NATURAL LANGUAGE PROCESSING WITH RECURRENT NEURAL NETWORKS, SEQ2SEQ MODELS, ATTENTION AND TRANSFORMERS

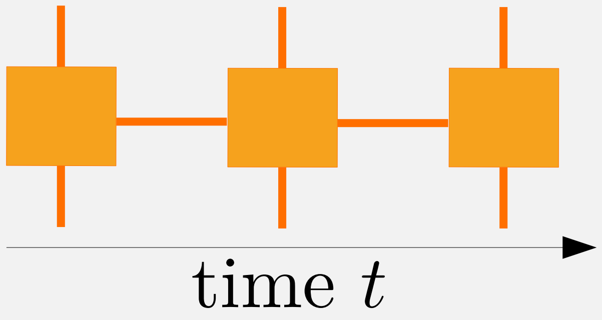
Korbinian Kottmann
Quantum Optics Theory group
Quantum Information Theory group
ICFO, Castelldefels

/Qottmann
@Qottmann

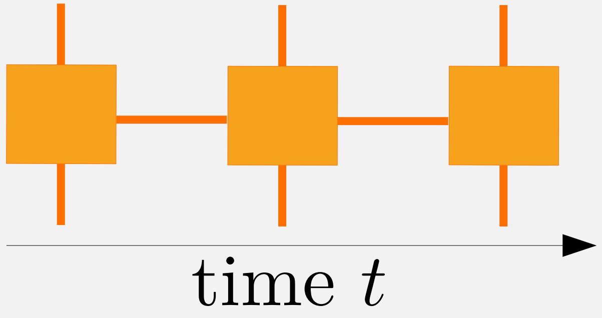
ICFO
UNIÓ EUROPEA
Fons Europeu
de Desenvolupament Regional

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

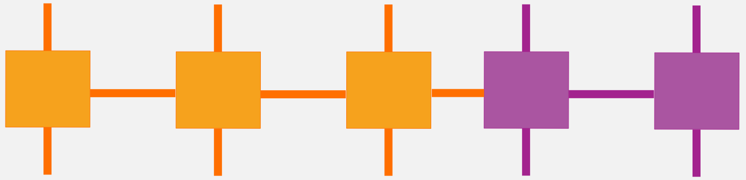
1. RECURRENT NEURAL NETWORKS



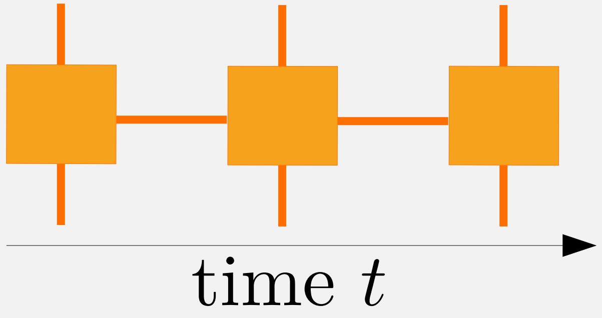
1. RECURRENT NEURAL NETWORKS



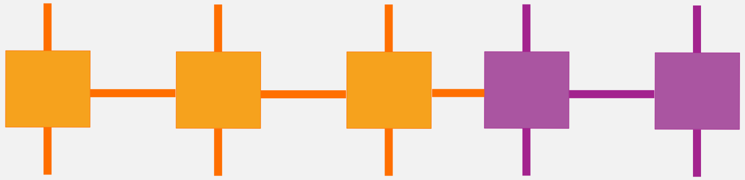
2. SEQ 2 SEQ MODELS



1. RECURRENT NEURAL NETWORKS



2. SEQ 2 SEQ MODELS

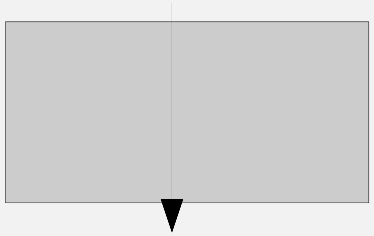


3. TRANSFORMERS

RECURRENT NEURAL NETWORKS

Natural Language as a stochastic process

I like to play football



Me gusta jugar futbol

Translation

Natural Language as a stochastic process

I like to play football

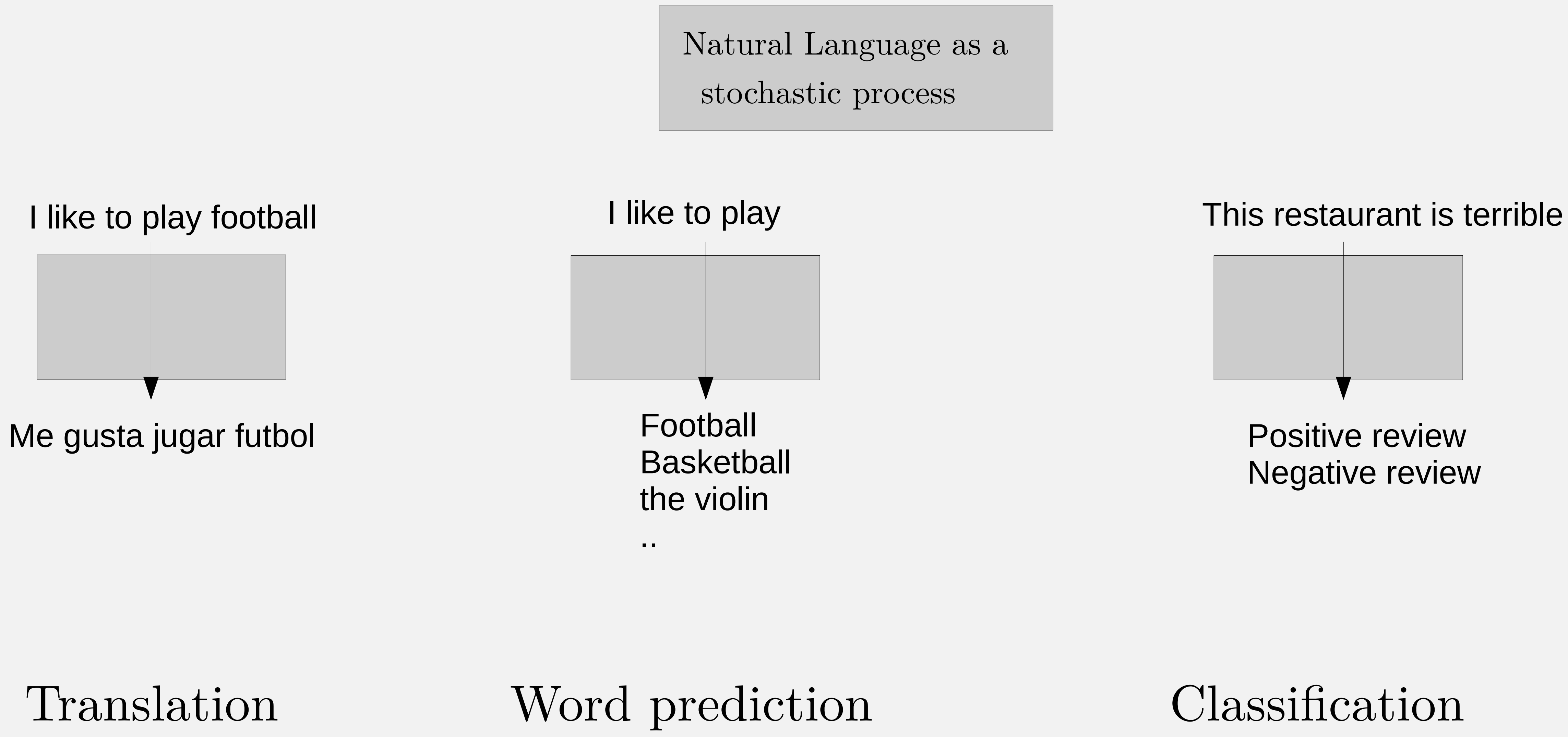
Me gusta jugar futbol

Translation

I like to play

Football
Basketball
the violin
..

Word prediction


Natural Language as a stochastic process
I like to play football
Me gusta jugar futbol
Translation
I like to play
Football
Basketball
the violin
..
Word prediction
This restaurant is terrible
Positive review
Negative review
Classification

1. RECURRENT NEURAL NETWORKS

Output: conditional probability
for every word in the output
vocabulary

Input: *Sentence*
Sentence is a sequence of *words*
Words are elements of a *vocabulary*

General ML jargon:
Sequences consisting of token

1. RECURRENT NEURAL NETWORKS

Output: conditional probability for every word in the output vocabulary

Input: *Sentence*
Sentence is a sequence of *words*
Words are elements of a *vocabulary*

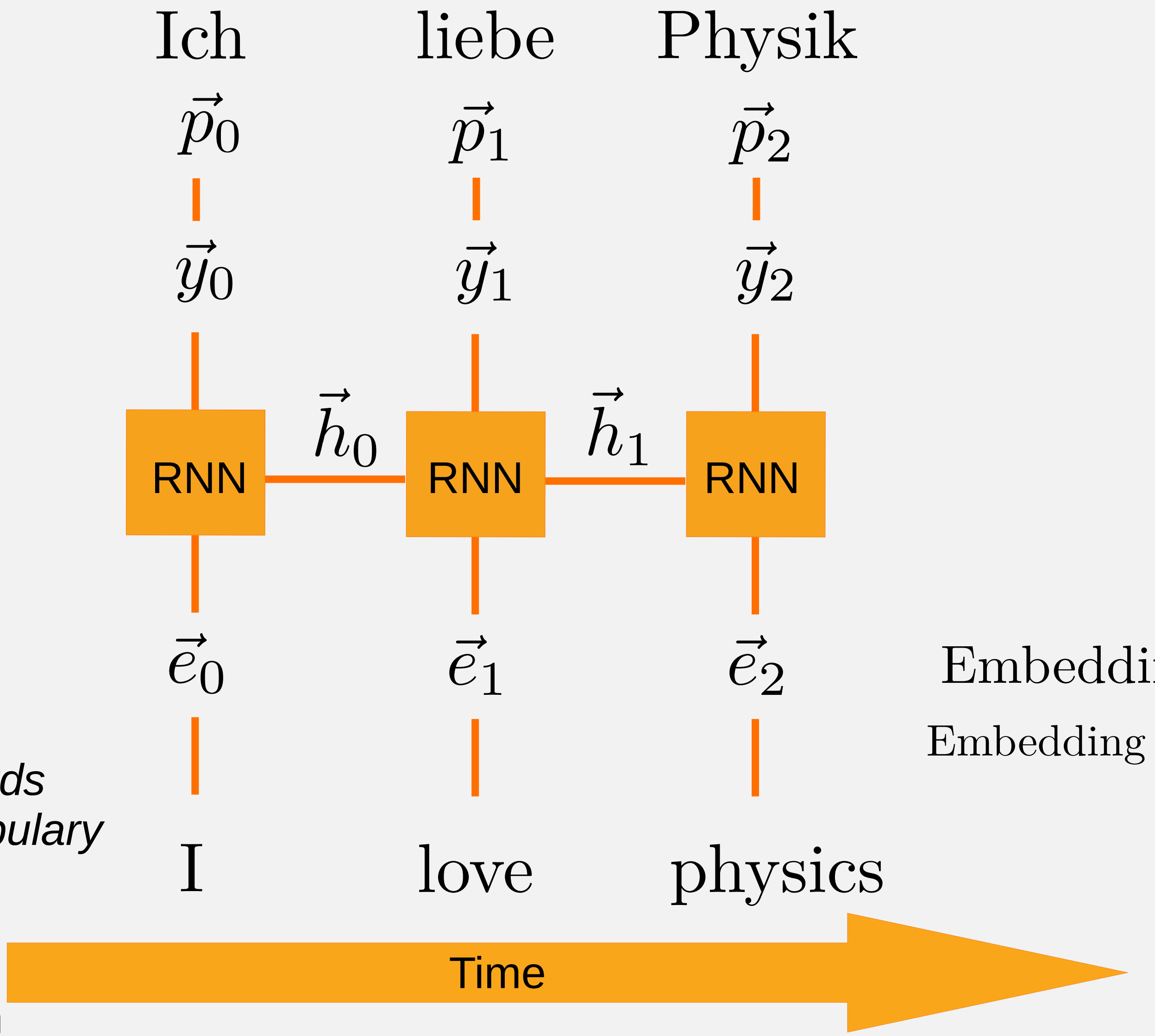
$$\text{Embedding} : \text{id}_{\text{word}} \mapsto \vec{e} \in \mathbb{R}^{d_e}$$

Embedding dim d_e (hyper parameter)

General ML jargon:
Sequences consisting of token

1. RECURRENT NEURAL NETWORKS

Output: conditional probability for every word in the output vocabulary



$$\text{Embedding} : \text{id}_{\text{word}} \mapsto \vec{e} \in \mathbb{R}^{d_e}$$

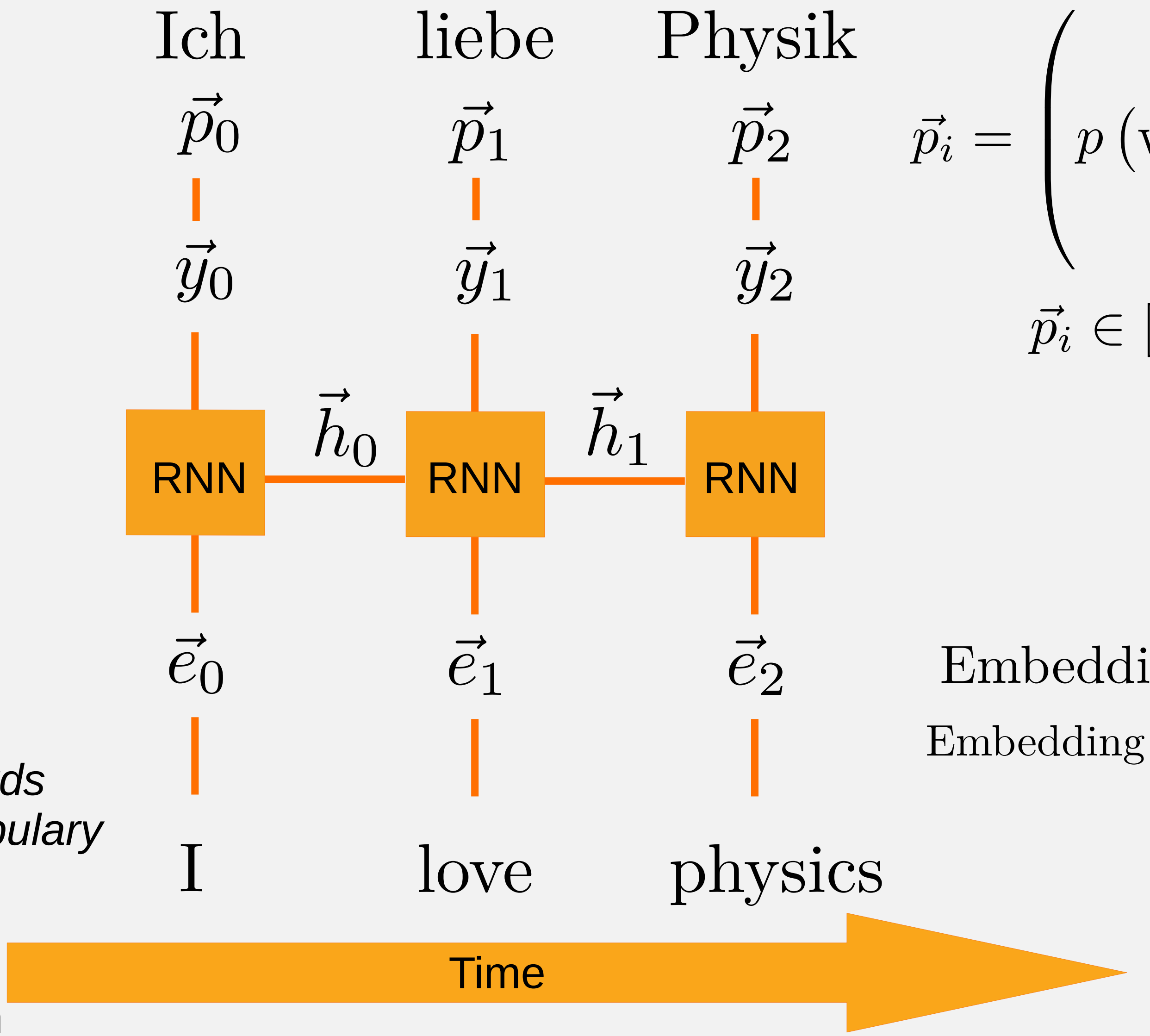
Embedding dim d_e (hyper parameter)

Input: *Sentence*
Sentence is a sequence of *words*
Words are elements of a *vocabulary*

General ML jargon:
Sequences consisting of token

1. RECURRENT NEURAL NETWORKS

Output: conditional probability for every word in the output vocabulary



$$\vec{p}_i = \begin{pmatrix} \vdots \\ p\left(\mathrm{word}_i^{\mathrm{out}} | \mathrm{word}_{j<i}^{\mathrm{in}}\right) \\ \vdots \end{pmatrix}$$

$$\vec{p}_i \in [0,1]^{|\mathrm{output\ vocab.}|}$$

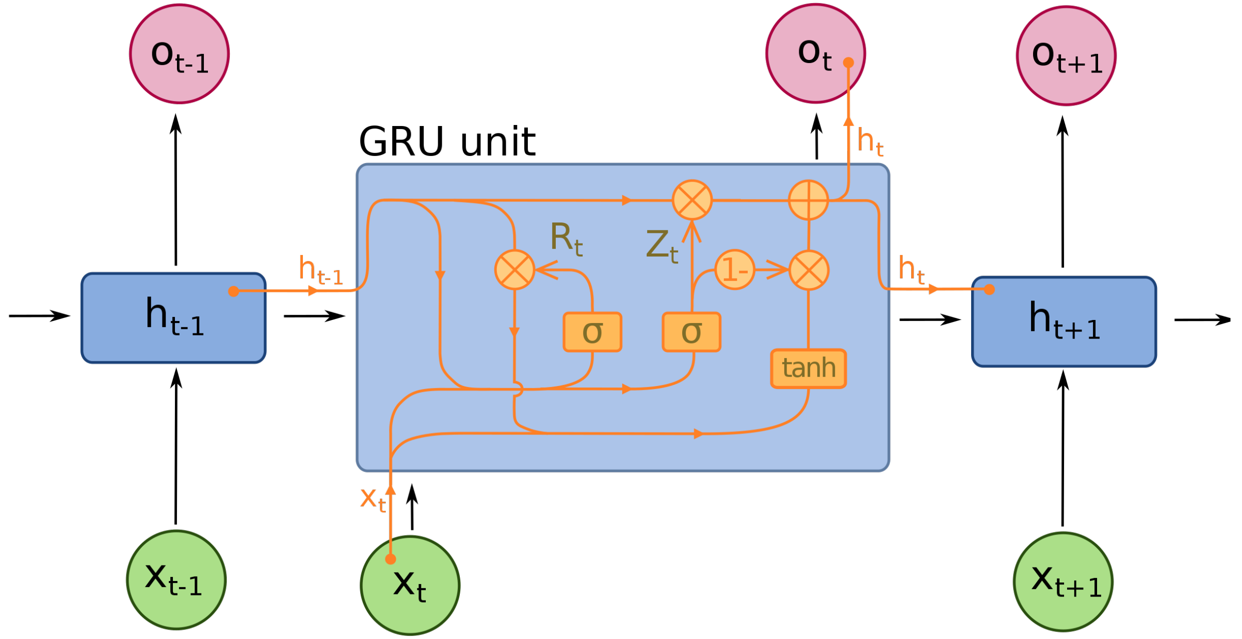
$$\mathrm{Embedding} : \mathrm{id}_{\mathrm{word}} \mapsto \vec{e} \in \mathbb{R}^{d_e}$$

Embedding dim d_e (hyper parameter)

Input: *Sentence*
Sentence is a sequence of *words*
Words are elements of a *vocabulary*

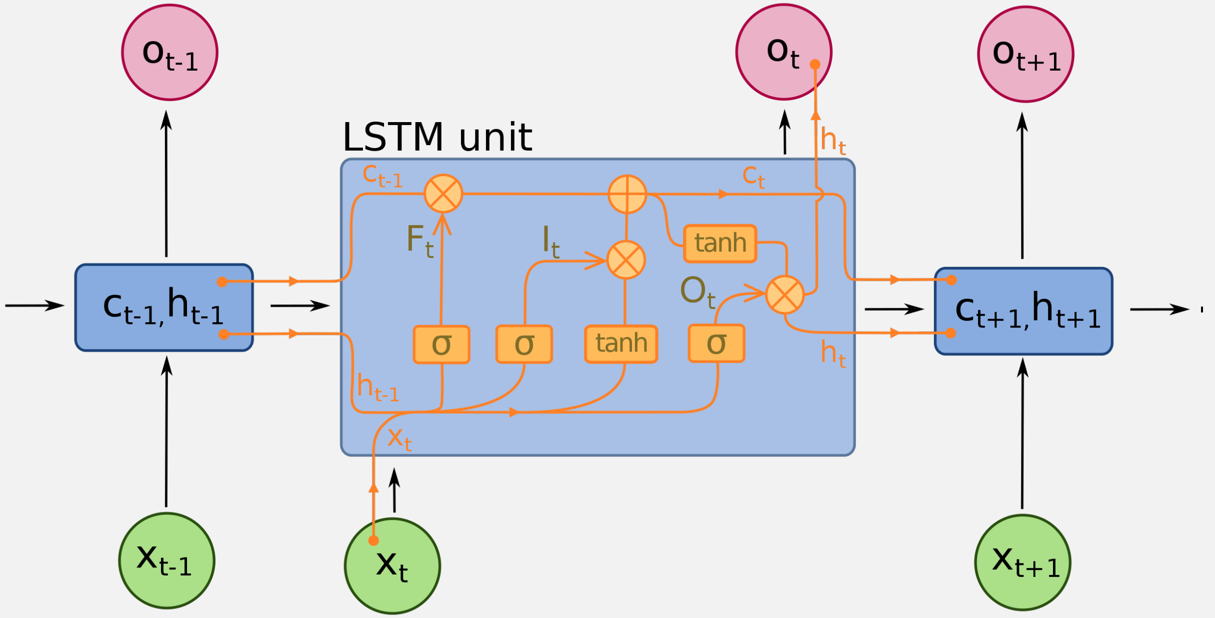
General ML jargon:
Sequences consisting of token

1. RECURRENT NEURAL NETWORKS



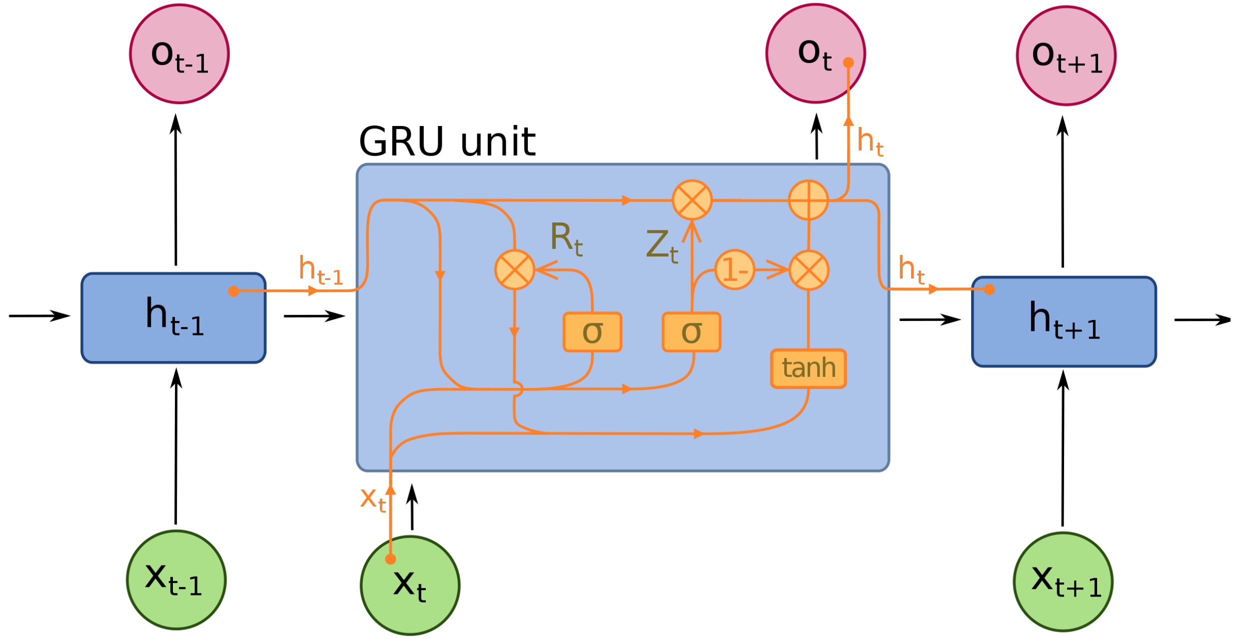
Gated Recurrent Unit
(Kyunghyun Cho et al. arXiv:1409.1259 2014)

Long Short-Term Memory
Sepp Hochreiter; Jürgen Schmidhuber (1997)



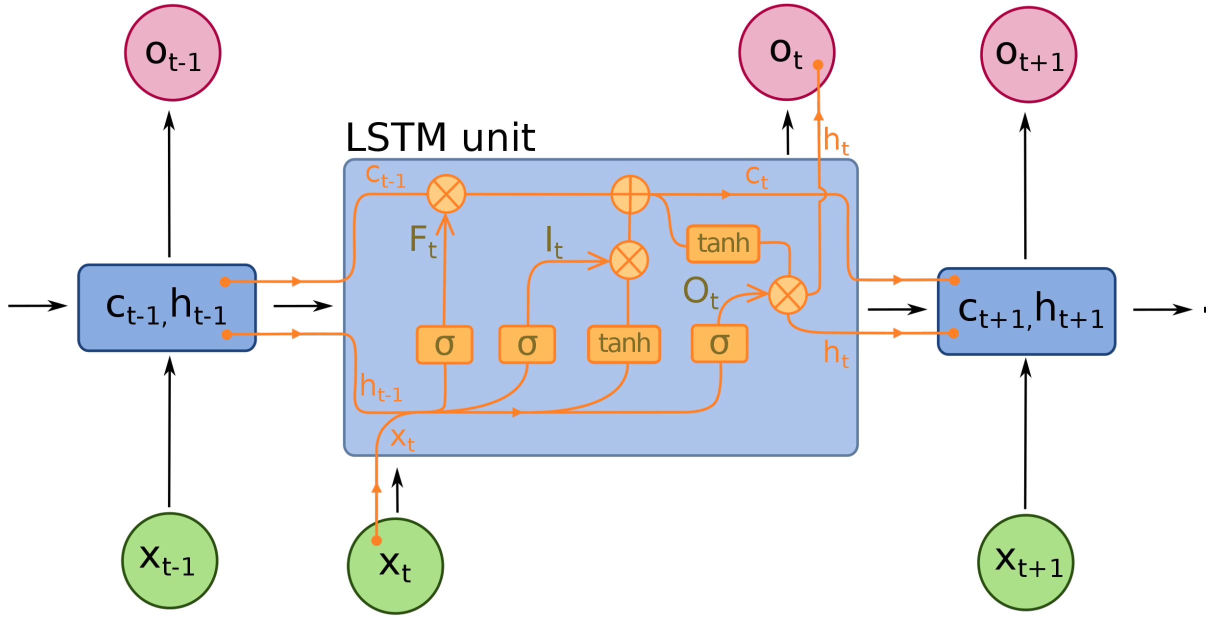
https://en.wikipedia.org/wiki/Recurrent_neural_network

1. RECURRENT NEURAL NETWORKS



Gated Recurrent Unit
(Kyunghyun Cho et al. arXiv:1409.1259 2014)

Long Short-Term Memory
Sepp Hochreiter; Jürgen Schmidhuber (1997)

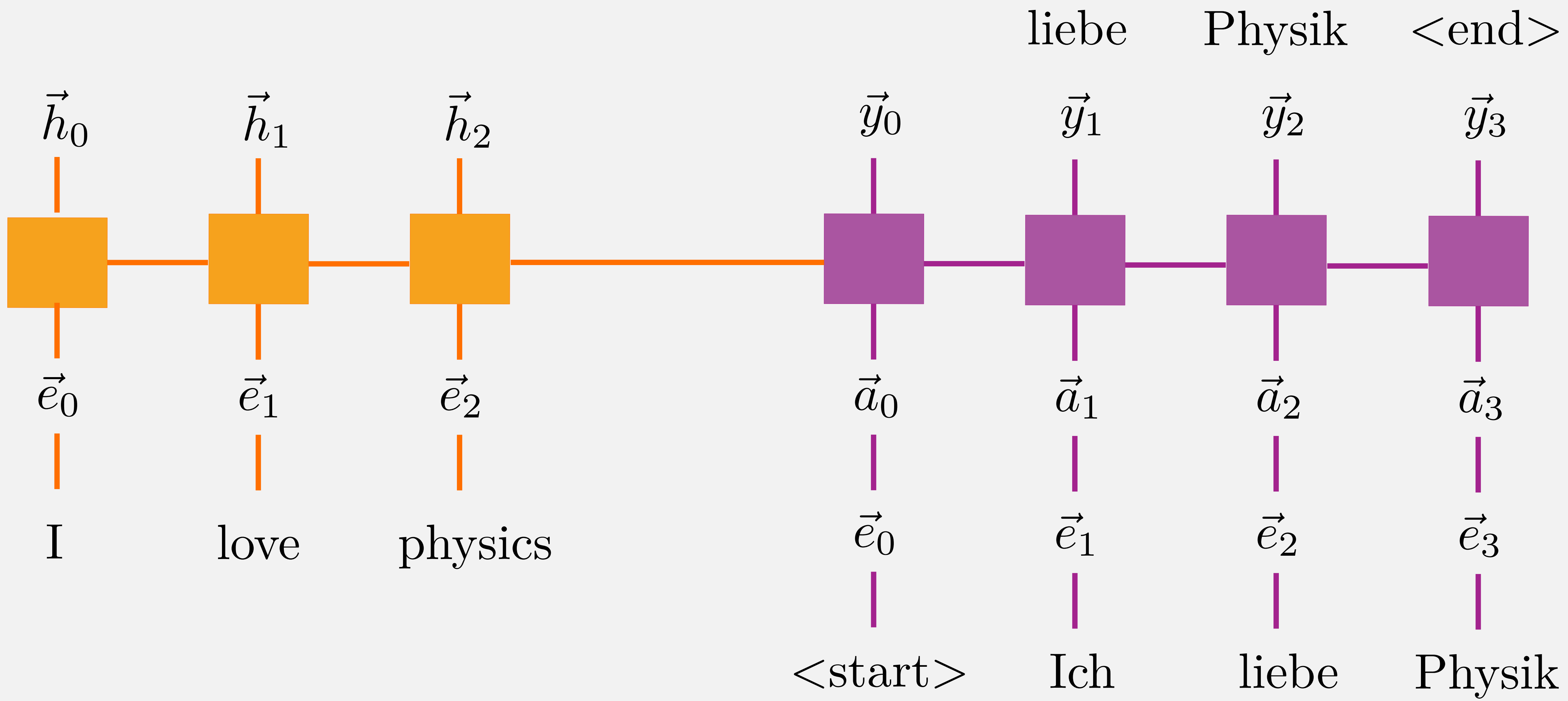


https://en.wikipedia.org/wiki/Recurrent$_{n}eural_{n}etwork$

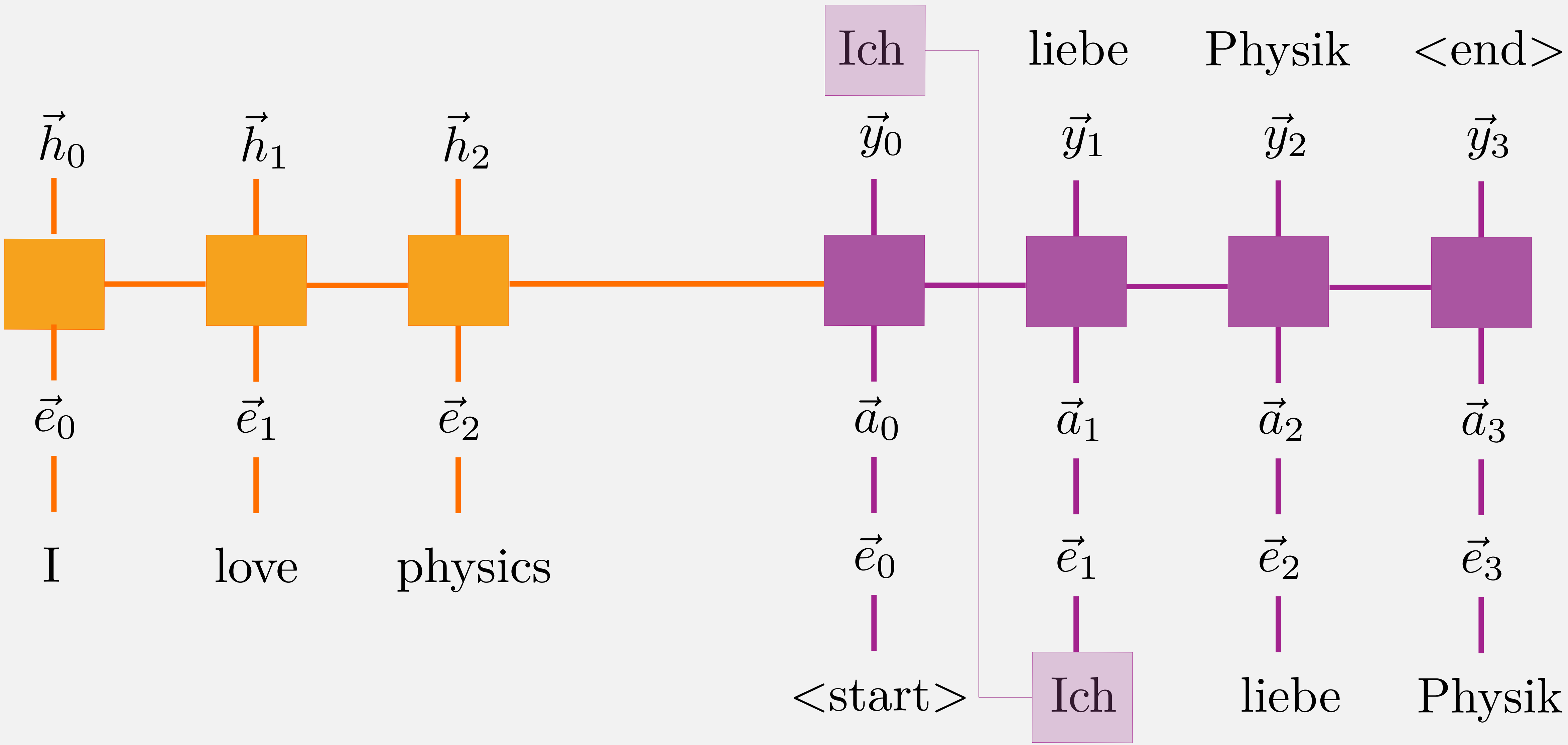
1. RECURRENT NEURAL NETWORKS

Problems with RNNs:
1. In translation, only sentences with same input and output length
2. Hidden state $\vec{h}_i$ is supposed to carry the sentiment of the whole sentence

2. SEQ-2-SEQ MODELS

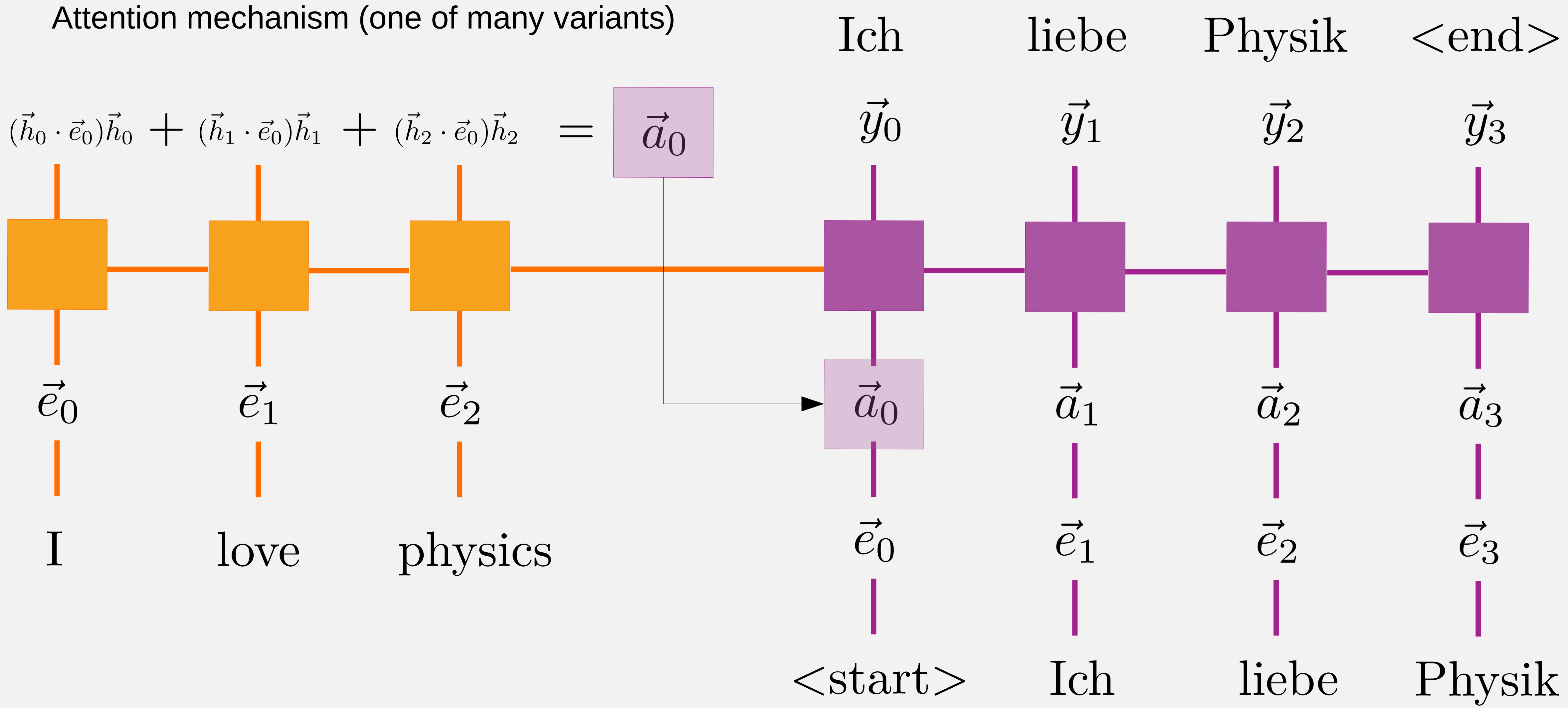


2. SEQ-2-SEQ MODELS



2. SEQ-2-SEQ MODELS

Attention mechanism (one of many variants)

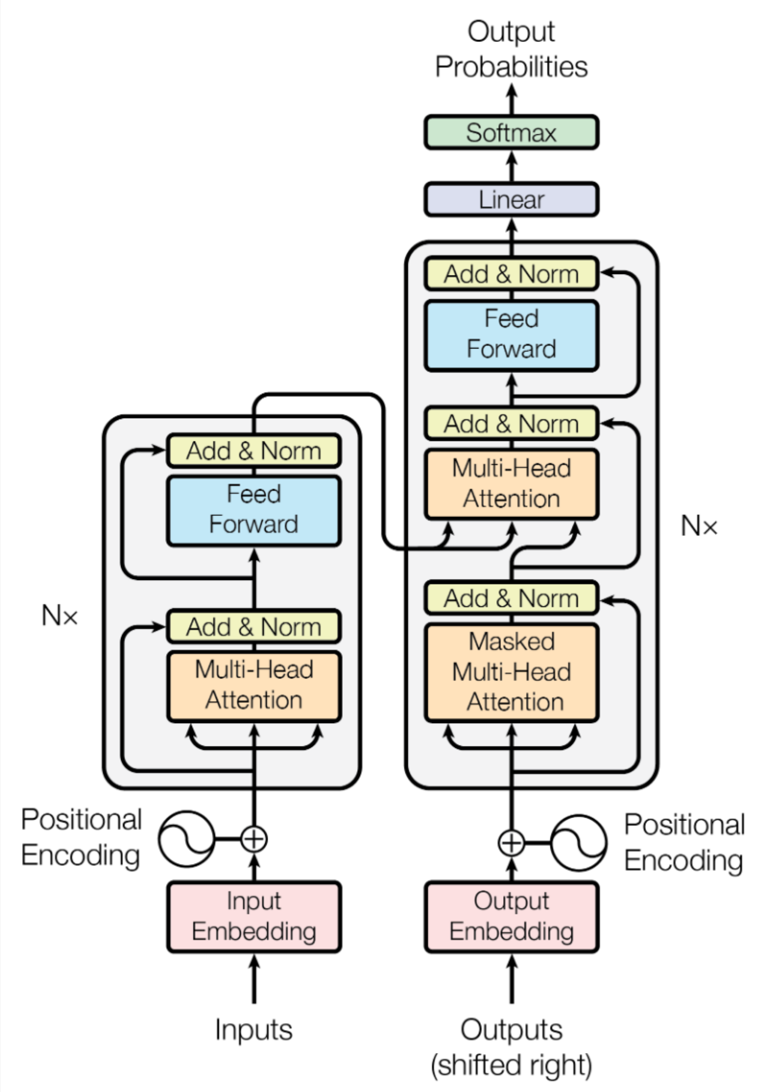


2. SEQ-2-SEQ MODELS

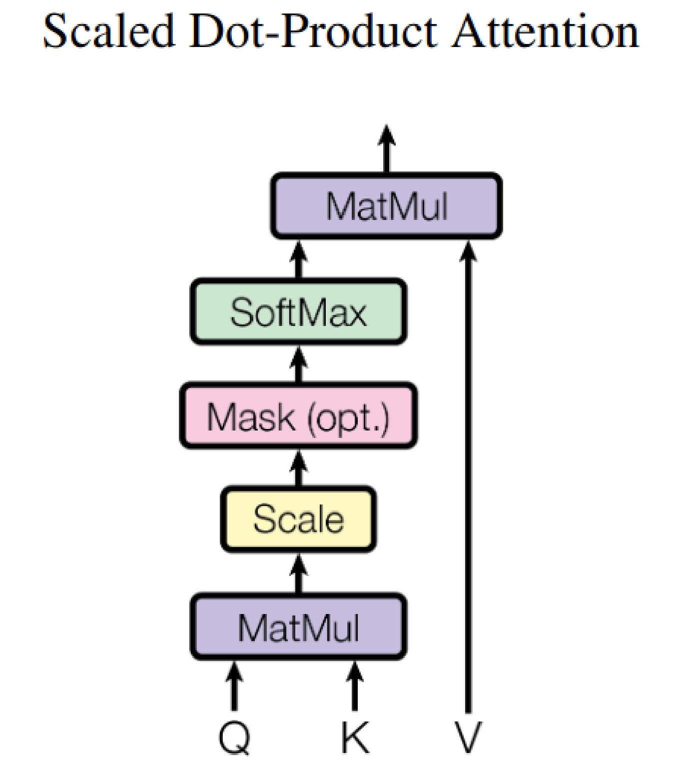
Problems with seq-2-seq models:
- Sequential processing not parallelizable and therefore slow training

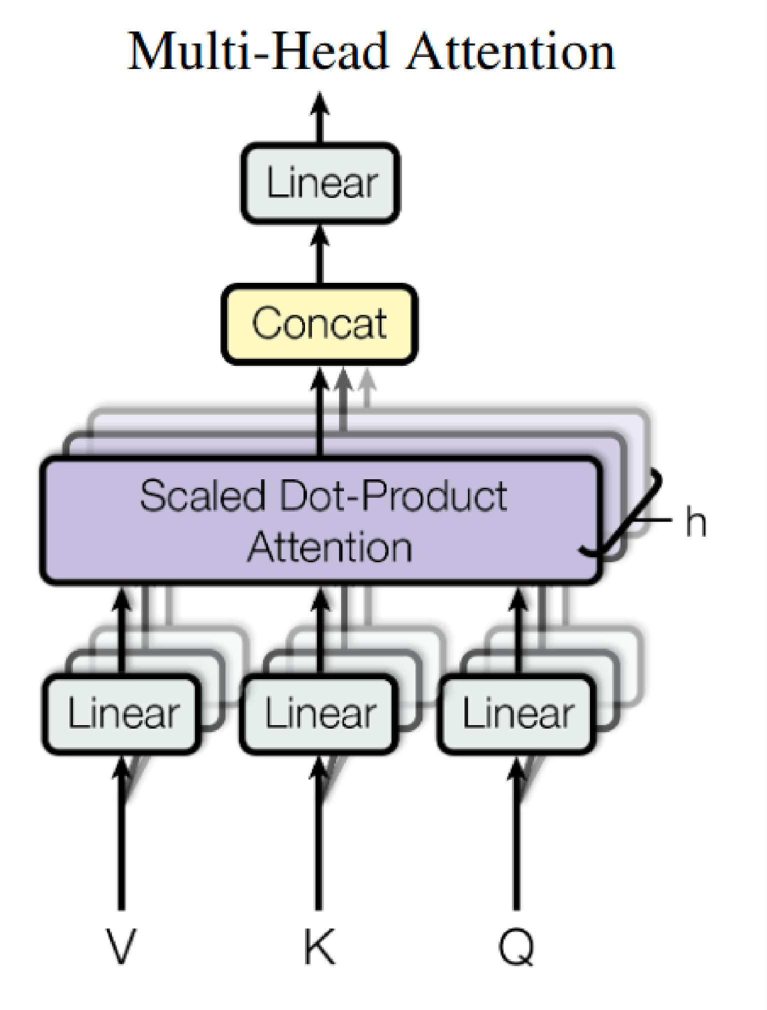
3. TRANSFORMERS

Vaswani et al “Attention is all you need” arxiv:1706.03762 2017



3. TRANSFORMERS





Closer look, always Q = K,
So this is just all possible
Combinations of dot-products
Between inputs = attention scores

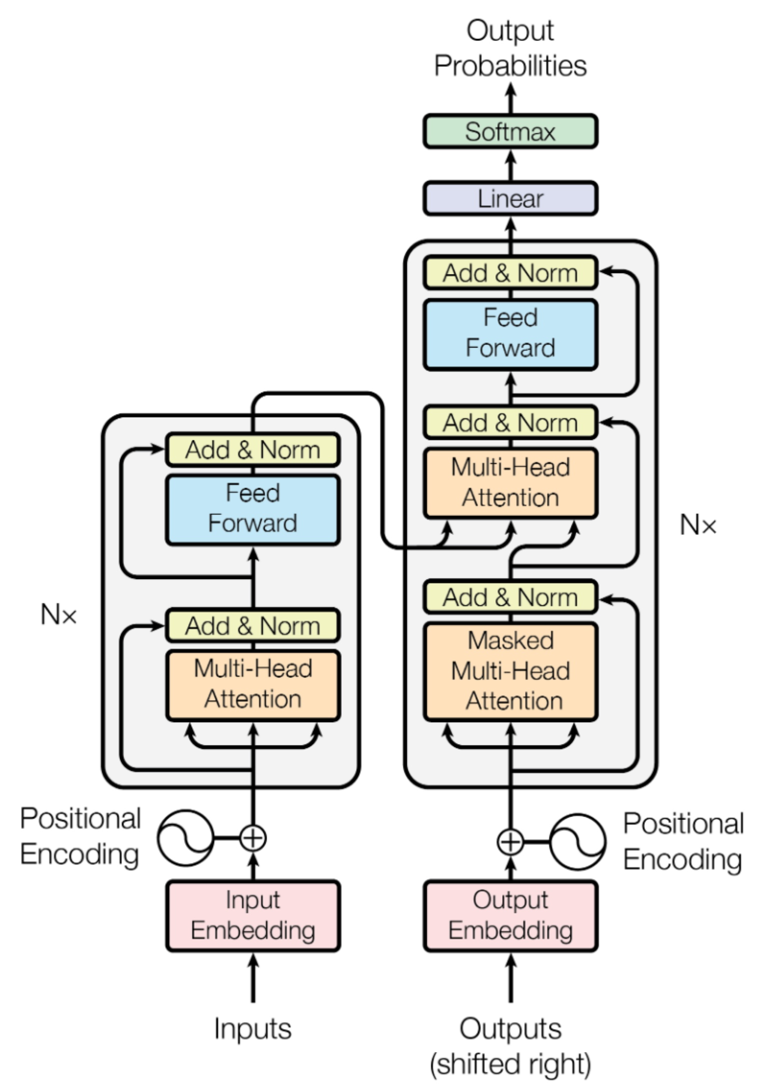
$$\mathrm{Attention}(Q, V, K) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

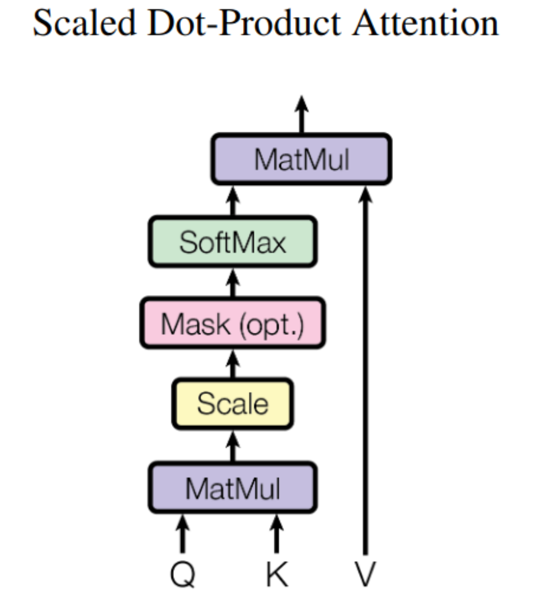
Convex combination of all value vectors

Vaswani et al "Attention is all you need" arxiv:1706.03762 2017

3. TRANSFORMERS

Vaswani et al “Attention is all you need” arxiv:1706.03762 2017





$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

$$\vec{e}_{\text{pos}} \mapsto \vec{e}_{\text{pos}} + \vec{\text{PE}}$$

Vaswani et al “Attention is all you need” arxiv:1706.03762 2017

3. TRANSFORMERS

Special token

`<unk>`, `<pad>`, `<bos>`, `<eos>`

```
tokenize(["Hello", "my", "name", "is", "Korbinian", "who", "are", "you"])
>> [<bos>, 3, 45, 23, 14, <unk>, 23, 66, 90, <eos>, <pad>, <pad>, ...]
```

Open problems:
Extra-long sequences: e.g. “Longformer” https://arxiv.org/pdf/2004.05150.pdf (ongoing research)

No reasoning due to a lack of abstraction level. These language models do not ‘understand’ language, but rather become extremely well at mimicing it (like a very sophisticated parrot).

3. TRANSFORMERS

Nonetheless, they are commercially used and very successful due to the large amounts of data that can be efficiently processed with them

BERT (language model)

From Wikipedia, the free encyclopedia

Bidirectional Encoder Representations from Transformers (**BERT**) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google.[1][2] In 2019, Google announced that it had begun leveraging BERT in its search engine, and by late 2020 it was using BERT in almost every English-language query. A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model.[3]

Generative Pre-trained Transformer 3 (**GPT-3**) is an autoregressive language model that uses deep learning to produce human-like text.

It is the third-generation language prediction model in the GPT-n series (and the successor to GPT-2) created by OpenAI, a San Francisco-based artificial intelligence research laboratory.[2] GPT-3's full version has a capacity of 175 billion machine learning parameters. GPT-3, which was introduced in May 2020, and was in beta testing as of July 2020,[3] is part of a trend in natural language processing (NLP) systems of pre-trained language representations.[1]

MACHINE LEARNING AND QUANTUM PHYSICS

THESIS PROPOSAL

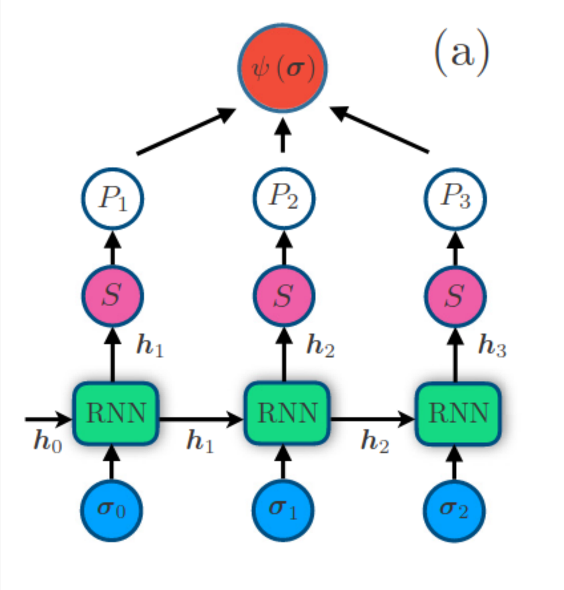
arXiv:2003.09905
Korbinian Kottmann, Patrick Huembeli,
Maciej Lewenstein, Antonio Acin

/Qottmann @Qottmann

UNIÓ EUROPEA
Fons Europeu
de Desenvolupament Regional

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA Y COMPETITIVIDAD

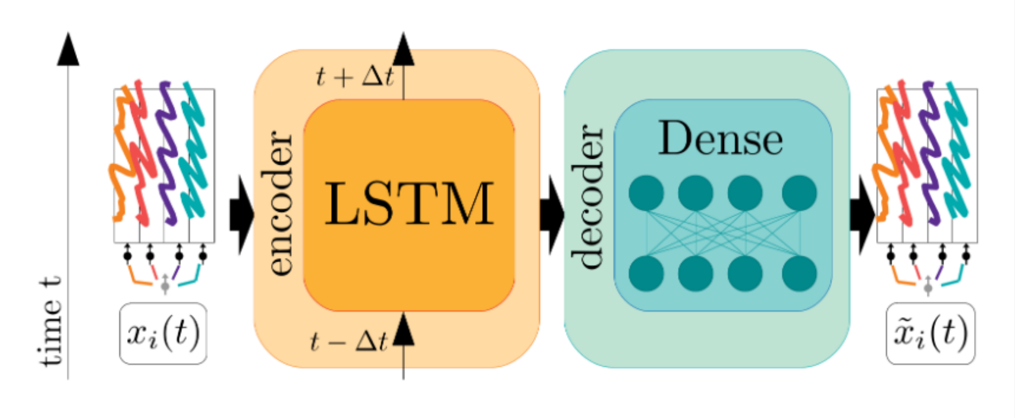
4. AUTOREGRESSIVE MODELS IN PHYSICS

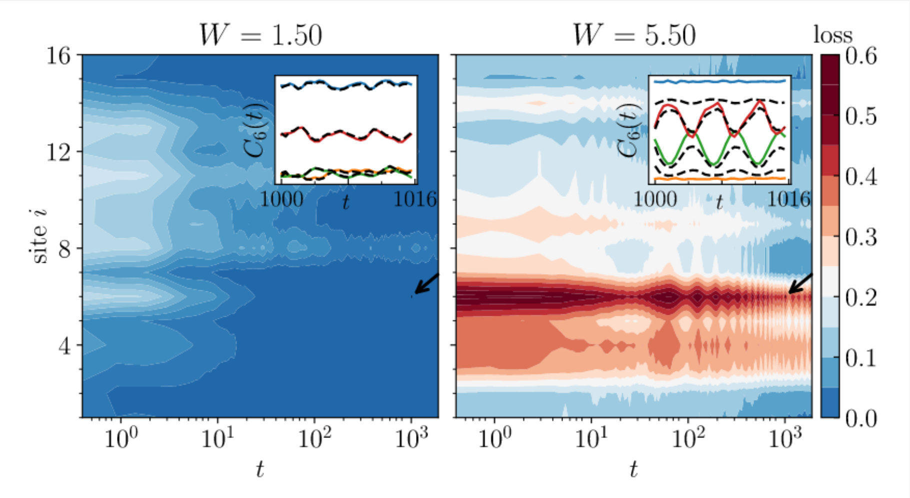


Mohamed Hibat-Allah, Martin Ganahl, Lauren E. Hayward, Roger G. Melko, and Juan Carrasquilla
“Recurrent neural network wave functions”

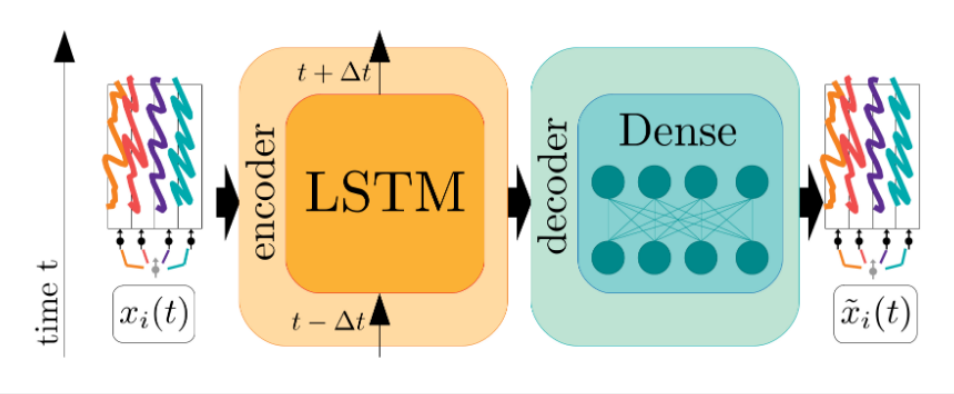
Phys. Rev. Research 2, 023358 – Published 17 June 2020

https://doi.org/10.1103/PhysRevResearch.2.023358





4. AUTOREGRESSIVE MODELS IN PHYSICS

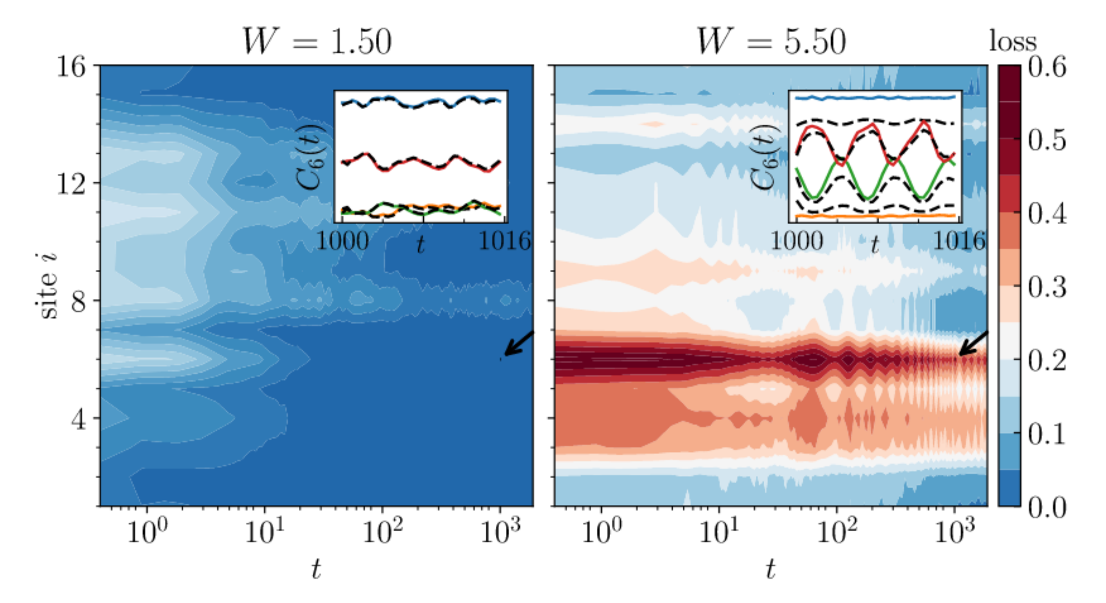


Tomasz Szołdra, Piotr Sierant, Korbinian Kottmann, Maciej Lewenstein, and Jakub Zakrzewski

“Detecting ergodic bubbles at the crossover to many-body localization using neural networks”

Phys. Rev. B 104, L140202 arxiv:2106.01811

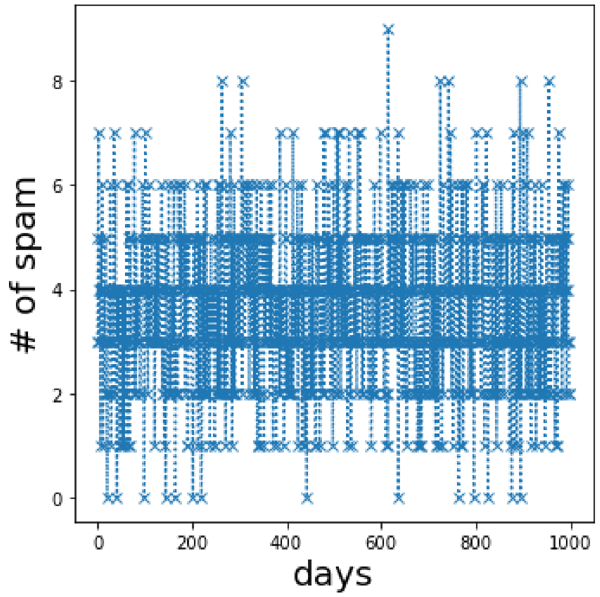
https://doi.org/10.1103/PhysRevB.104.L140202

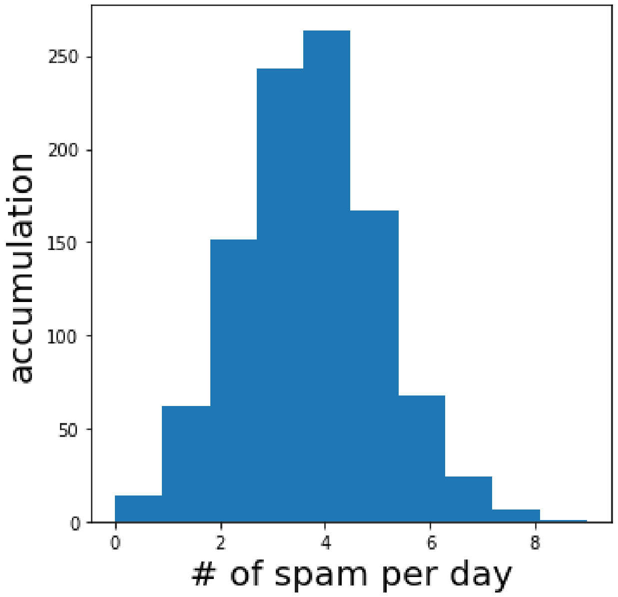


0. MAX LIKELIHOOD METHOD

Imagine you are receiving a fix number of n=10 emails a day. There is a chance of Θ that it is spam. You record the number of spam emails a day over the course of N=1000 days and this is what you obtain.

How do you determine Θ?



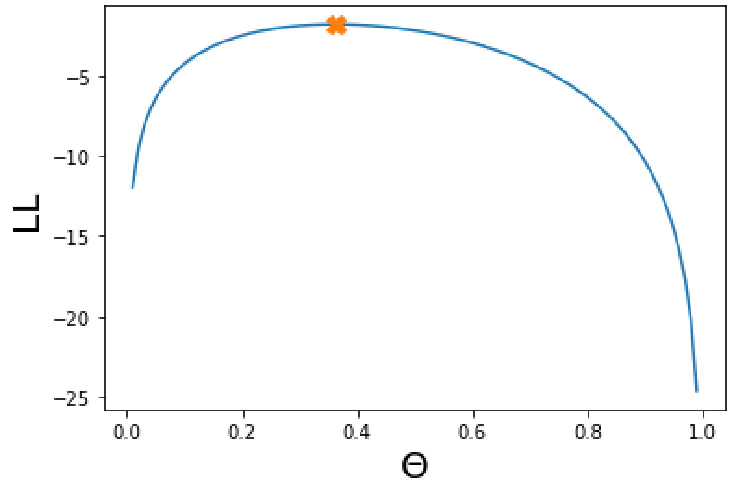


Binomial distribution

$$p_\Theta(x) = \binom{n}{x} \Theta^x (1-\Theta)^{n-x}$$

Max (Log)Likelihood method

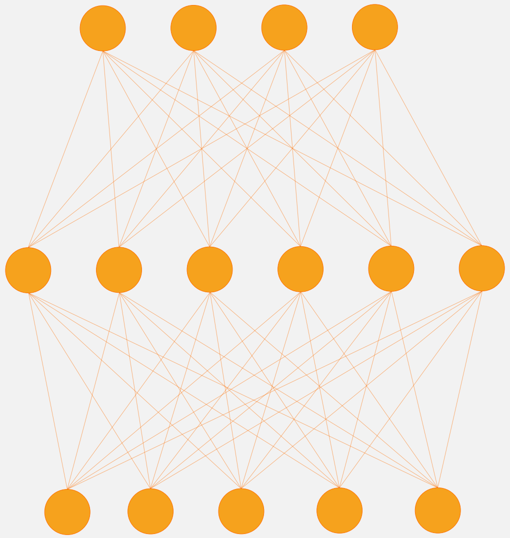
$$\max_\Theta \mathrm{LL} = \max_\Theta \sum_{i=1}^{N} \log(p_\Theta(x_i))$$



0. MAX LIKELIHOOD METHOD

stochastic process

True distribution $p_{\text{true}}(x)$ $\approx$ $p_\Theta(x)$



x

Obtain ideal Θ by $\max_\Theta \sum_{x \in \mathcal{X}} \log(p_\Theta(x))$

Log-Likelihood

When functional form of $p_\Theta(x)$ is not known, we can utilize Neural Networks!