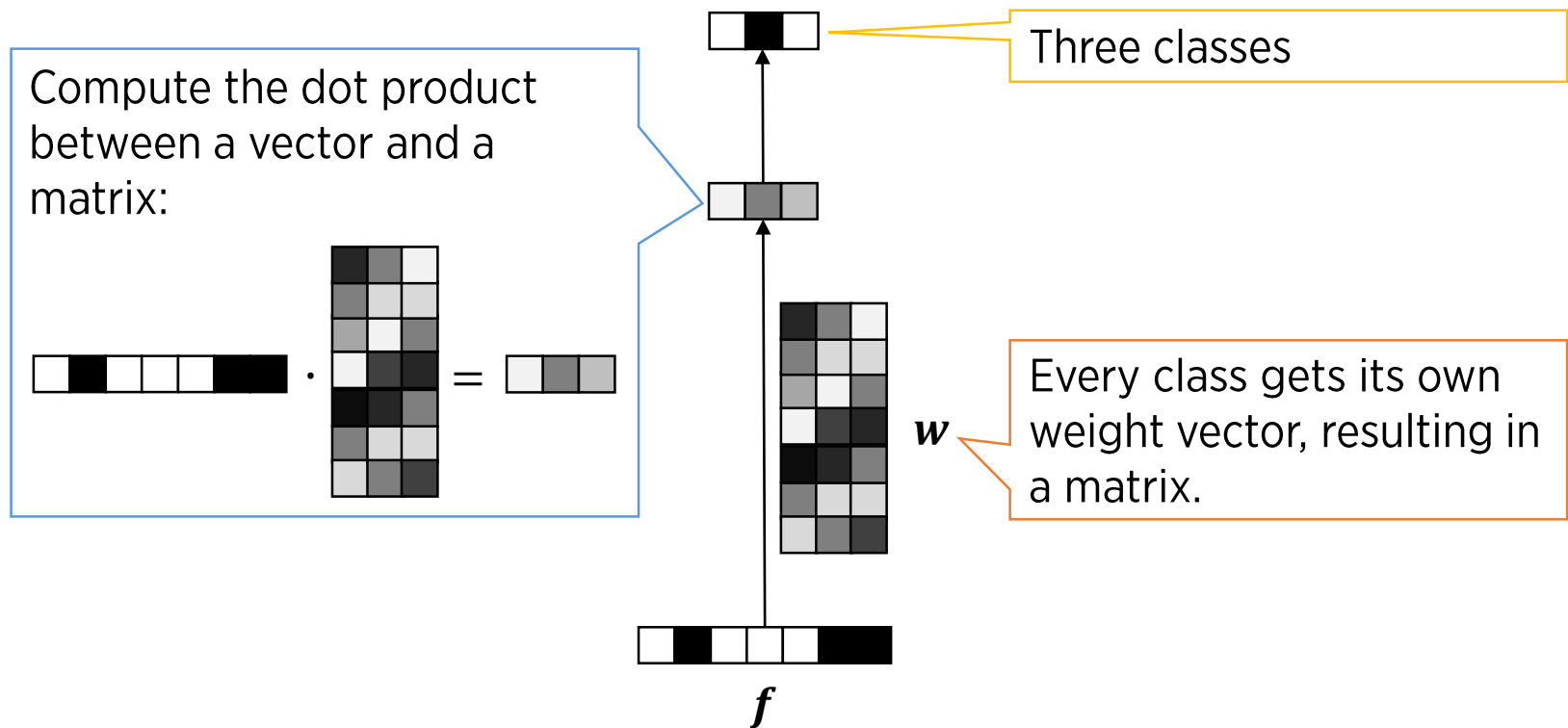
- Example: Determine if a word has negative sentiment



Linear models

Multi-class prediction

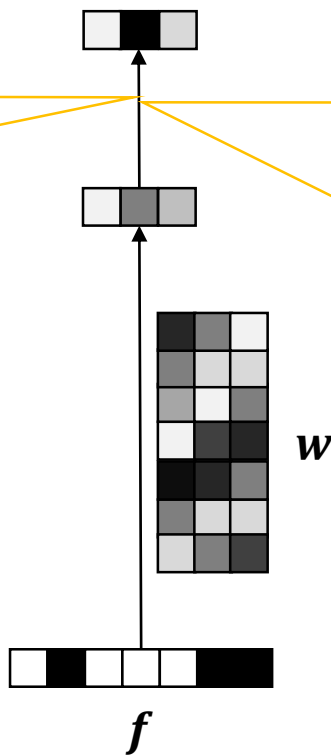
- Example: Determine the dominant sentiment of a word



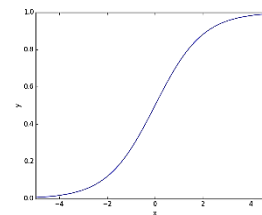
Sigmoid functions

- Example: Determine the dominant sentiment of a word

This part is also called **activation function**.



Threshold functions are not always practical. Use a different type of S-shaped (“sigmoid”) function:

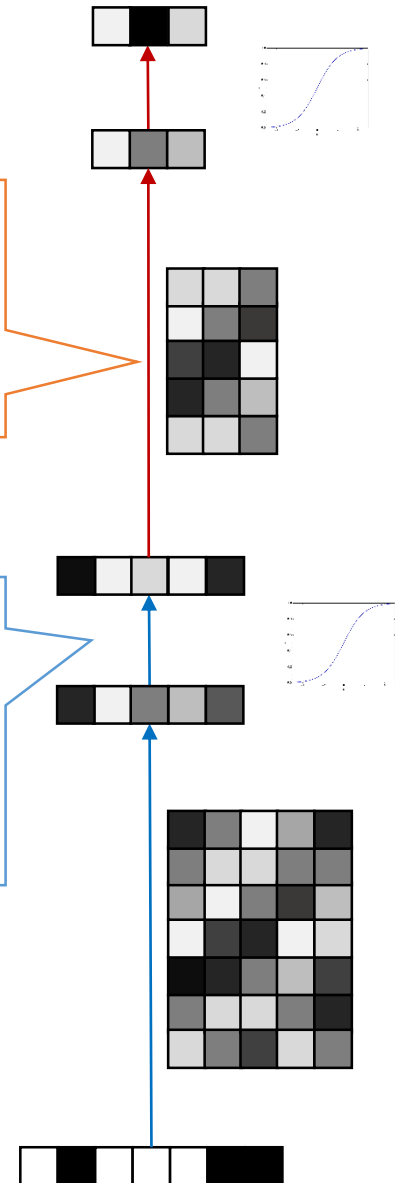


Two-step classification

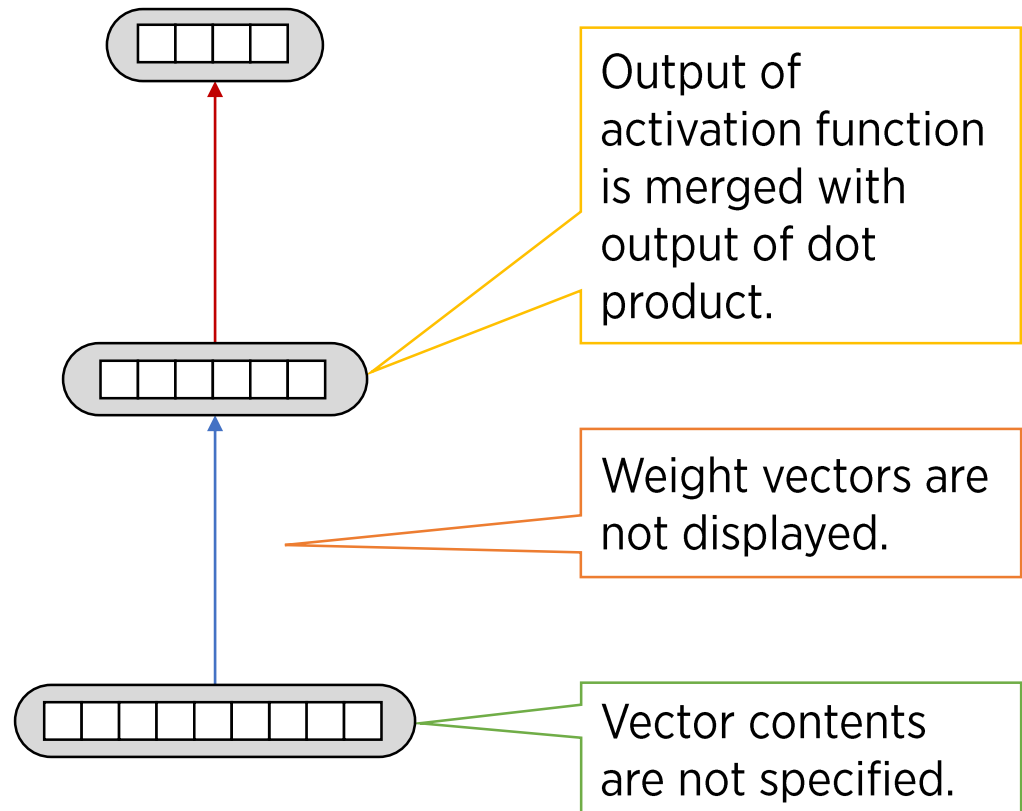
- Idea:
 - Partition the vocabulary into e.g. 100 word classes
 - Base the sentiment decision on the word class
- Let's use the output of the first model as the input features of the second model

Both weight matrices can be trained in a single pass:
backpropagation

This is a **hidden layer**.
Models with hidden layers are called **neural networks**.



A more compact representation



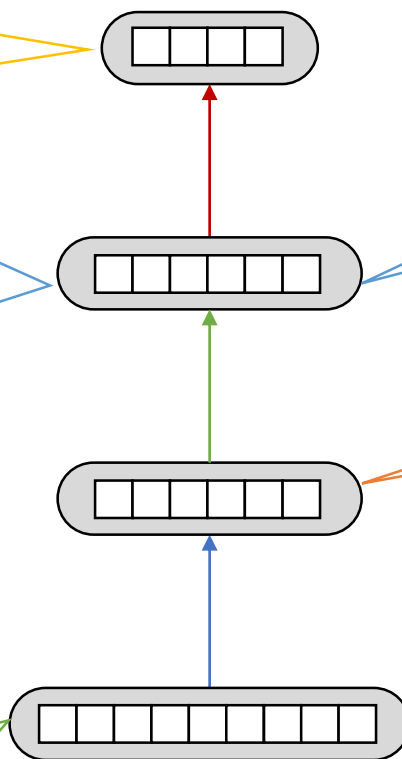
Neural networks

The size of the prediction vector is defined by the number of classes.

Several hidden layers can be added.

Neural networks with more than one hidden layer are called **deep models**.

The size of the input vector is defined by the number of distinct words in the training corpus.



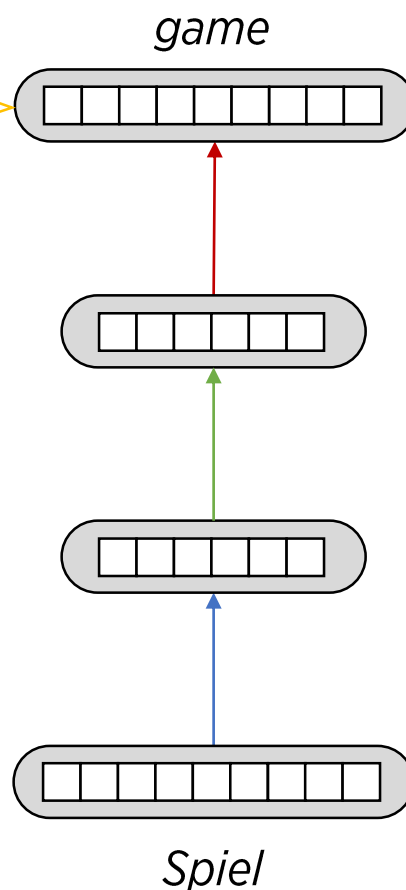
The size of the hidden layer vectors and the number of hidden layers can be freely chosen.

The first hidden layer is generally defined as the **embedding layer**.

We usually don't bother about the input features. Every word is represented by exactly one active feature. This is called **one-hot encoding**.

What about MT?

The prediction vector will be very large: one value per target word.



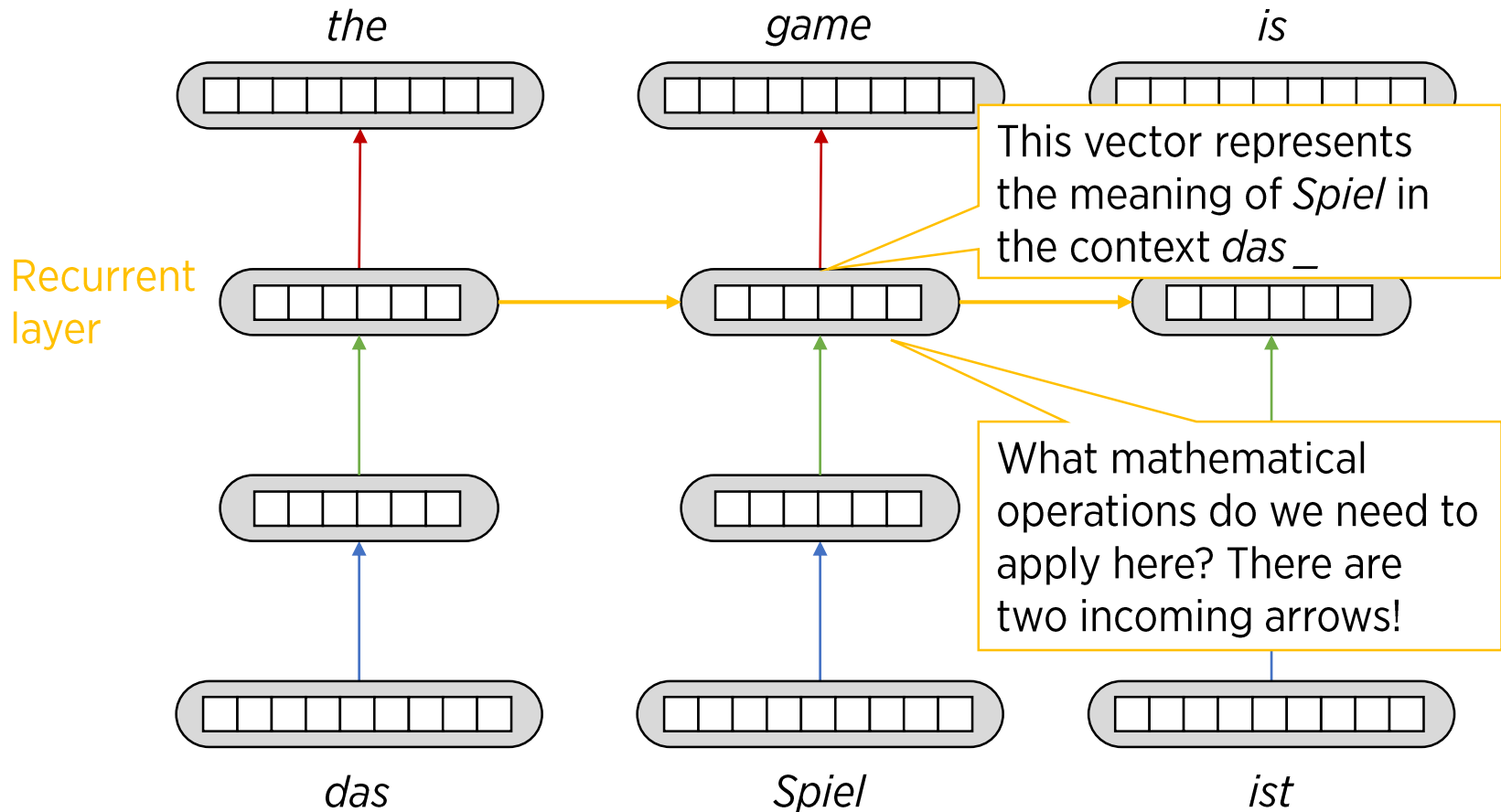
This model processes each word in isolation. It is just a sophisticated dictionary.

NLP is full of sequential data:

- Words in sentences
- Characters in words
- Sentences in text

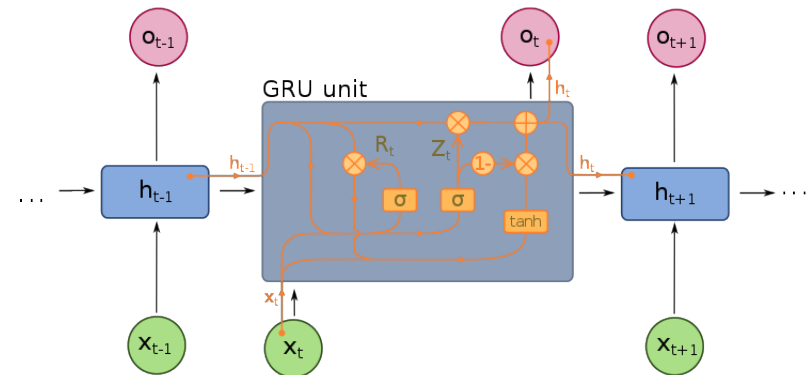
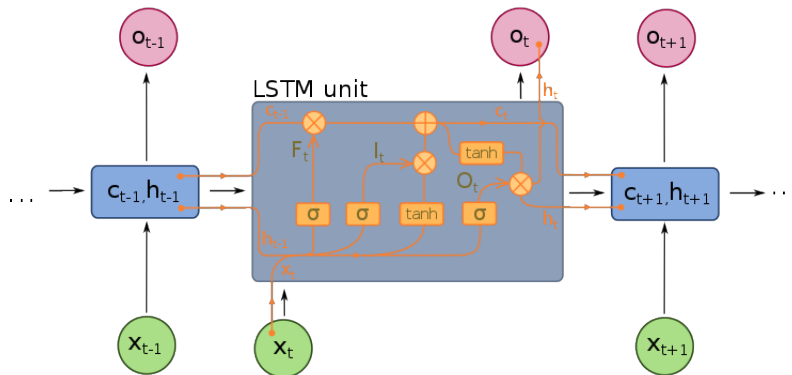
Models should take that into account.

Recurrent neural networks

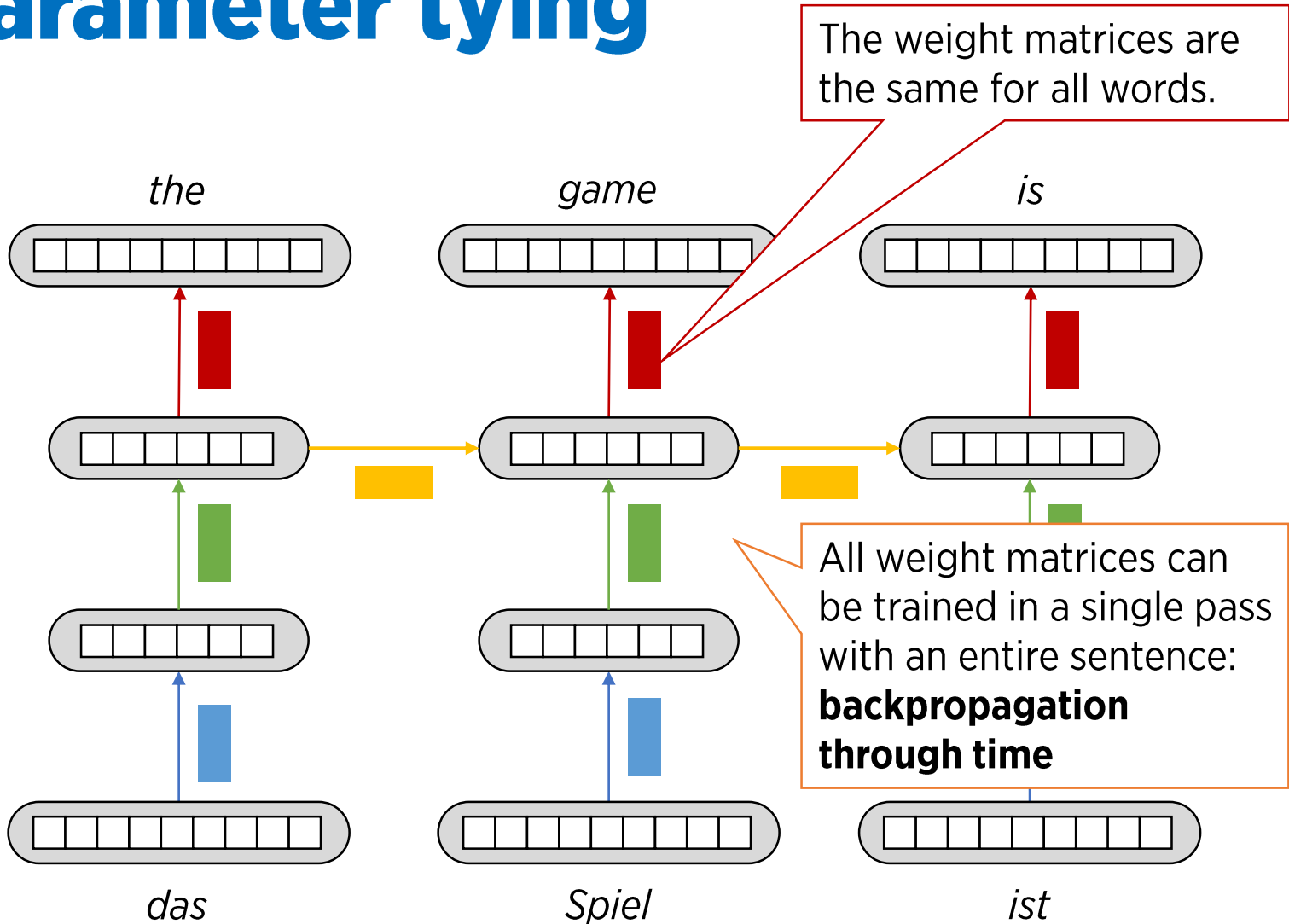


Recurrent neural networks

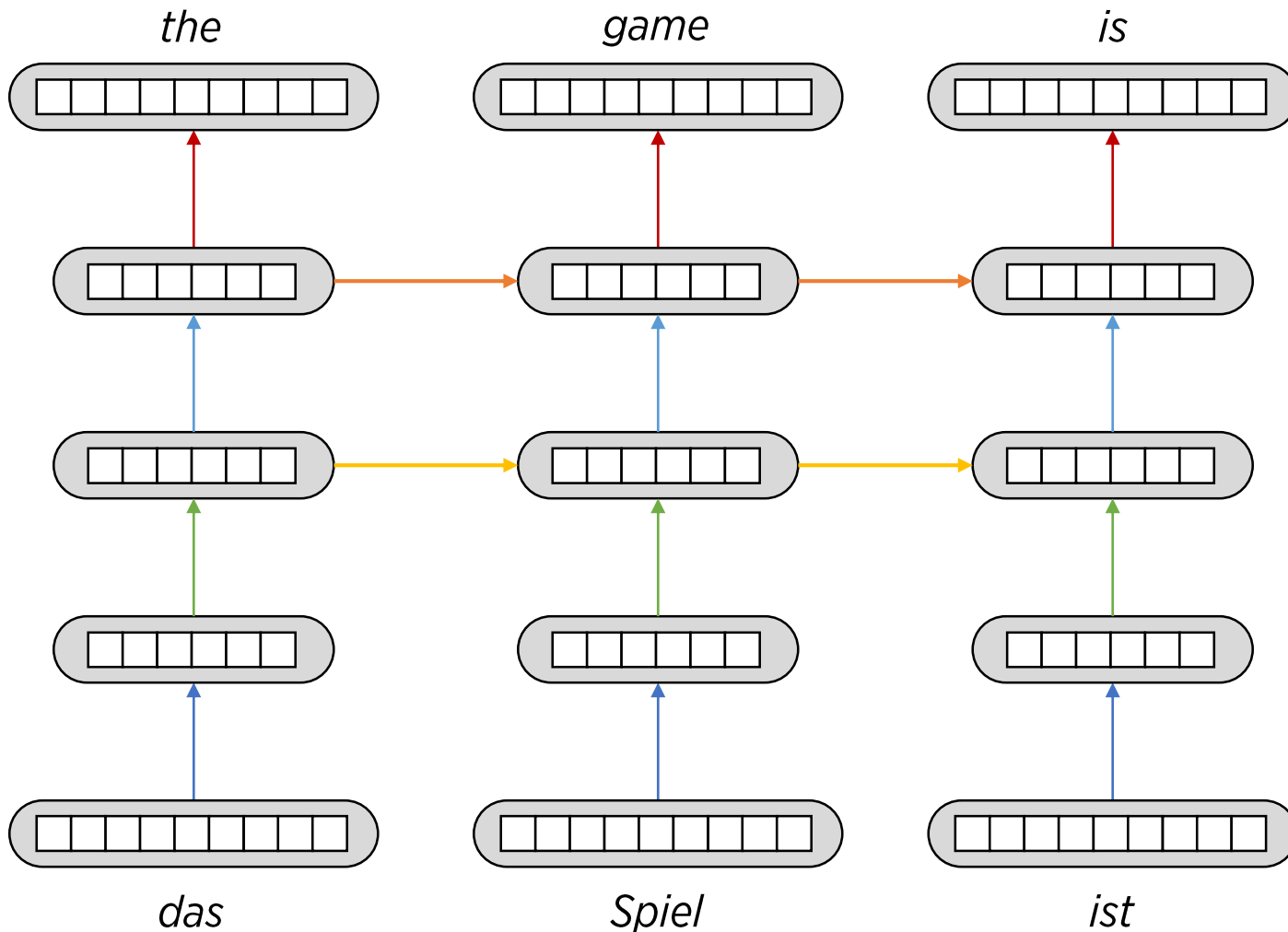
- Plain RNN:
 - Input vector dot product + context vector dot product
- Long-short-term memory (LSTM):
- Gated recurrent unit (GRU):



Parameter tying



Multiple recurrent layers



Recurrent neural networks

- What is the problem with this approach for machine translation?

Recurrent neural networks

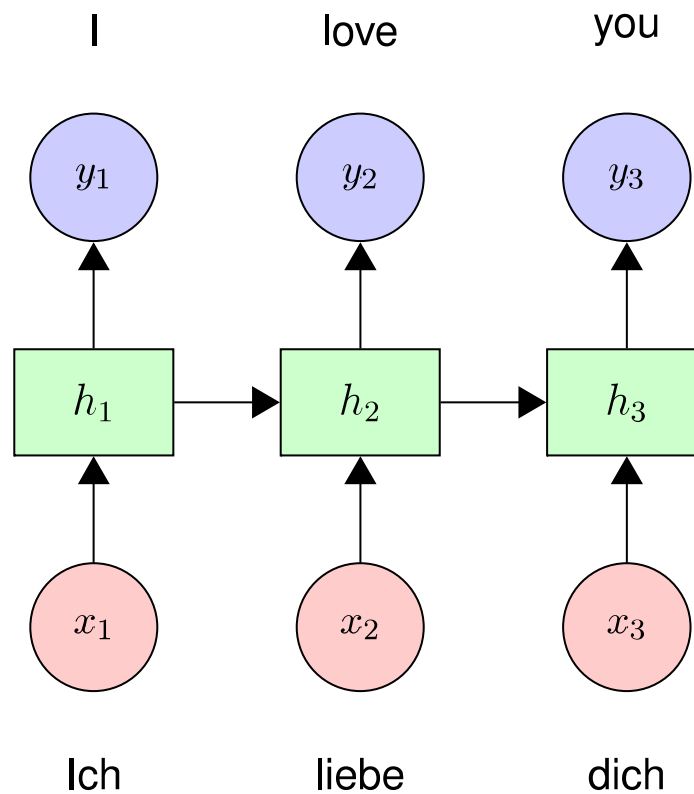


Illustration: Rico Sennrich

Recurrent neural networks

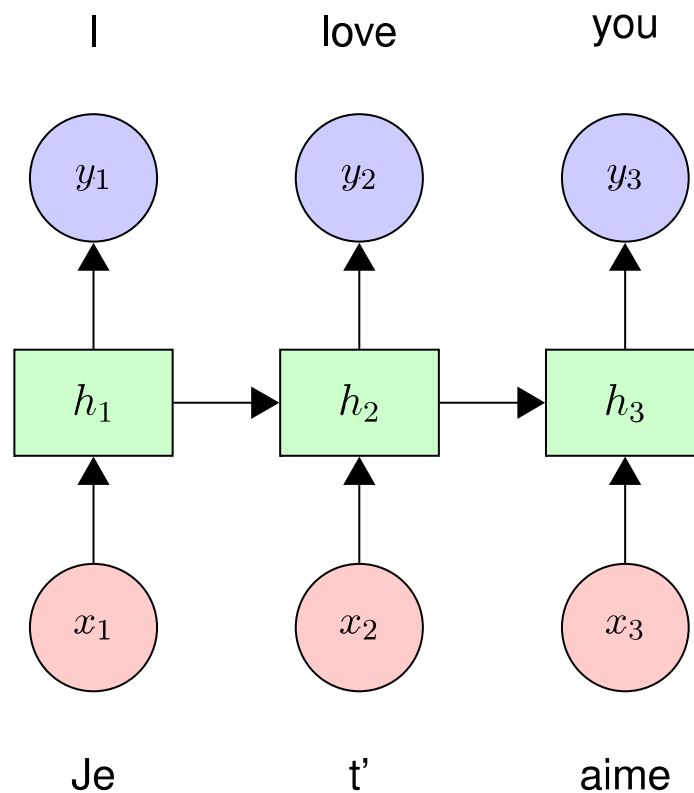


Illustration: Rico Sennrich

Recurrent neural networks

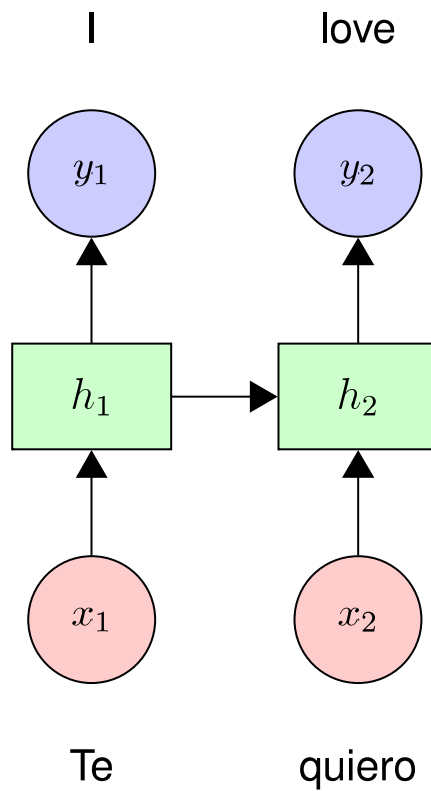


Illustration: Rico Sennrich

Architectures for machine translation

- RNNs are good for **sequence labeling** tasks
 - Part-of-speech tagging, ...
 - Each word corresponds to exactly one output label
 - This doesn't work for machine translation
- Let's try a different approach...
- **Language modeling**: predict the next word
 - For each word there is exactly one next word (except when the sentence is finished...)

Traditional language models

- A sentence T of length n is a sequence $w_1 \dots w_n$
- Sentence probability:

$$p(T) = p(w_1, \dots, w_n)$$

- Chain rule:

$$p(T) = \prod_{i=1}^n p(w_i | w_1, \dots, w_{i-1})$$

- Markov assumption (n-gram model):

$$p(T) = \prod_{i=1}^n p(w_i | w_{i-k}, \dots, w_{i-1})$$

Sequential data

Long-distance dependencies

- Agreement in number, gender, etc.
 - **He** does not have very much confidence in **himself**.
 - **She** does not have very much confidence in **herself**.
- Selectional preference
 - The **reign** has lasted as long as the life of the **queen**.
 - The **rain** has lasted as long as the life of the **clouds**.
- Coreference
 - The trophy would not fit in the brown suitcase because it was too **big**.
 - The trophy would not fit in the brown suitcase because it was too **small**.

Language models

- Traditional language models need really large n-gram sizes to cover long distance dependencies
 - See examples
- We can use RNNs for language modeling
 - The hidden layers keep track of the previous words – potentially back to the beginning of the sentence
 - No hard cutoff after n characters

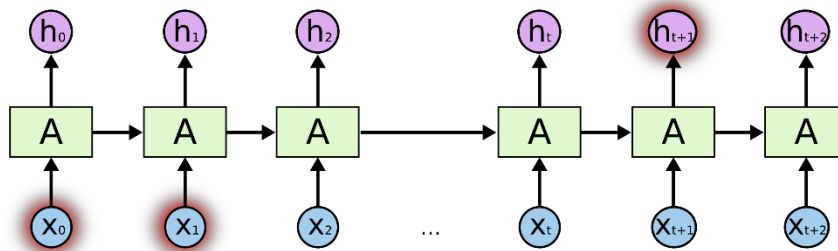
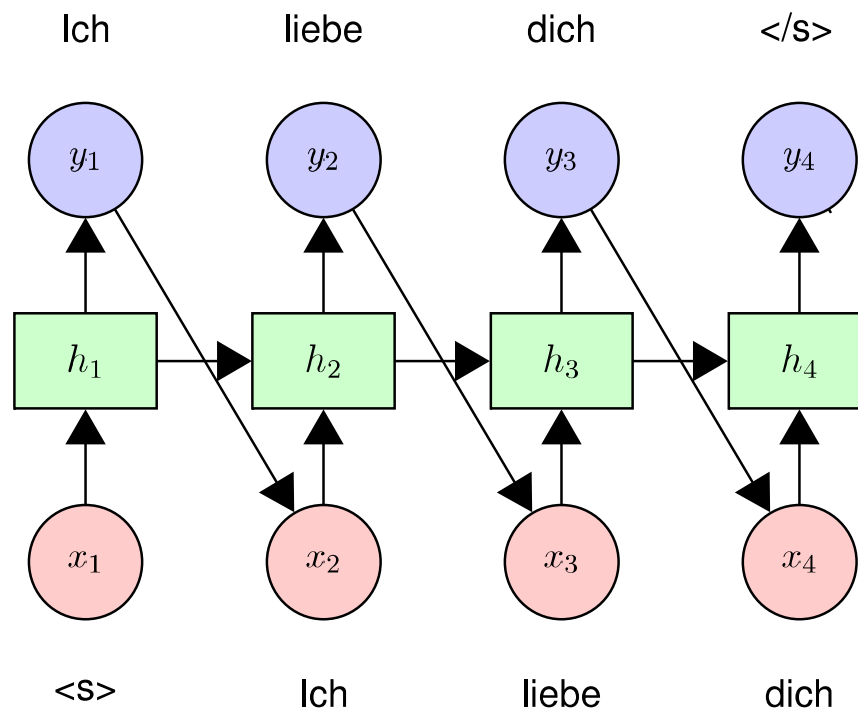


Illustration:

<http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

RNN language models



How does this relate to MT?

- Language models generate text
- **Conditioned language models** do not just generate text – they generate text according to some specification

Input	Output (Text)	Task
Document	Short description	Summarization
Question	Answer	Question answering
Image	Text	Image captioning
Speech signal	Transcription	Speech recognition
English text	Japanese text	Machine translation

Conditioned language models

- Suppose we have:

- a source sentence S of length m : x_1, \dots, x_m
- a target sentence T of length n : y_1, \dots, y_n

- Translation probability:

$$p(T|S) = p(y_1, \dots, y_n | x_1, \dots, x_m)$$

- Chain rule:

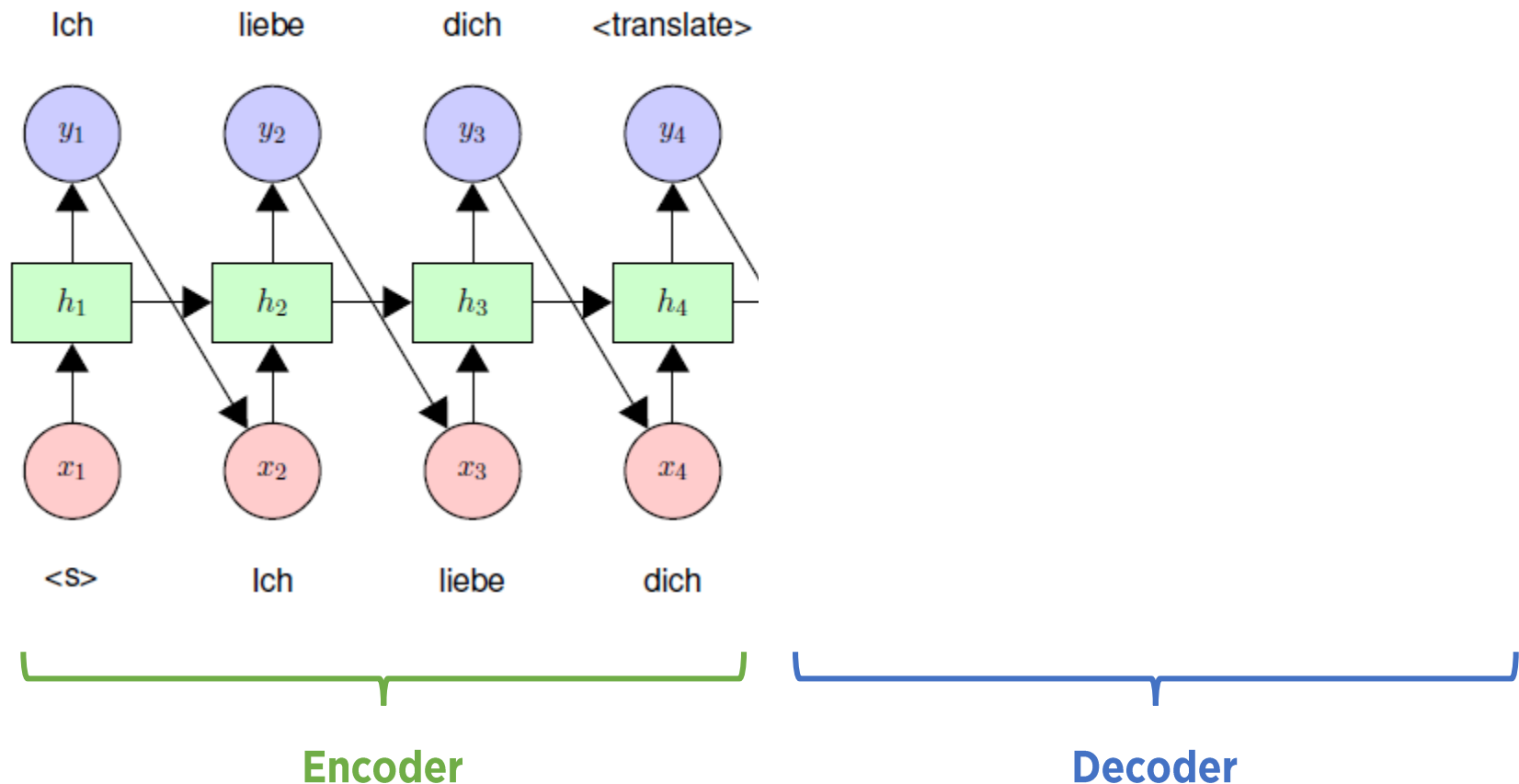
$$p(T|S) = \prod_{i=1}^n p(y_i | x_1, \dots, x_m, y_1, \dots, y_{i-1})$$

- Let's just treat the sentence pair T, S as one long sequence...

Conditioned language models

This type of RNN is called **encoder-decoder model**.

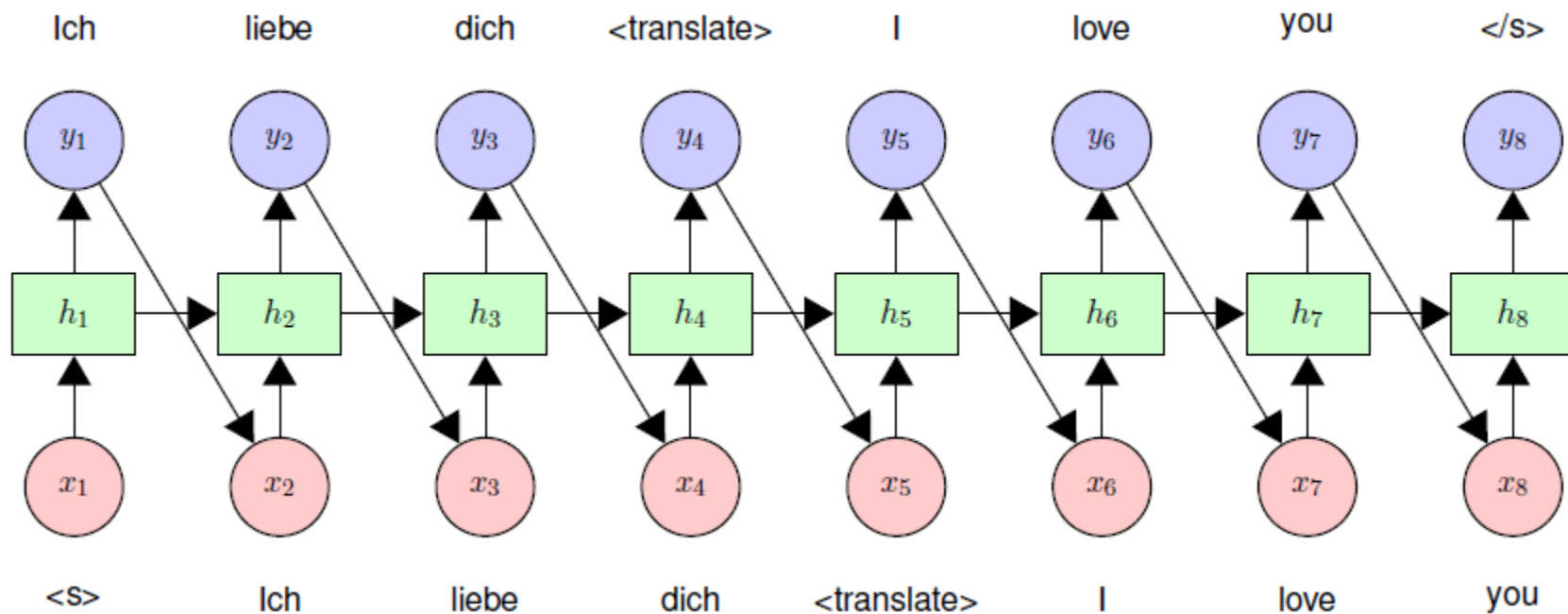
No restrictions on output order and length.
The model will decide by itself when “it is done”.



Conditioned language models

We are making the task harder than it needs to be.
We do not really care about the source sentence.

Any potential problem
with this approach?



Practical considerations

Computing

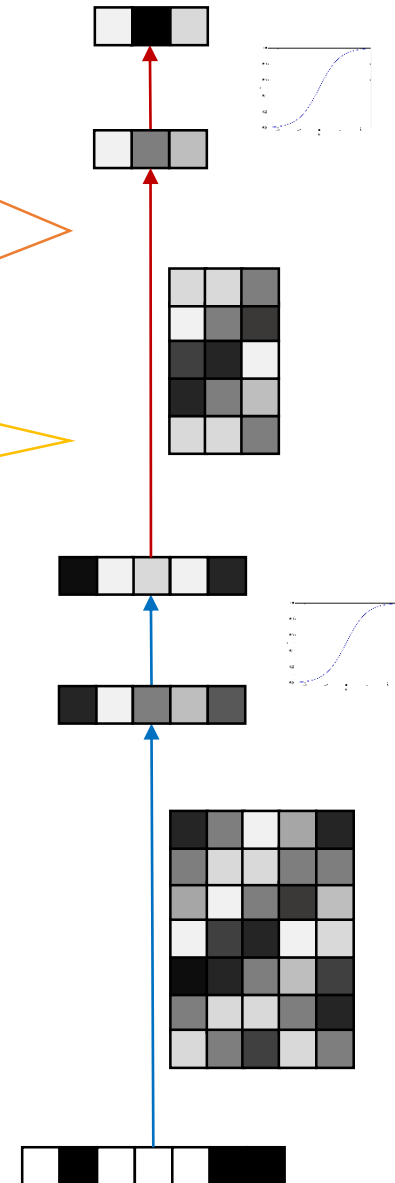
Training:

- Forward pass
- Loss computation
- Backward pass

Prediction:

- Forward pass

- A lot of computations as networks grow bigger
 - But quite simple computations
 - Can be parallelized easily
- Solution:
 - Perform computations on GPU



Graphics processing units

- Very efficient at manipulating computer graphics and image processing



CPU, like a motorcycle



Quick to start, top speed
not shabby

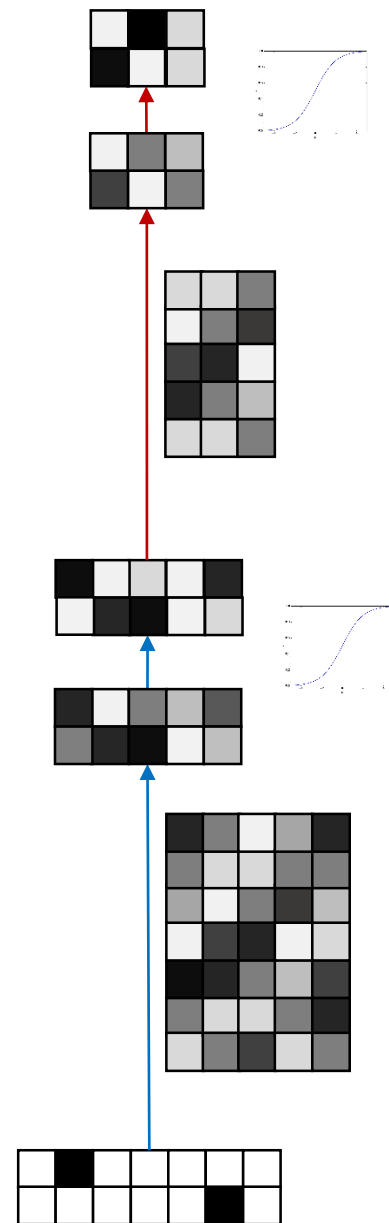
GPU, like an airplane



Takes forever to get off the
ground, but super-fast
once flying

Minibatching

- Run several examples (words / sentences) through the network at the same time
 - Increased parallelization, thus speed
 - Fewer weight updates during training, thus more stable
 - Increased memory requirements
- Minibatch size determined according to above criteria



Large vocabularies

Large vector: tens of thousands of possible output words

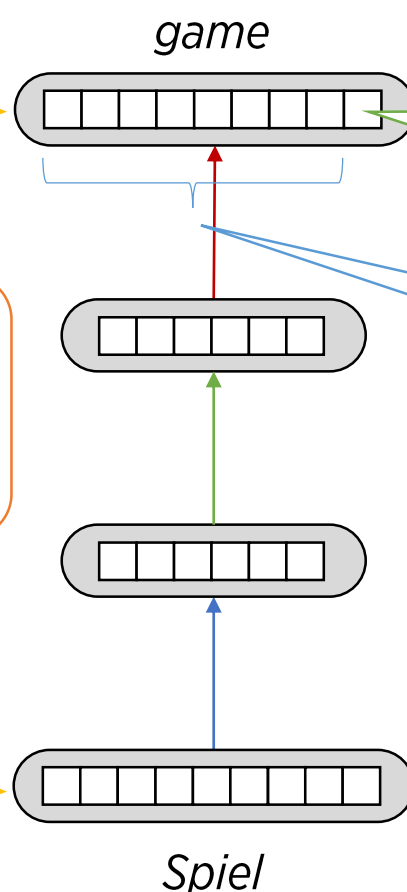
Reserve one unit for rare and unknown words: **<unk>**

Impossible to include all words of a language:

- Memory limitations
- Zipfian distribution

Limit vector to n most frequent words

Large vector: tens of thousands of possible input words



Large vocabularies

We'll talk about better solutions later...

- Examples English – Czech:

- The author Stephen Jay Gould died 20 years after diagnosis.

Autor <unk> <unk> <unk> zemřel 20 let po <unk>.

- As the Reverend Martin Luther King Jr. said fifty years ago:

Jak řekl reverend Martin <unk> King <unk> před padesáti lety:

- Her 11-year-old daughter, Shani Bart, said it felt a "little bit weird" [...] back to school.

Její <unk> dcera <unk> <unk> řekla, že je to "trochu divné", [...] vrací do školy.

Decisions...

- **Model architecture:**

- Source/target vocabulary size
- Number, sizes of hidden layers
- Type of recurrent layers
- Directionality of recurrent layers

- **Training:**

- Initialization of weight matrices
- Minibatch size
- Learning and optimization algorithms
- Number of epochs/iterations, stopping criterion

Readings

- Mikel Forcada (2017): *Making sense of neural machine translation*. Translation Spaces 6(2).
 - PDF available within university network, link on Moodle
- Philipp Koehn (2017): *Neural machine translation*. Appendix chapter for SMT book.
 - PDF available online, link on Moodle
 - Sections 13.1 – 13.4