Friedrich-Alexander-Universität
Technische Fakultät

Machine Learning
Data Analytics

Machine Learning for Time Series

Dr. Emmanuelle Salin
Machine Learning and Data Analytics (MaD) Lab
Friedrich-Alexander-Universität Erlangen-Nürnberg
09.01.2025

Topics Overview

1. Time series fundamentals and definitions (Part 1)
2. Time series fundamentals and definitions (Part 2)
3. Bayesian Inference and Gaussian Processes
4. State space models (Kalman Filters)
5. State space models (Particle Filters)
6. Autoregressive models
7. Data mining on time series
8. Deep Learning (DL) for Time Series (Introduction to DL)
9. DL – Convolutional models (CNNs)
10. DL – Recurrent models (RNNs and LSTMs)
11. DL – Attention-based models (Transformers)
12. DL – From BERT to ChatGPT
13. DL – New Trends in Time Series processing
14. Time series in the real world

Topics Overview

1. Time series fundamentals and definitions (Part 1)
2. Time series fundamentals and definitions (Part 2)
3. Bayesian Inference and Gaussian Processes
4. State space models (Kalman Filters)
5. State space models (Particle Filters)
6. Autoregressive models
7. Data mining on time series
8. Deep Learning (DL) for Time Series (Introduction to DL)
9. DL – Convolutional models (CNNs)
10. DL – Recurrent models (RNNs and LSTMs)
11. **DL – Attention-based models (Transformers)**
12. DL – From BERT to ChatGPT
13. DL – New Trends in Time Series processing
14. Time series in the real world

Recap: Recurrent Neural Networks

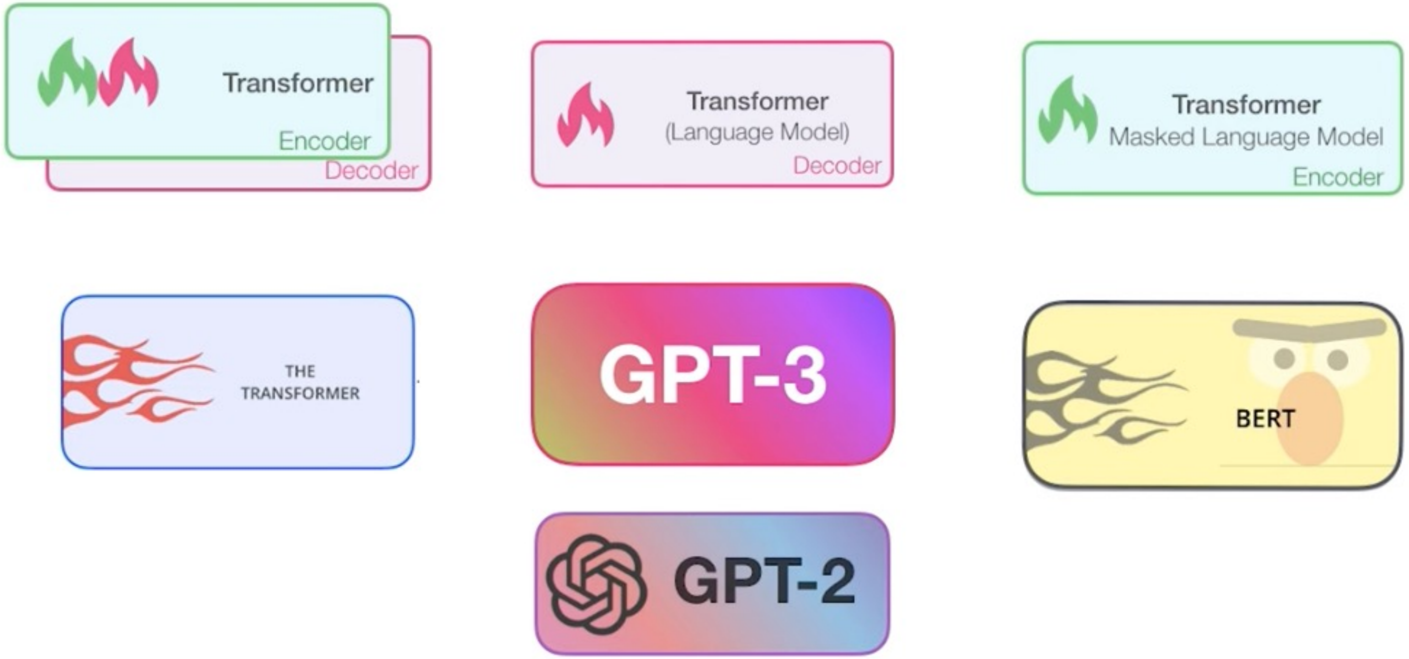
RNN/LSTM can model:

- Sequences of variable lengths
- Long term dependencies (using LSTMs)

However:

- Non-parallelism → Long training time
- Large memory usage
- Difficult to train (vanishing/exploding gradients)

Motivation



- Transformers *perceive* the entire sequence at the same time.
- They are based on the seq2seq concept, i.e., transforming sequences into other sequences.
- State of the art in many NLP tasks.

https://www.youtube.com/watch?v=-QH8fRhqFHM&t=422s

In This Lecture...

- **Introduction to Natural Language Processing**
- **The Attention Mechanism**
- **The Transformers Architecture**
- **Trasformer-based Models and their Applications**

Deep Learning for Time Series – Attention Models

Introduction to Natural Language Processing

Natural Language Processing

Natural Language Processing (NLP) is a subfield of AI that focuses on developing algorithms to **process**, **generate**, and **understand** human languages.

Applications:

- Machine Translation
- Question Answering
- Grammatical Analysis
- Image Captioning
- Information Extraction
- …

➢ In practice, it concerns any task where **input** and/or **output** involves language.

Natural Language Processing: Data

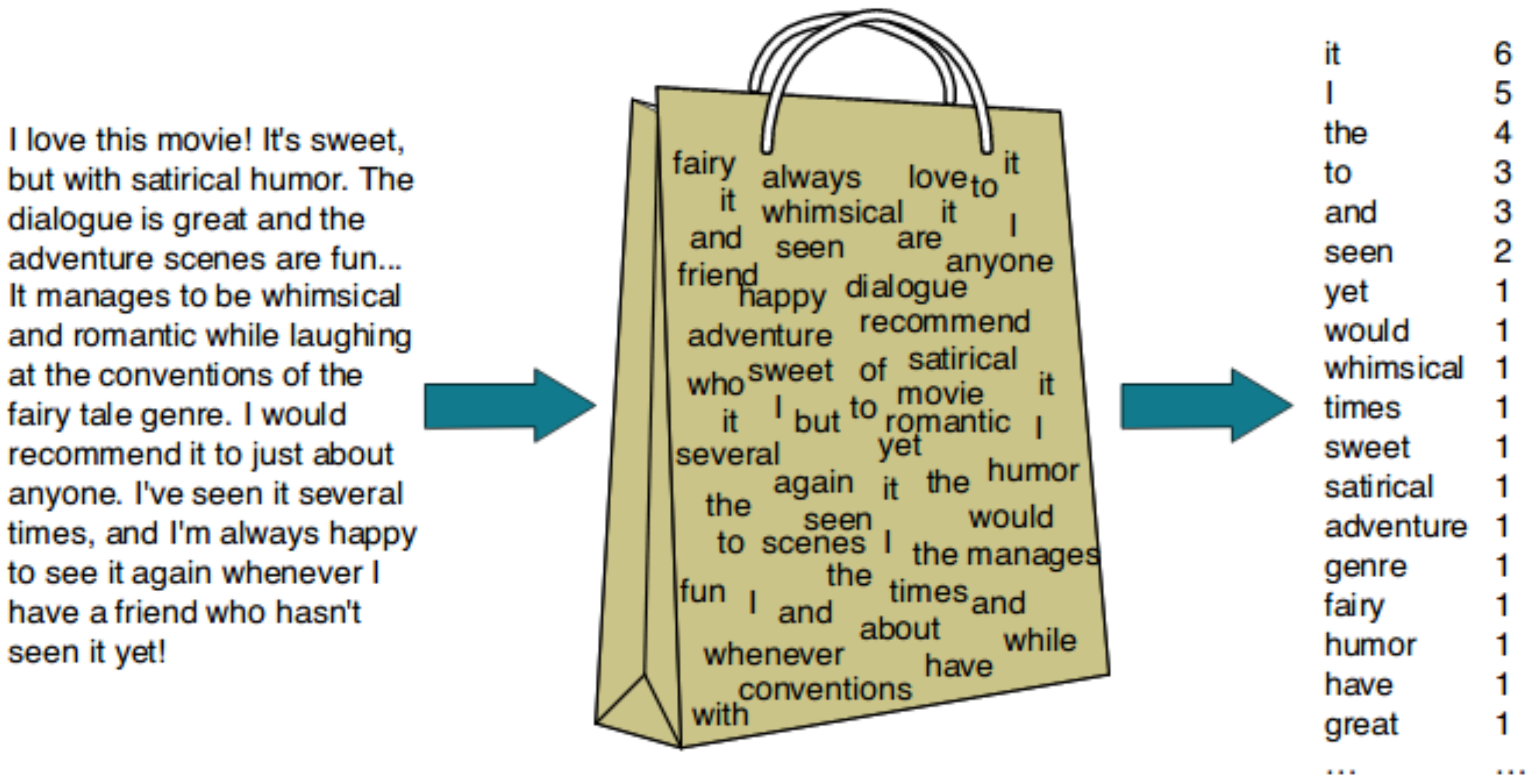
Natural Language Processing models can be used for **text**, **audio** or **multimodal** data. Due to the abundance of freely-available text data, NLP models and applications often focus on text data.

- ➢ Similarly to time series data, text data is **sequential** and involves **temporal dependencies** (and often long range dependencies)

- ➢ However, contrary to most time series, text data is **not composed of numeric values** but words or characters (tokens). The process to get sequences of words or characters from a text is called **tokenization.**

Modeling Words: the Bag of Words Model

Bag of words (BOW) models resentent text using an **unordered** collection of words.



I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
...	...

Image from https://ogre51.medium.com/nlp-explain-bag-of-words-3b9fc4f211e8

Language Models

A **language model** is a model that predicts the probability of a sequence of words or the next word in a sequence.

Deep learning (e.g. LSTMs) can be used to train complex models that extract relevant features. They are can be trained using the **autoregressive language modelling** task:

Goal: Predict $\mathrm{P}(x^{t+1} | \, x^{t}, x^{t-1}, \dots, x^{0})$, with x^{i} the word at position i

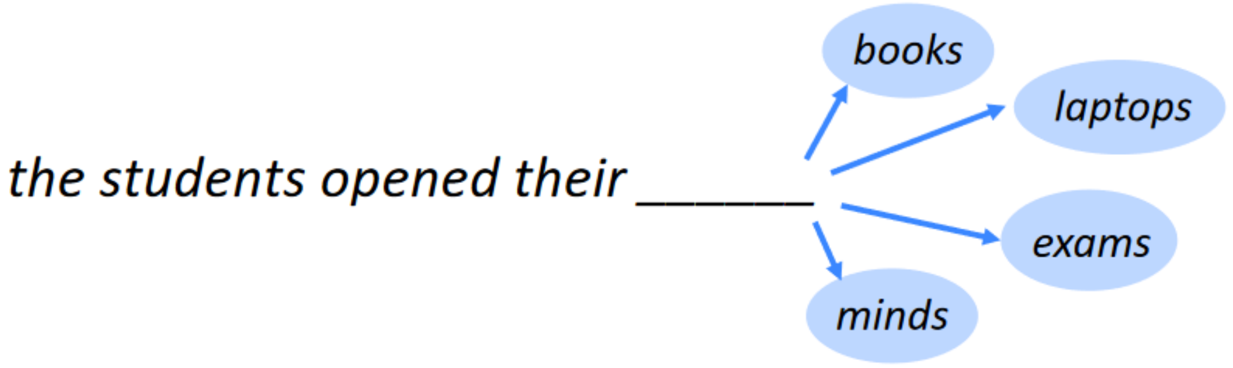
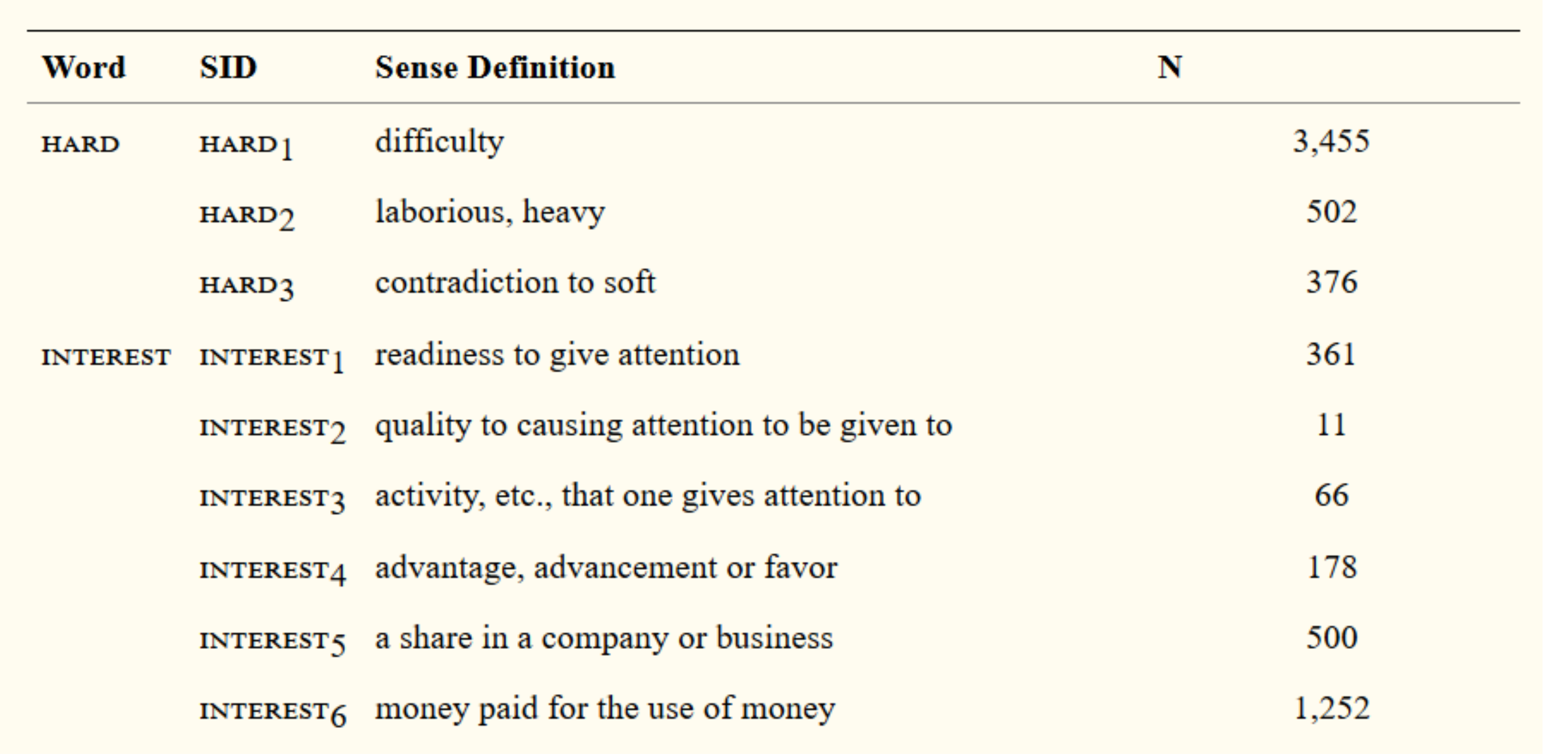


Image from https://medium.com/@antonio.lopardo/the-basics-of-language-modeling-1c8832f21079

Contextual Language Models

Contextual language models are language models that represent word differently depending on their context.

It can help understand **sentence structure**, **negation**, **word sense disambiguation**.



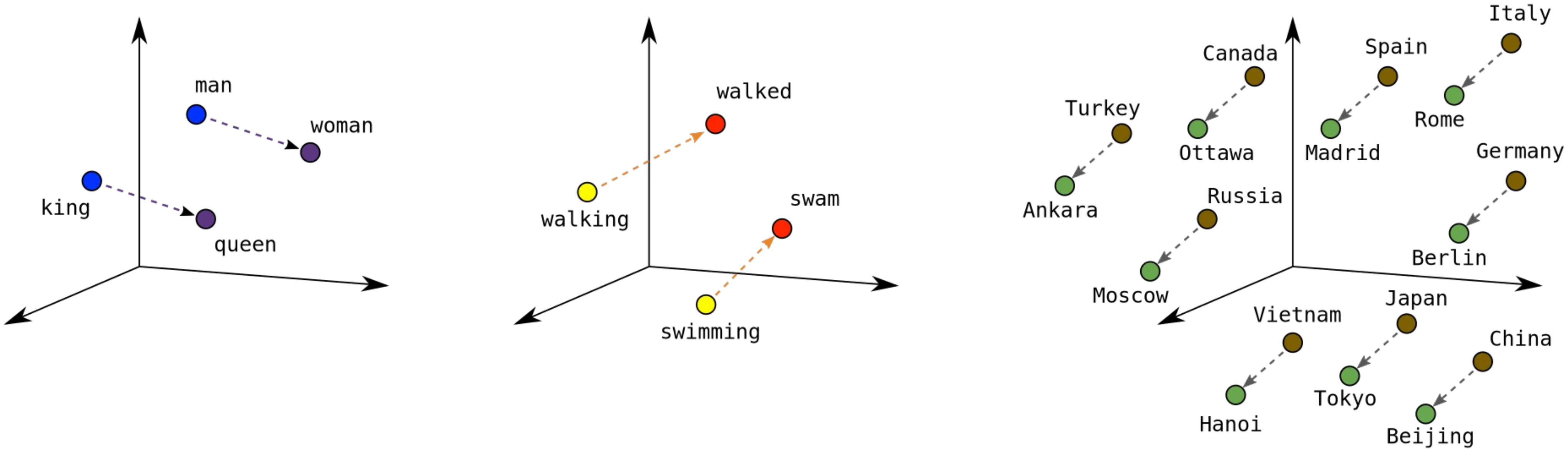
Word	SID	Sense Definition	N
HARD	HARD1	difficulty	3,455
	HARD2	laborious, heavy	502
	HARD3	contradiction to soft	376
INTEREST	INTEREST1	readiness to give attention	361
	INTEREST2	quality to causing attention to be given to	11
	INTEREST3	activity, etc., that one gives attention to	66
	INTEREST4	advantage, advancement or favor	178
	INTEREST5	a share in a company or business	500
	INTEREST6	money paid for the use of money	1,252

Examples from Hongfang Liu, Virginia Teller, Carol Friedman, A Multi-aspect Comparison Study of Supervised Word Sense Disambiguation, *Journal of the American Medical Informatics Association*, Volume 11, Issue 4, July 2004, Pages 320–331, https://doi.org/10.1197/jamia.M1533

Word Representations: Word Embeddings

Embeddings refers to **dense high-dimensional vectors** used to represent words in a continuous space.
They can represent sub-words to limit **vocabulary size** (*walking -> walk #ing)*

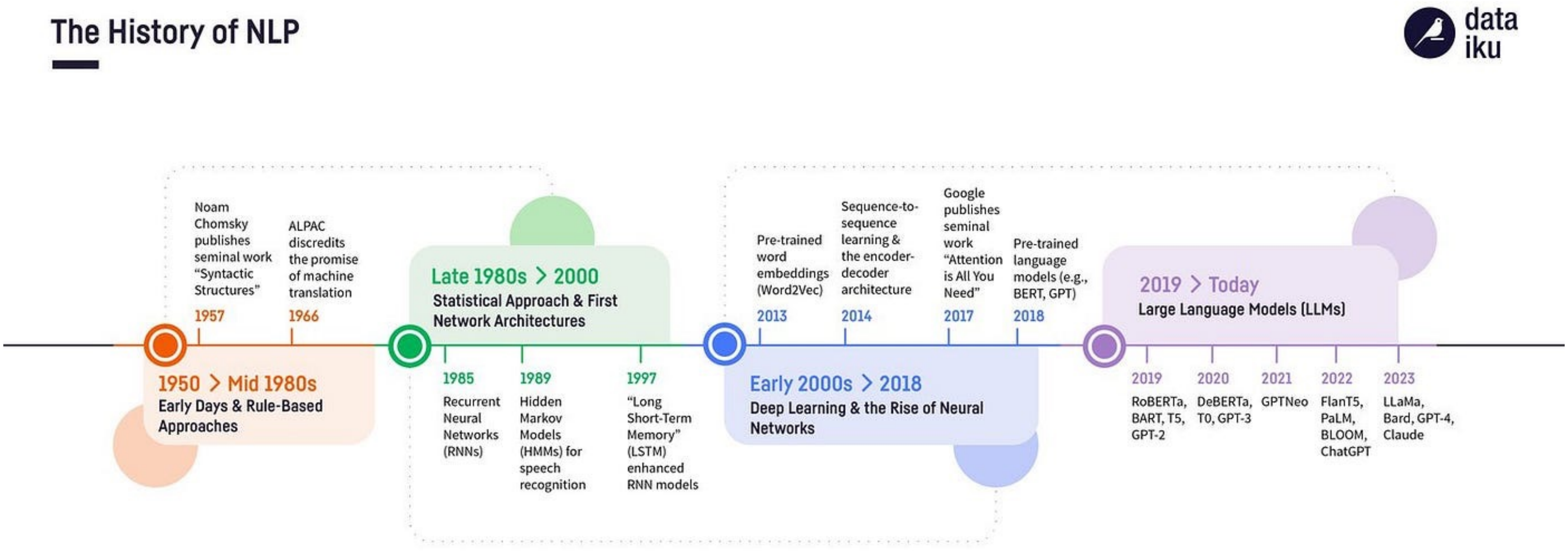
They are trained during training to capture **syntax, semantics and context**.



Male-Female   Verb Tense   Country-Capital

Image from https://www.askyourdata.co/blog/an-introduction-to-natural-language-processing-nlp

Natural Language Processing Timeline



The History of NLP

dataiku

1950 > Mid 1980s
Early Days & Rule-Based Approaches

1957
Noam Chomsky publishes seminal work "Syntactic Structures"

1966
ALPAC discredits the promise of machine translation

Late 1980s > 2000
Statistical Approach & First Network Architectures

1985
Recurrent Neural Networks (RNNs)

1989
Hidden Markov Models (HMMs) for speech recognition

1997
"Long Short-Term Memory" (LSTM) enhanced RNN models

Early 2000s > 2018
Deep Learning & the Rise of Neural Networks

2013
Pre-trained word embeddings (Word2Vec)

2014
Sequence-to-sequence learning & the encoder-decoder architecture

2017
Google publishes seminal work "Attention is All You Need"

2018
Pre-trained language models (e.g., BERT, GPT)

2019 > Today
Large Language Models (LLMs)

2019
RoBERTa, BART, T5, GPT-2

2020
DeBERTa, T0, GPT-3

2021
GPTNeo

2022
FlanT5, PaLM, BLOOM, ChatGPT

2023
LLaMa, Bard, GPT-4, Claude

https://medium.com/@jeevanchiru17/the-evolution-of-natural-language-processing-ad214cb9f6ca

Sequence-to-Sequence Models

Sequence-to-sequence (seq2seq) models aim at transforming an input sequence to an output sequence

- E.g., machine translation.

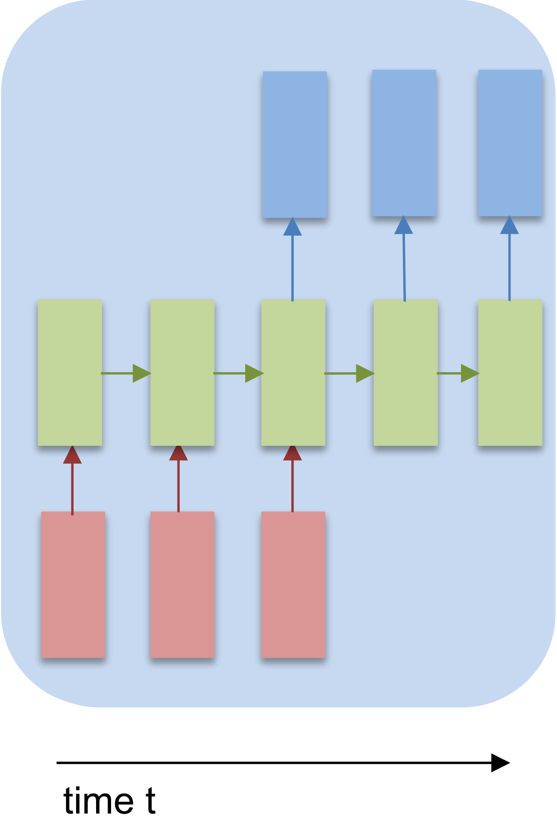
Input:

Horrible service the room was dirty

Output:

Un servizio orribile la camera era sporca

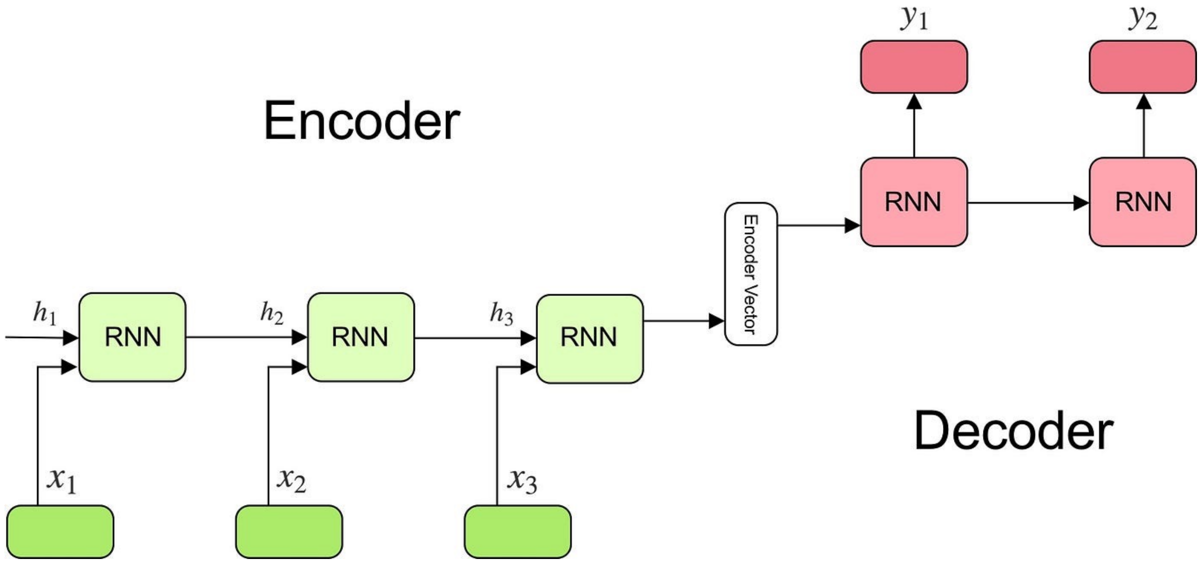
many to many



Sequence-to-Sequence Models

Seq2seq models generally have an **encoder-decoder** architecture, composed of:

- An encoder that processes the input sequence and compress the information into a **context vector** (also said **embedding**), which represents the entire sentence.
- A decoder that processes the context vector and produces the transformed output.

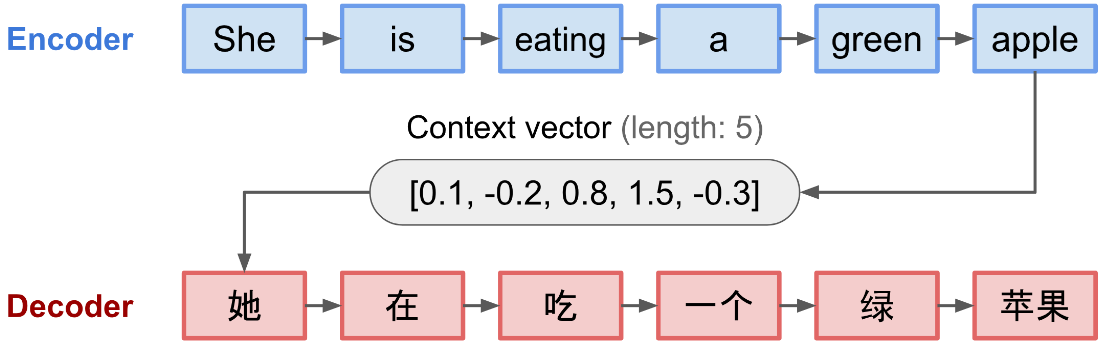


Sequence-to-Sequence Models

Disadvantages of the fixed length context vector design are:

- Incapability of remembering long sequences
- Too complex dynamics to be encoded in the hidden state

→ Attention mechanisms are proposed to solve this problem.
(First example in machine translation [1])



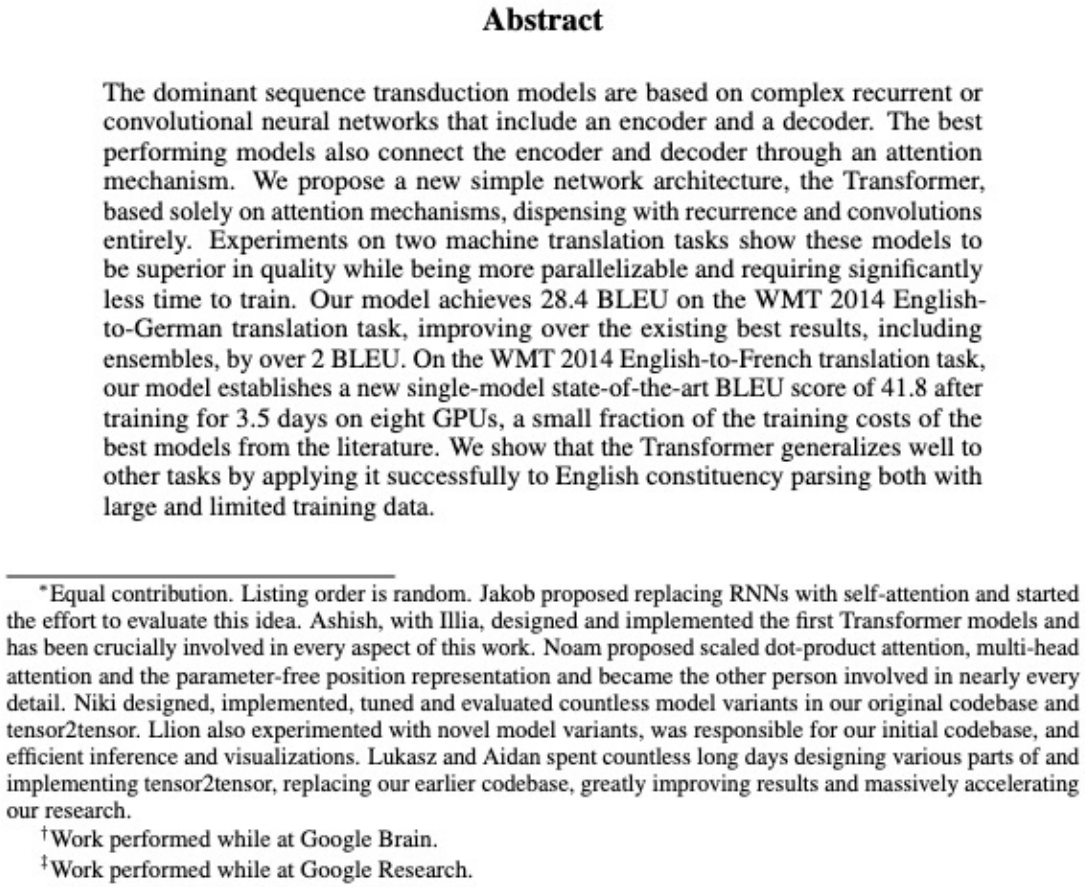
[1] “Neural machine translation by jointly learning to align and translate”, Bahdanau et al.
Image from: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Deep Learning for Time Series – Attention Models

The Attention Mechanism

Attention Mechanism: Intuition

Usually, when humans read, instead of reading every word carefully, scan for the most **relevant** parts.



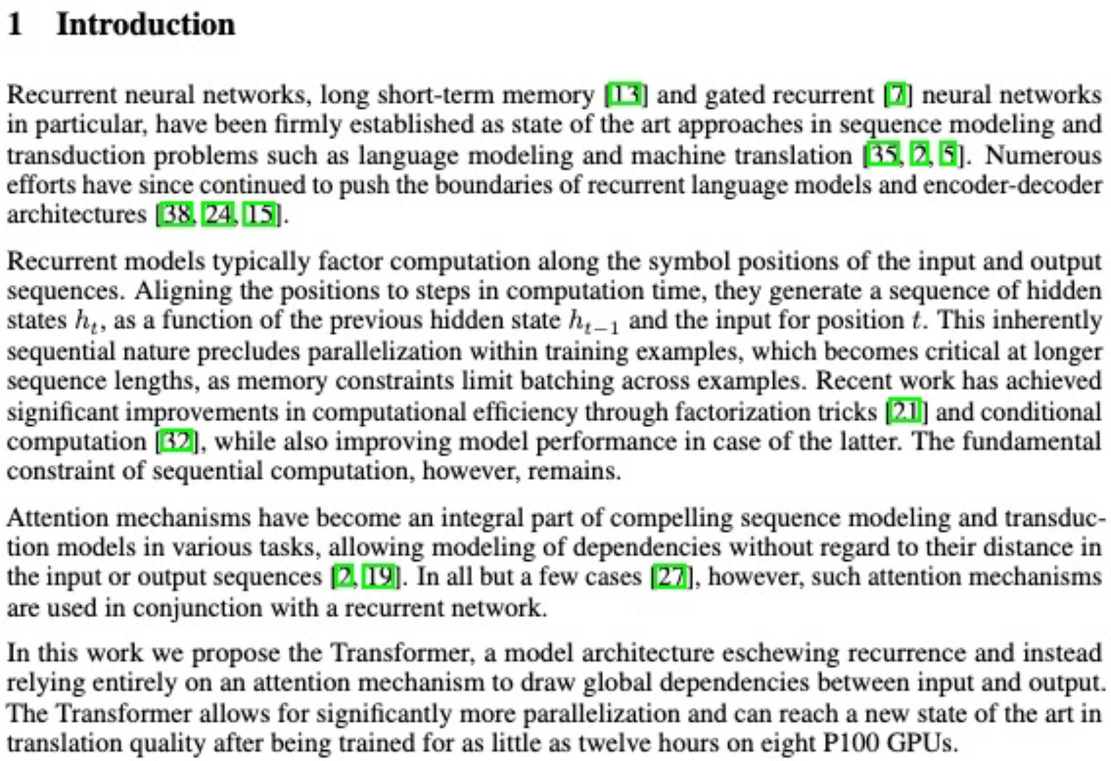
Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

[*] Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

[†] Work performed while at Google Brain.

[‡] Work performed while at Google Research.



1 Introduction

Recurrent neural networks, long short-term memory [13] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [35, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [38, 24, 15].

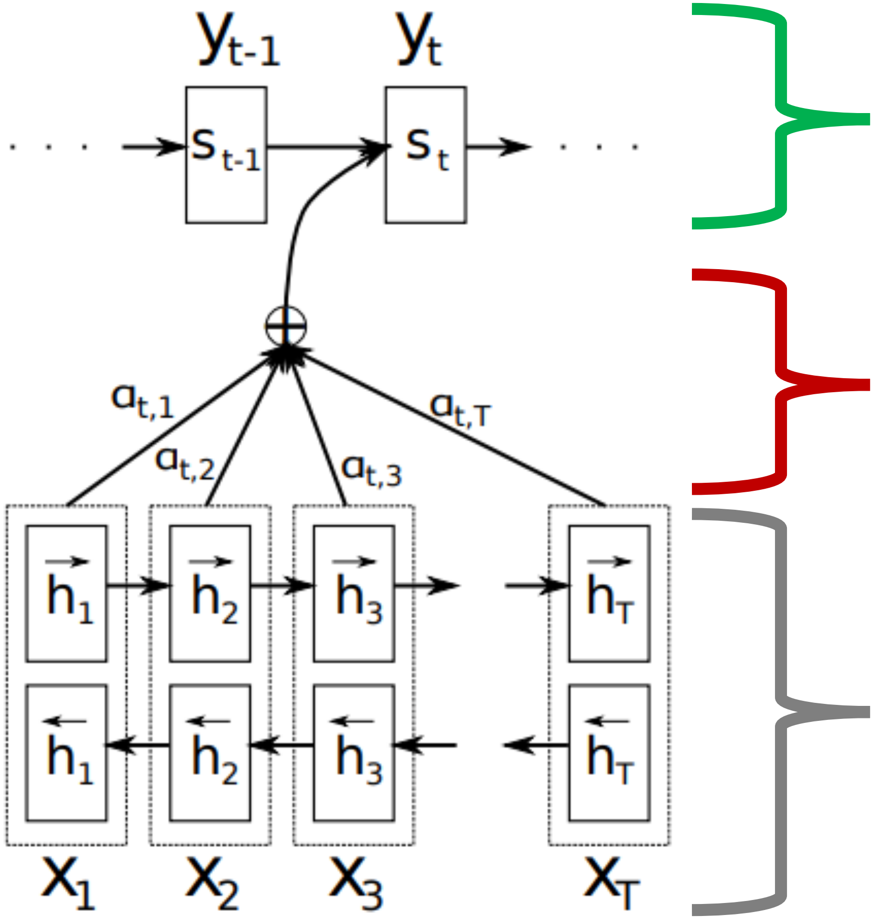
Recurrent models typically factor computation along the symbol positions of the input and output sequences. Aligning the positions to steps in computation time, they generate a sequence of hidden states h_t, as a function of the previous hidden state h_{t-1} and the input for position t. This inherently sequential nature precludes parallelization within training examples, which becomes critical at longer sequence lengths, as memory constraints limit batching across examples. Recent work has achieved significant improvements in computational efficiency through factorization tricks [21] and conditional computation [32], while also improving model performance in case of the latter. The fundamental constraint of sequential computation, however, remains.

Attention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences [2, 19]. In all but a few cases [27], however, such attention mechanisms are used in conjunction with a recurrent network.

In this work we propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little as twelve hours on eight P100 GPUs.

Extracts from Vaswani, A. "Attention is all you need." *Advances in Neural Information Processing Systems* (2017).

Origin of the Attention Mechanism



The **decoder** is an RNN, which receives a context vector as input

The **attention** (called “global alignment weights” by the authors) is a weighted sum parametrized by a feedforward neural network.

The **encoder** is a bi-directional RNN

[1] “Neural machine translation by jointly learning to align and translate”, Bahdanau et al.

Origin of the Attention Mechanism: Formalization

Let x be the input sequence of length n, and y the output sequence of length m.

Let $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$ be the encoder state, given by the concatenation of the forward and backward hidden states of the bidirectional RNN.

Le denote with s_t the decoder state for the output at position t. The context vector is defined as the **sum of the encoder states**, **weighted by the alignment scores**:

$$c_t = \sum_{i=1}^{n} a_{ti} h_i$$

where $a_{ti} = softmax(score(s_{t-1}, h_i))$, and $score(s_{t-1}, h_i) = \mathrm{v_a} \tanh(\mathrm{W_a}[s_{t-1}; h_i])$.

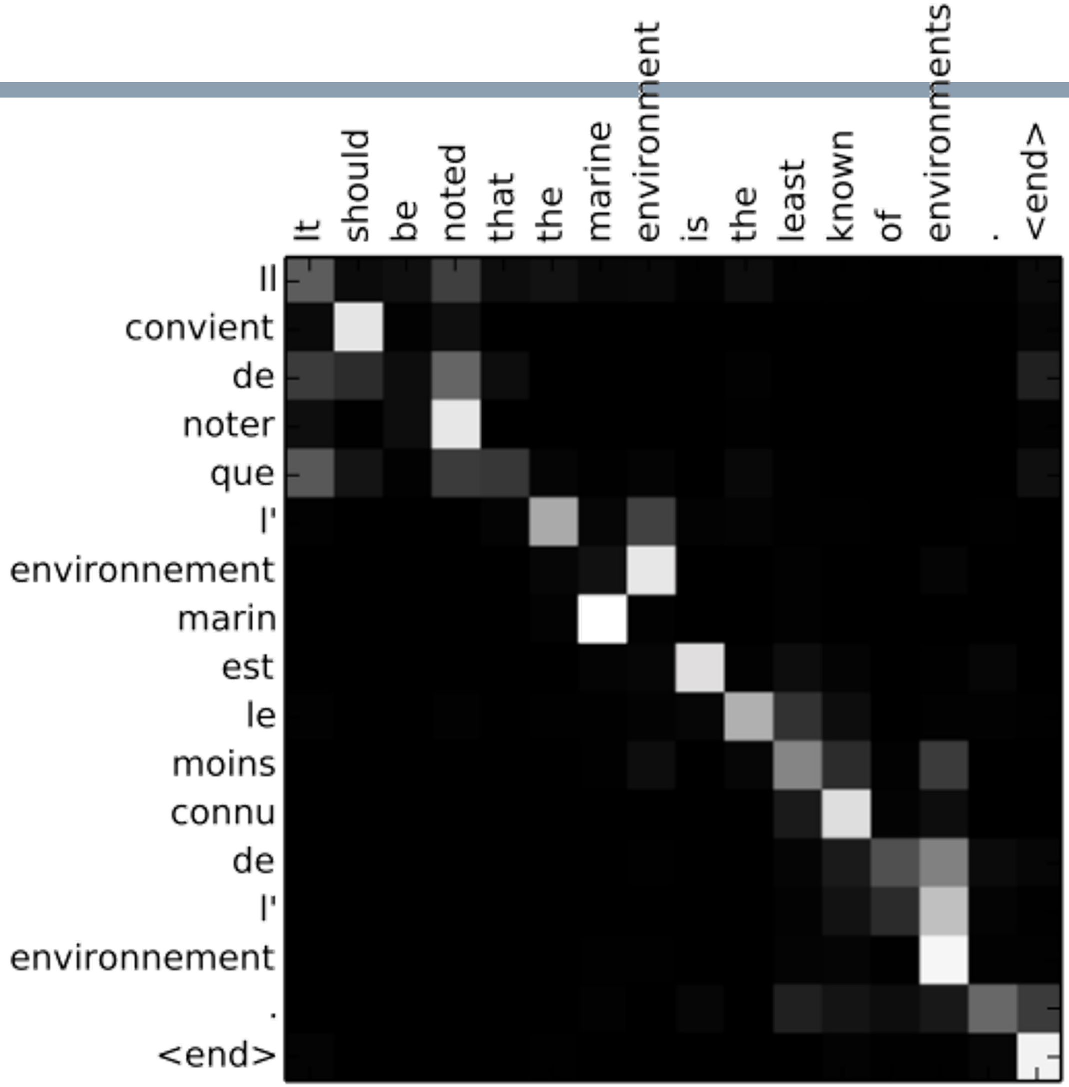
[1] “Neural machine translation by jointly learning to align and translate”, Bahdanau et al.

Attention Mechanism: Example

Attention produces a different context vector for each output token, enabling the model to **selectively** consider input words.

In the example on the side, the x-axis represents the source sentence and the y-axis the generated sentence.

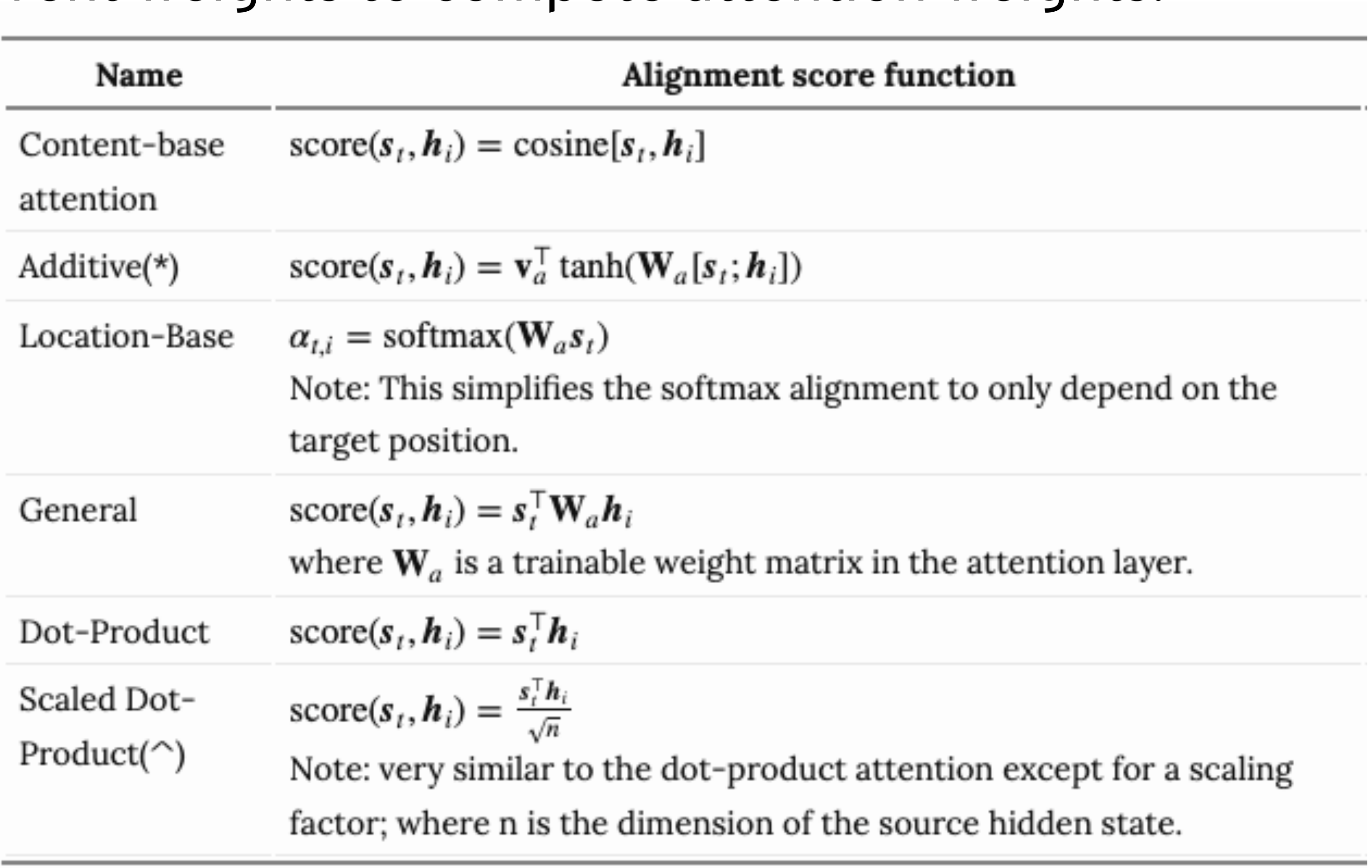
Every pixel (i, j) indicates the weight the j-th source word embedding when generating the i-th word.



[1] “Neural machine translation by jointly learning to align and translate”, Bahdanau et al.

Attention Mechanism: Variations

There are different weights to compute attention weights.



Name	Alignment score function
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.

Table from: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.htm

Attention Mechanism: Variations

Attention mechanisms can be characterised according to their function or design as:

- **Soft or hard attention**
- **Global or local attention**
- **Self-attention**
- **Cross-attention**

Soft / Hard Attention

- **Soft attention** computes a weighted sum of all input tokens based on their relevance to the current output token. *Example application: weigh how each time step influences the current time step.*

- **Hard attention** select subsets of the input in a binary manner. *Example application: focus on specific intervals of a time series.*

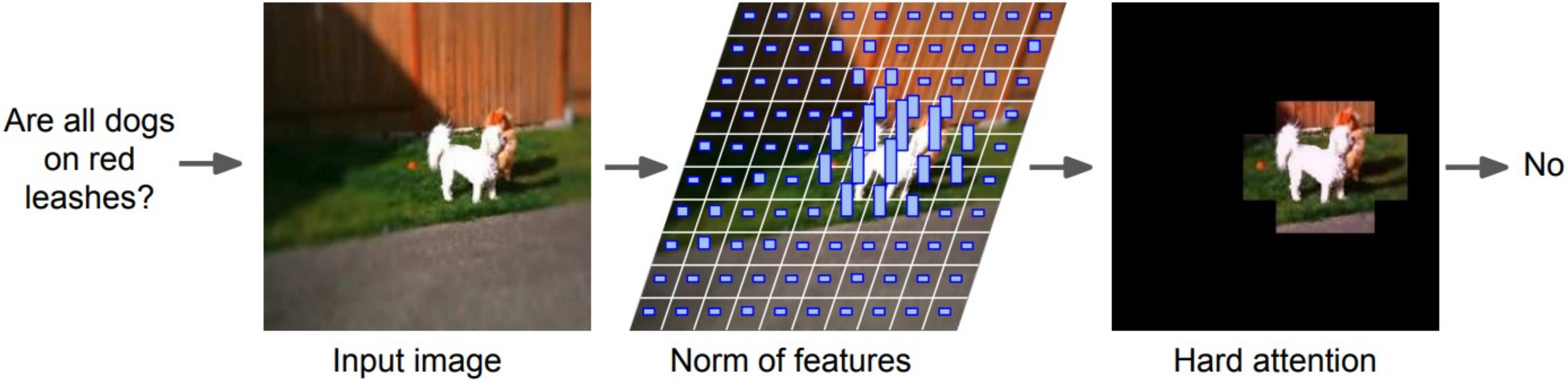


Image from: https://lyusungwon.github.io/studies/2018/11/23/han/

Global / Local attention

- **Global attention**: the context vectors depend on the whole input sequence.
  Example application: climate modelling to integrate long term historical data.

- **Local attention**: The context vectors depends on a subset of the input sequence.
  Example application: short term traffic prediction.

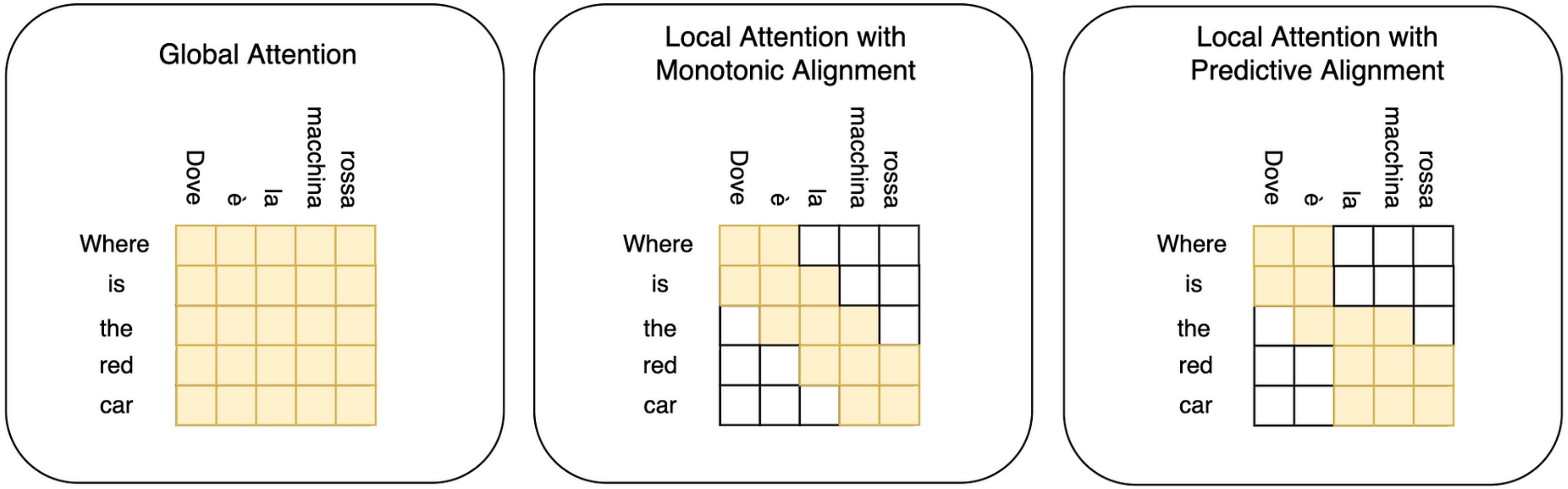

Global Attention
Local Attention with Monotonic Alignment
Local Attention with Predictive Alignment
Dove
è
la
macchina
rossa
Where
is
the
red
car

Figure from: https://medium.com/nlplanet/two-minutes-nlp-visualizing-global-vs-local-attention-c61b42758019

Self-Attention

Self-attention is an attention mechanism which relates different positions of an input sequence to generate a representation of the same sequence.

It allows the model to capture relationships within the sequence.

Example application: Predicting the next word in a sentence.

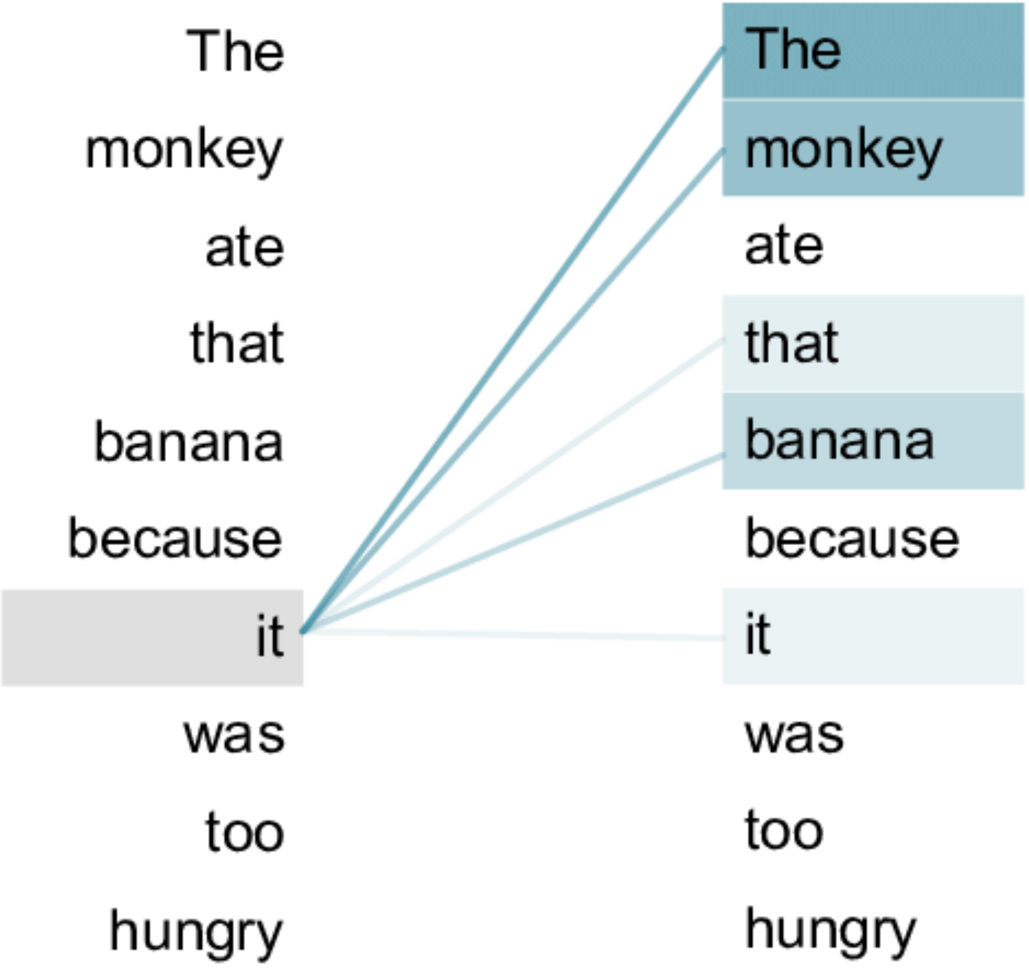
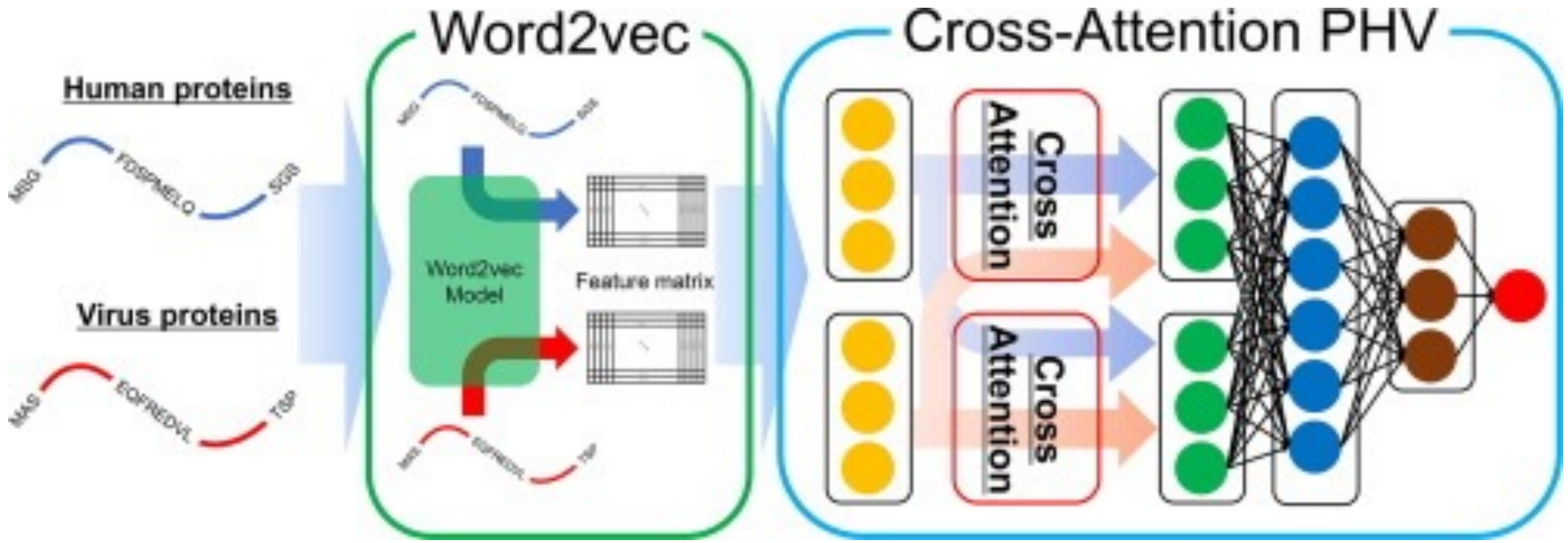


Figure from: Xie, Huiqiang & Qin, Zhijin & Li, Geoffrey & Juang, Biing-Hwang. (2020). Deep Learning Enabled Semantic Communication Systems. 10.48550/arXiv.2006.10685.

Cross-Attention

Cross-attention is used in tasks where two input sequences are present, so that the context vectors depend on both sequences.
Example application: Predicting interactions between two sequences.



Tsukiyama, S., & Kurata, H. (2022). Cross-attention PHV: Prediction of human and virus protein-protein interactions using cross-attention–based neural networks. *Computational and Structural Biotechnology Journal*, *20*, 5564-5573.

Attention-based Models in Time Series

The attention mechanism can be **integrated** in various architectures (RNNs, …)

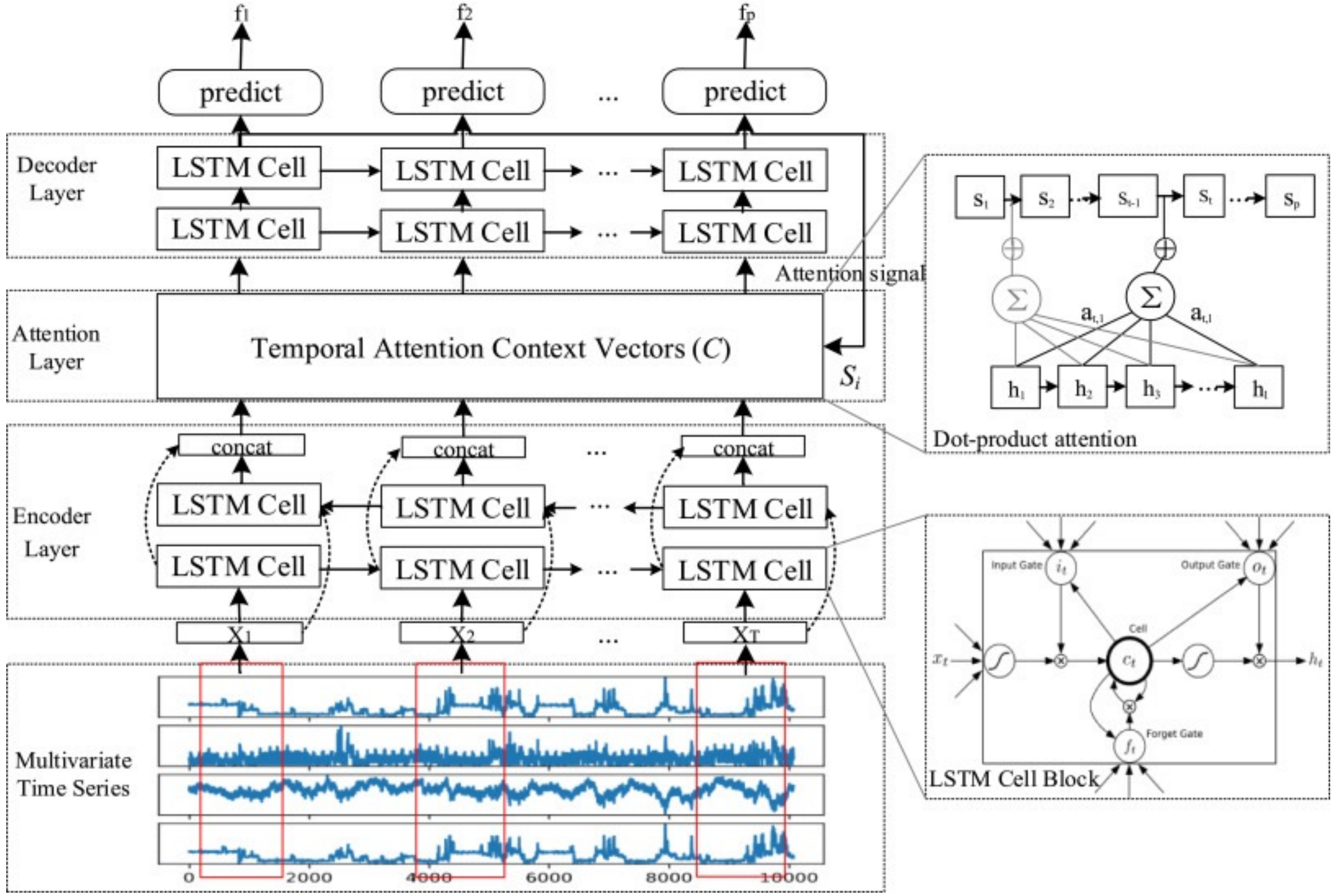


Image from Du, Shengdong, et al. "Multivariate time series forecasting via attention-based encoder–decoder framework." *Neurocomputing* 388 (2020): 269-279.

Deep Learning for Time Series – Attention Models

The Transformer Architecture

Attention Is All You Need [1]

The **Transformer** [1] architecture was published in 2017. Models built on this architecture have become state-of-the-art in many domains, starting with **Natural Language Processing**.

ChatGPT

- Completely built on the **self-attention** mechanism
- Does **not** use sequence any recurrent architecture:
  - More efficient: input sequences can be processed in **parallel**
  - Has no inherent understanding of sequence order
- Can be applied to various tasks, including in time series machine learning

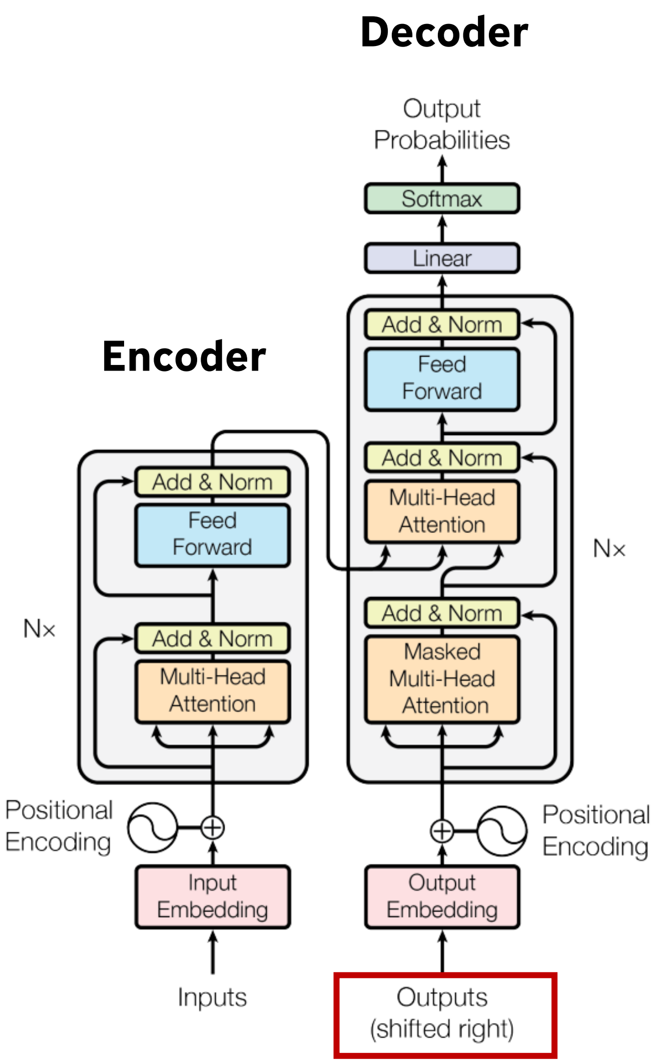
[1] “Attention is all you need”, Vaswani, et al.

The Transformer Architecture

The fundamentals components of the Transformer architecture are:

- **Positional encoding**
- **Multi-head self attention** based scaled dot product attention
- An **encoder-decoder** architecture

The transformer was first proposed as a **machine translation** model.



“Attention is all you need”, Vaswani, et al.

Transformer: Positional Encoding

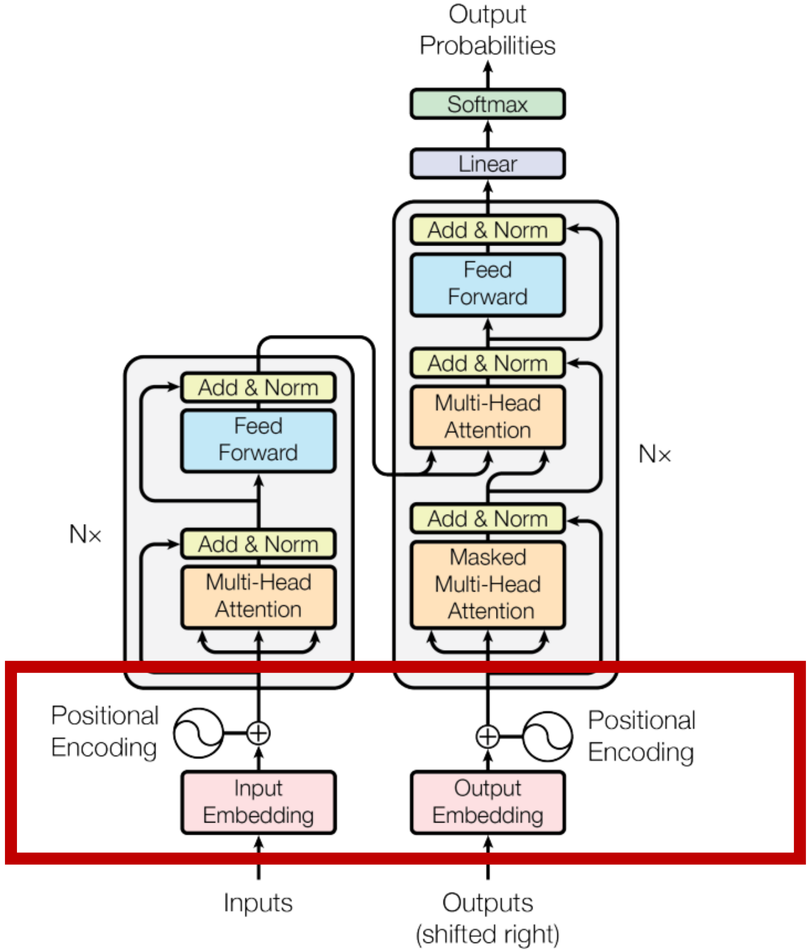
Positional encoding is added to the input and output sequence embeddings in the encoder and decoder. It is necessary to give information on **word order**.

Positional encoding is defined by using sine and cosine functions with different frequencies:

$$PE_{(pos,\,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

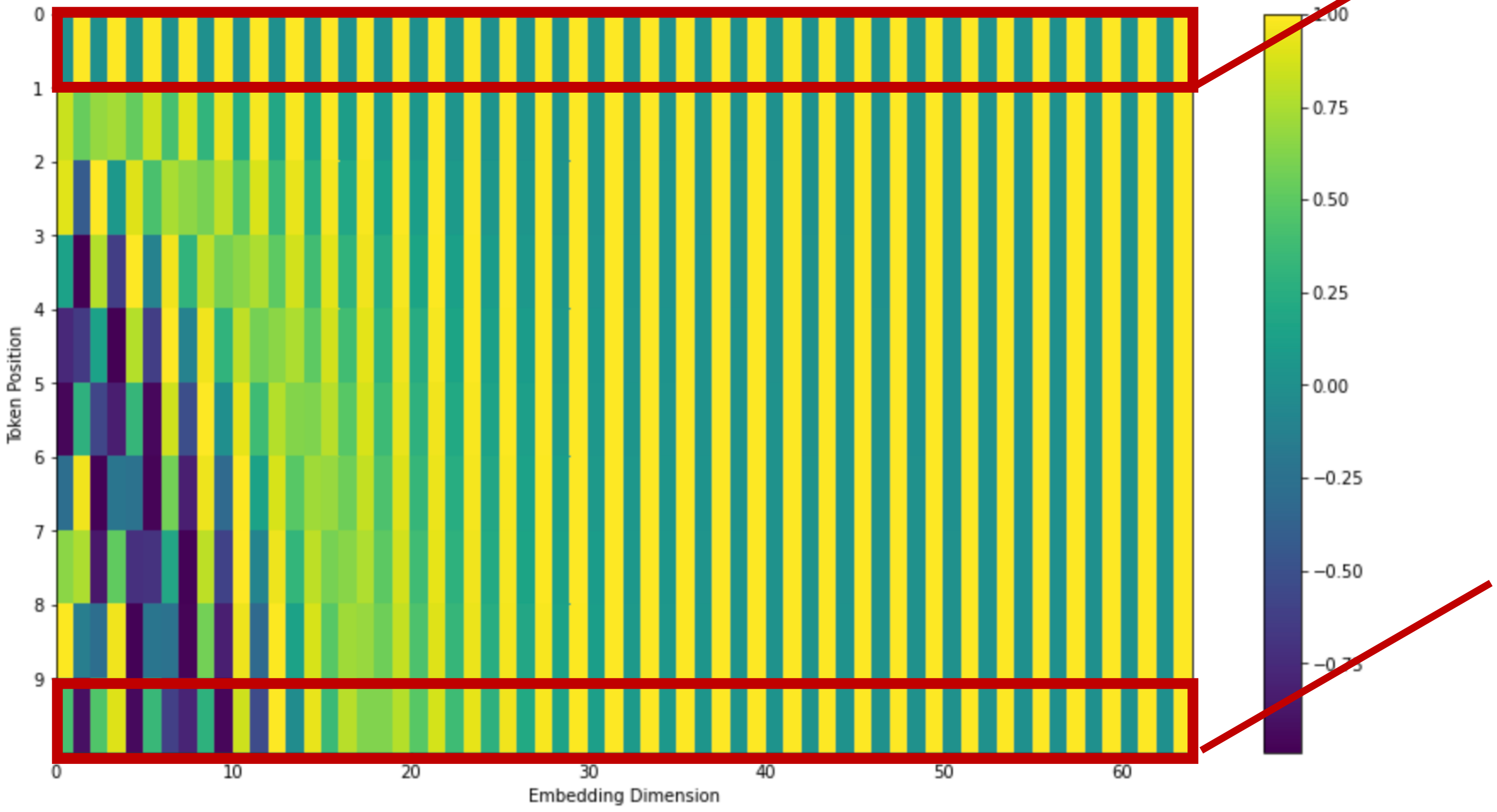
$$PE_{(pos,\,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

where pos is the position and i is the embedding dimension.



"Attention is all you need", Vaswani, et al.

Transformer: Positional Encoding

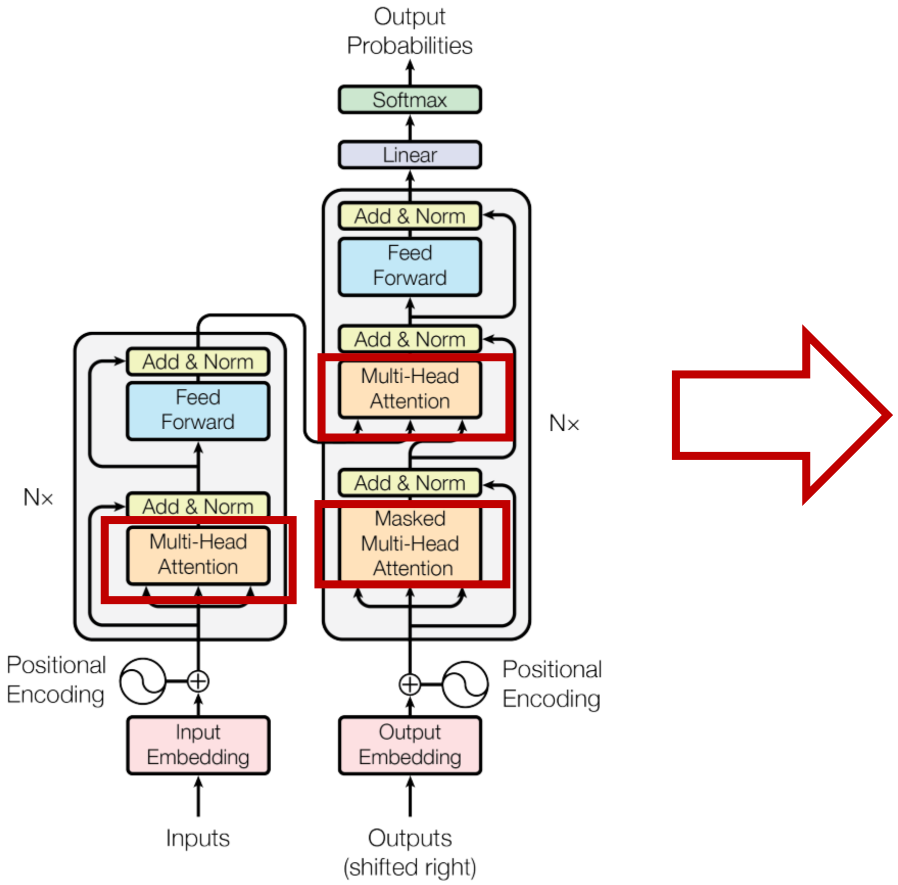


Positional encoding of the 1st input

Code to generate this example available at: https://github.com/jalammar/jalammar.github.io/blob/master/notebookes/transformer/transformer_positional_encoding_graph.ipynb

Positional encoding of the 10th input

Transformer: Multi-head Attention Mechanisms



Three multi-head attention mechanisms are used in the Transformer architecture:

- **Encoder self-attention**: captures relationships between input sequence tokens
- **Decoder self-attention:** captures relationships between output sequence tokens
- **Encoder-decoder cross-attention**: captures relationships between input and output sequences

“Attention is all you need”, Vaswani, et al.

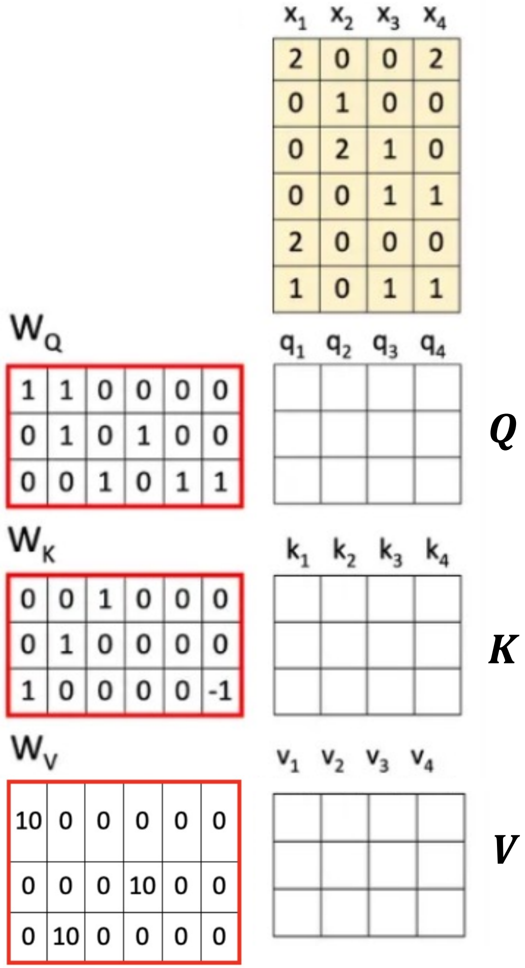
Transformer: Scaled Dot Product Attention

Scaled dot product attention takes three sequences of vectors as input: the query $\boldsymbol{Q}$, the key $\boldsymbol{K}$ and value $\boldsymbol{V}$.

- Key: what token to focus on and compare to.
- Query: how is each token relevant with respect to the key
- Value: information needed, once relevance is calculated

In the case of self-attention, those sequences are all transformed representations of the **same sequence**.

These vector are calculated by linear transformation using **weight matrices**, resp., W^Q, W^K, and W^V.



$$\begin{array}{cccc} x_1 & x_2 & x_3 & x_4 \\ 2 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{array}$$

$$W_Q = \begin{array}{|cccccc|} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{array} \quad \begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \\ \ & \ & \ & \ \\ \ & \ & \ & \ \\ \ & \ & \ & \ \end{array} \quad \boldsymbol{Q}$$

$$W_K = \begin{array}{|cccccc|} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \end{array} \quad \begin{array}{cccc} k_1 & k_2 & k_3 & k_4 \\ \ & \ & \ & \ \\ \ & \ & \ & \ \\ \ & \ & \ & \ \end{array} \quad \boldsymbol{K}$$

$$W_V = \begin{array}{|cccccc|} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \end{array} \quad \begin{array}{cccc} v_1 & v_2 & v_3 & v_4 \\ \ & \ & \ & \ \\ \ & \ & \ & \ \\ \ & \ & \ & \ \end{array} \quad \boldsymbol{V}$$

Image from https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

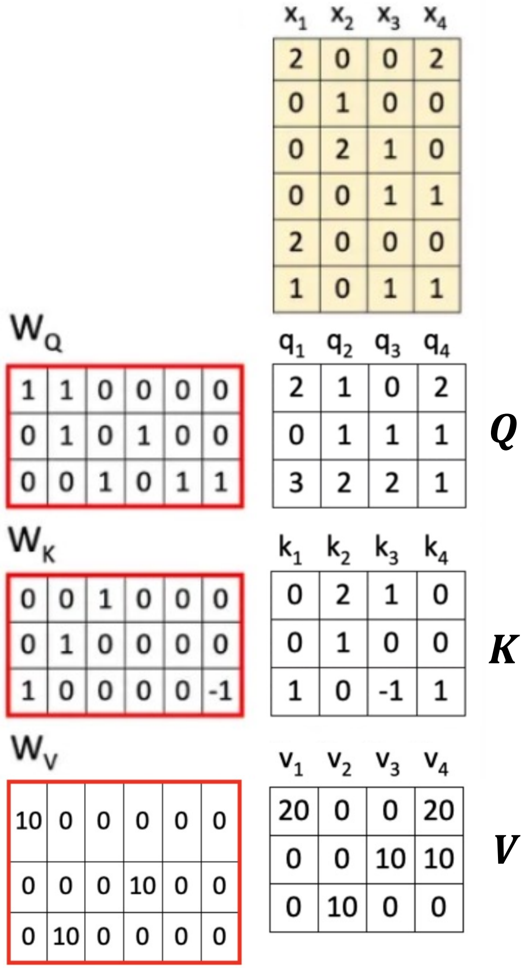
Transformer: Scaled Dot Product Attention

Scaled dot product attention takes three sequences of vectors as input: the query $\boldsymbol{Q}$, the key $\boldsymbol{K}$ and value $\boldsymbol{V}$.

- Key: what token to focus on and compare to.
- Query: how is each token relevant with respect to the key
- Value: information needed, once relevance is calculated

In the case of self-attention, those sequences are all transformed representations of the **same sequence**.

These vector are calculated by linear transformation using **weight matrices**, resp., W^Q, W^K, and W^V.



x_1	x_2	x_3	x_4
2	0	0	2
0	1	0	0
0	2	1	0
0	0	1	1
2	0	0	0
1	0	1	1

W_Q

1	1	0	0	0	0
0	1	0	1	0	0
0	0	1	0	1	1

q_1	q_2	q_3	q_4
2	1	0	2
0	1	1	1
3	2	2	1

$\boldsymbol{Q}$

W_K

0	0	1	0	0	0
0	1	0	0	0	0
1	0	0	0	0	-1

k_1	k_2	k_3	k_4
0	2	1	0
0	1	0	0
1	0	-1	1

$\boldsymbol{K}$

W_V

10	0	0	0	0	0
0	0	0	10	0	0
0	10	0	0	0	0

v_1	v_2	v_3	v_4
20	0	0	20
0	0	10	10
0	10	0	0

$\boldsymbol{V}$

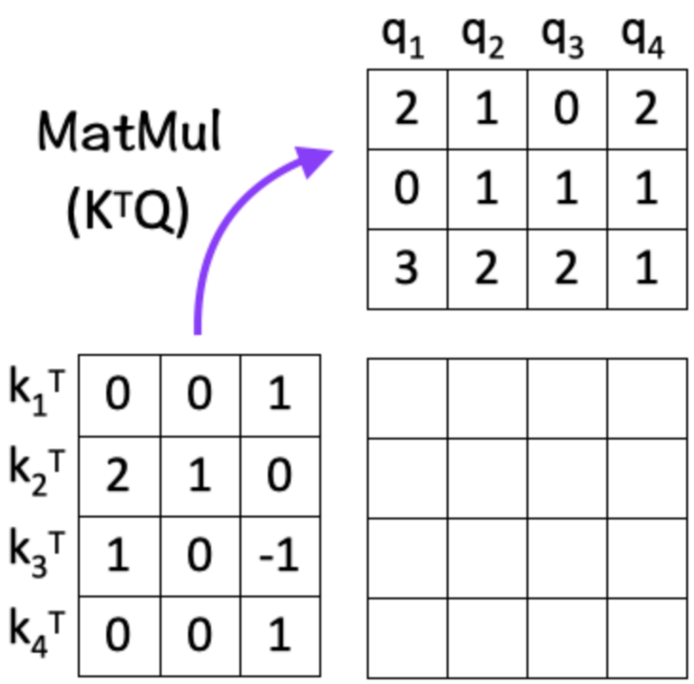
https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

Transformer: Scaled Dot Product Attention

Then, the scaled dot-product attention is computed by:

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

where d_k is the dimension of $\boldsymbol{Q}$ and $\boldsymbol{K}$.



MatMul

(K^TQ)

$$\begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \\ \hline 2 & 1 & 0 & 2 \\ 0 & 1 & 1 & 1 \\ 3 & 2 & 2 & 1 \end{array}$$

$$\begin{array}{c|ccc} k_1^T & 0 & 0 & 1 \\ k_2^T & 2 & 1 & 0 \\ k_3^T & 1 & 0 & -1 \\ k_4^T & 0 & 0 & 1 \end{array}$$

“Attention is all you need”, Vaswani, et al.

https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

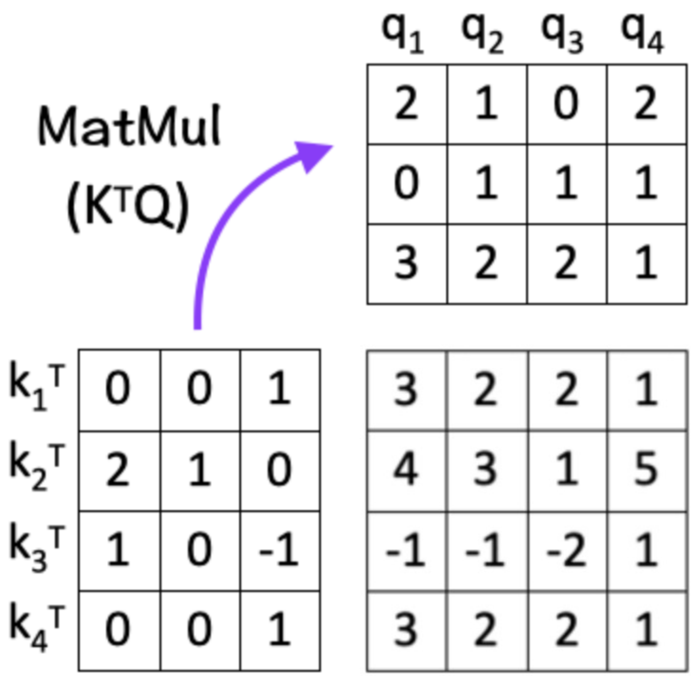
Transformer: Scaled Dot Product Attention

Then, the scaled dot-product attention is computed by:

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

where d_k is the dimension of $\boldsymbol{Q}$ and $\boldsymbol{K}$.

Reminder: $$\boxed{s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}}$$



MatMul

(K^TQ)

$$\begin{array}{c|c|c|c}
q_1 & q_2 & q_3 & q_4 \\
\hline
2 & 1 & 0 & 2 \\
0 & 1 & 1 & 1 \\
3 & 2 & 2 & 1
\end{array}$$

$$\begin{array}{c|c|c|c}
k_1^T & 0 & 0 & 1 \\
k_2^T & 2 & 1 & 0 \\
k_3^T & 1 & 0 & -1 \\
k_4^T & 0 & 0 & 1
\end{array}
\qquad
\begin{array}{c|c|c|c}
3 & 2 & 2 & 1 \\
4 & 3 & 1 & 5 \\
-1 & -1 & -2 & 1 \\
3 & 2 & 2 & 1
\end{array}$$

“Attention is all you need”, Vaswani, et al.

https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

Transformer: Scaled Dot Product Attention

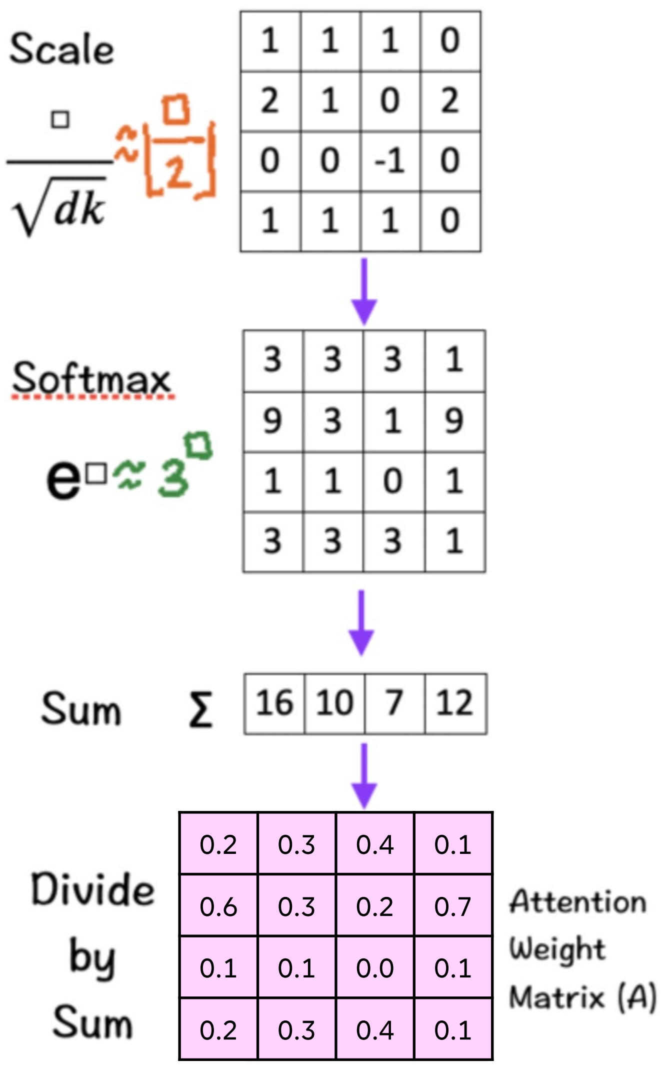
Then, the scaled dot-product attention is computed by:

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

where d_k is the dimension of $\boldsymbol{Q}$ *and* $\boldsymbol{K}$.

Reminder:

$$\boxed{s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}}$$



Scale $\frac{\square}{\sqrt{dk}} \approx \left[\frac{\square}{2}\right]$

1	1	1	0
2	1	0	2
0	0	-1	0
1	1	1	0

Softmax $e^{\square} \approx 3^{\square}$

3	3	3	1
9	3	1	9
1	1	0	1
3	3	3	1

Sum Σ

16	10	7	12

Divide by Sum

0.2	0.3	0.4	0.1
0.6	0.3	0.2	0.7
0.1	0.1	0.0	0.1
0.2	0.3	0.4	0.1

Attention Weight Matrix (A)

"Attention is all you need", Vaswani, et al.

https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

Transformer: Scaled Dot Product Attention

Then, the scaled dot-product attention is computed by:

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

where d_k is the dimension of $\boldsymbol{Q}$ and $\boldsymbol{K}$.

Reminder:

$$\boxed{s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}}$$

0.2	0.3	0.4	0.1
0.6	0.3	0.2	0.7
0.1	0.1	0.0	0.1
0.2	0.3	0.4	0.1

Attention Weight Matrix (A)

V_1	V_2	V_3	V_4
20	0	0	20
0	0	10	10
0	10	0	0

Z_1	Z_2	Z_3	Z_4
8	12	16	4
3	4	4	2
6	3	2	7

Attention Weighted Features

“Attention is all you need”, Vaswani, et al.

https://towardsdatascience.com/deep-dive-into-self-attention-by-hand-%EF%B8%8E-f02876e49857

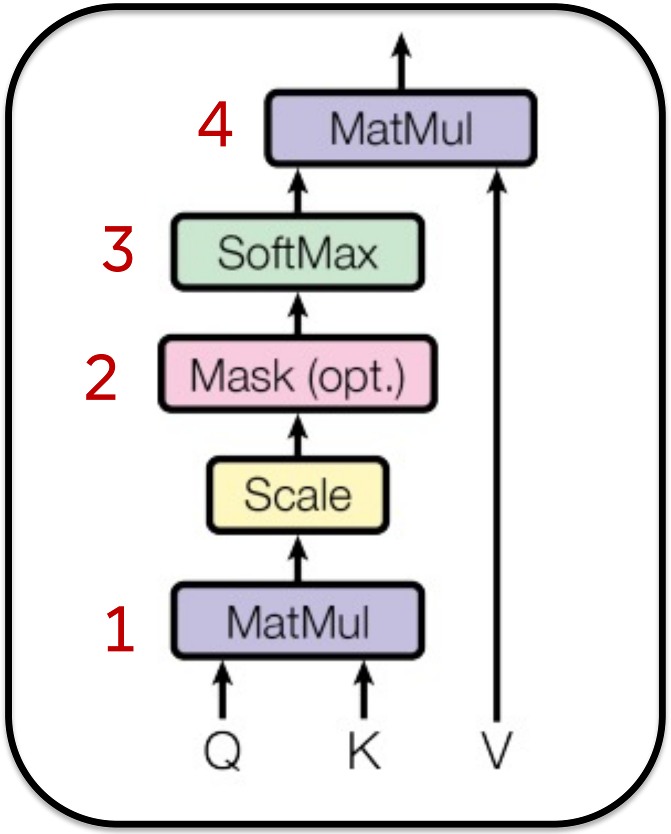
Transformer: Scaled Dot Product Attention

Then, the scaled dot-product attention is computed by:

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

where d_k is the dimension of $\boldsymbol{Q}$ *and* $\boldsymbol{K}$.

1. How each token is related to the query (i.e., similarity score)
2. Converts into probabilities (sum to 1)
3. **Causal masking** is used to prevent information leakage
4. Weighted sum of values that highlights relevant vectors



Scaled Dot-Product Attention

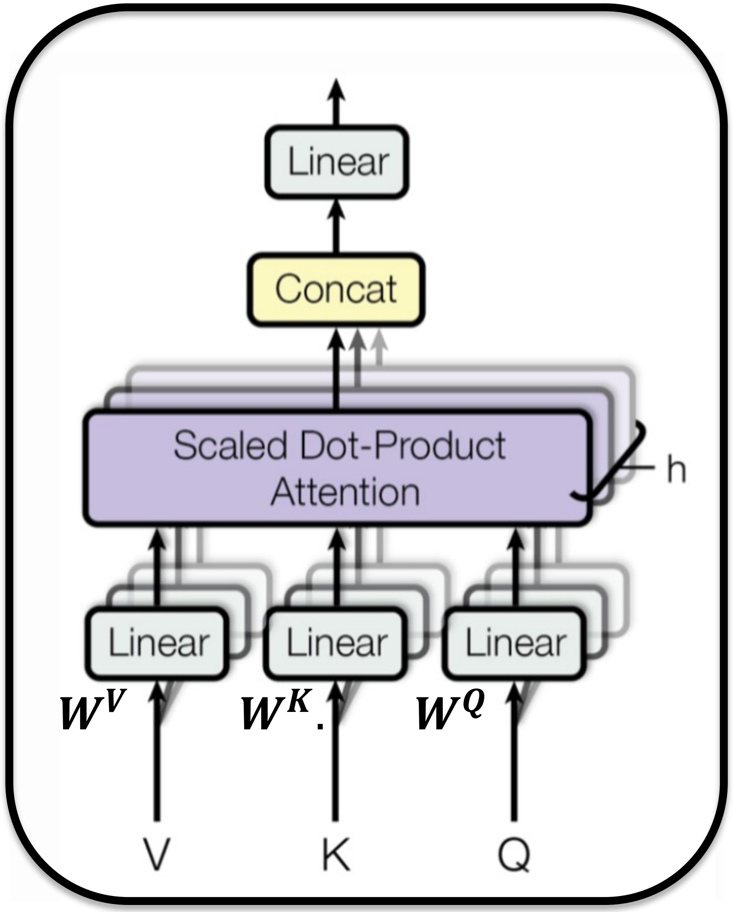
“Attention is all you need”, Vaswani, et al.

Transformer: Multi-Head Self-Attention

Multi-head attention is used to compute attention several times in **parallel**, using **independent** weight matrices $W^{V,i}$ $W^{K,i}$ and $W^{Q,i}$. These operations are called **attention heads**, and are then **concatenated**.

Basic scaled dot product attention is not sufficient to encode the **complexity** of language, as it might only focus on one aspect of relationships between tokens.

For instance, the multi-head attention mechanism can focus both on **long range and sort range relationships** through different attention heads.



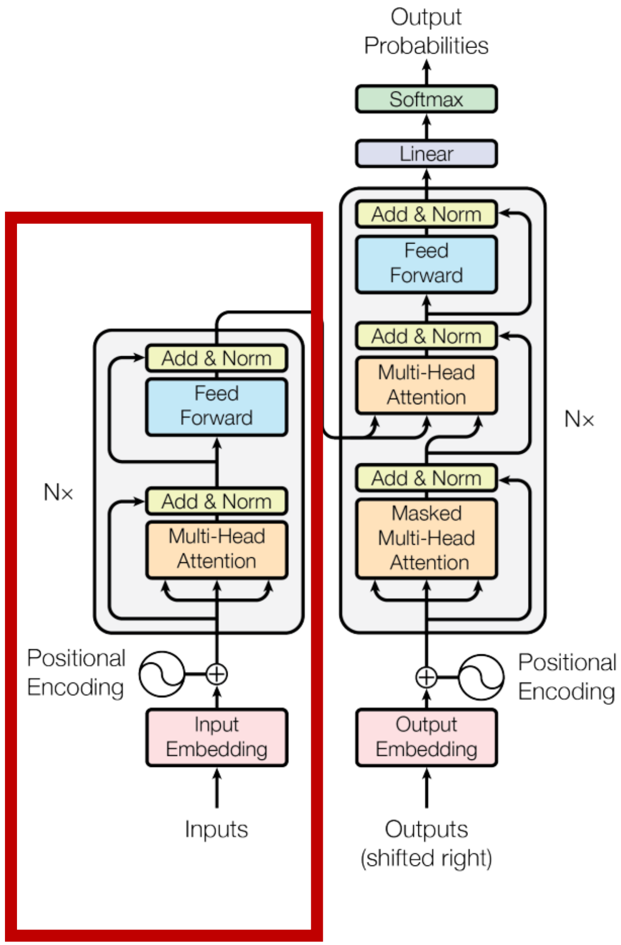
Multi-head attention

“Attention is all you need”, Vaswani, et al.

Transformer: Encoder

The encoder generates a **contextualized representation** of the **input sequence**. This representation has the same length as the input sequence, and is used to condition the decoder.

- It is made of N identical layers, each composed of
  - Multi-head **self-attention**
  - Feedforward neural network
  - Residual connection
  - Layer normalization

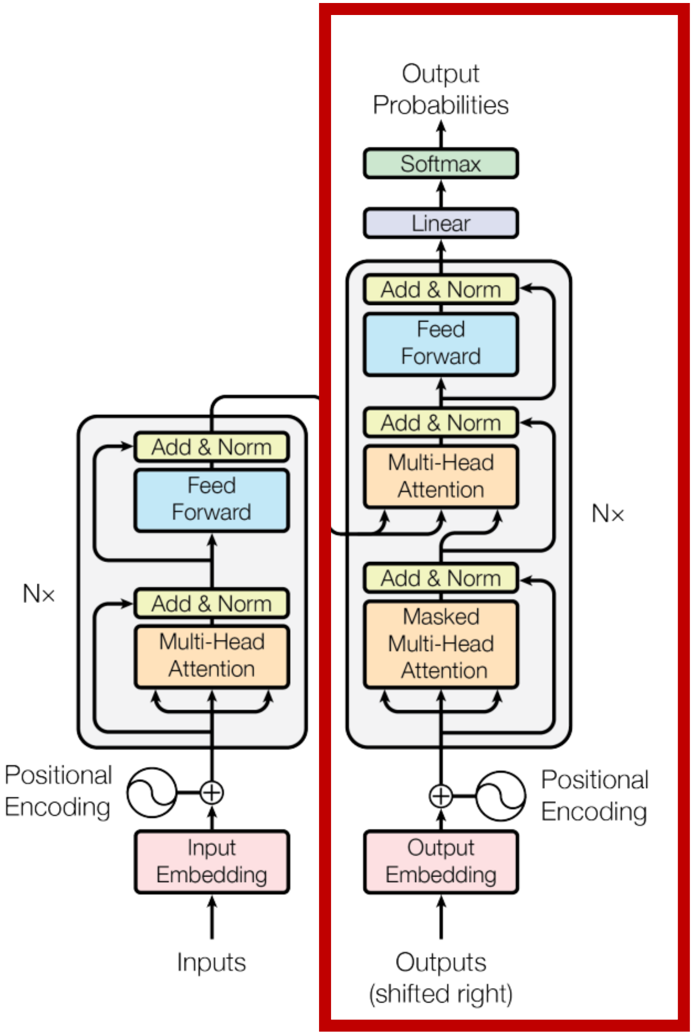


"Attention is all you need", Vaswani, et al.

Transformer: Decoder

The **decoder** generates the output of the model (e.g., translated text). It generates tokens one at a time, using network information from the **encoded representation**.

- It is also made of $\boldsymbol{N}$ identical layers:
  - **Masked** multi-head self-attention
  - Multi-head **cross-attention**
  - Feedforward neural network
  - Residual connection
  - Layer normalization

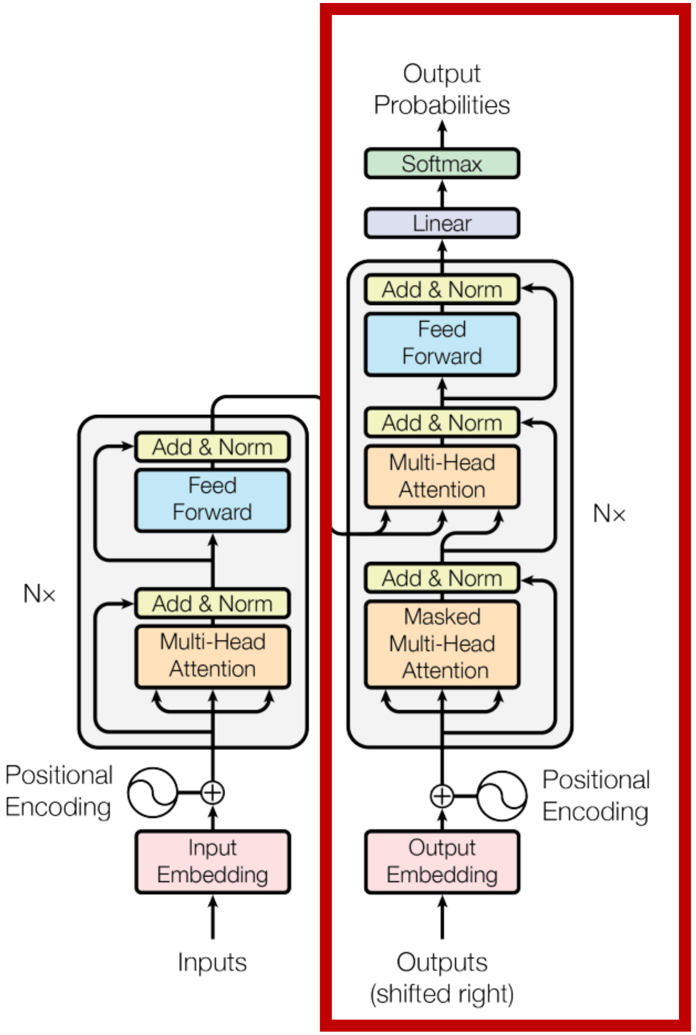


[2] “Attention is all you need”, Vaswani, et al.

Transformer: Decoder

The decoding works as follows:

1. The output of the **final encoder layer** is transformed into a set of vectors $\boldsymbol{K}$ and $\boldsymbol{V}$.
2. These are used by each decoder layer as Key and Value of a **cross-attention mechanism** where the Query is represents the already generated output tokens.
3. The output tokens are generated until a **special symbol** of <end> is generated.



[2] "Attention is all you need", Vaswani, et al.

Transformer Model: Pros and Cons

The main advantages of Transformer models are:

- **Parallelization:** Transformers can process all tokens in a sequence simultaneously.
- **Long-Range Dependencies:** Transformers can capturing dependencies between distant tokens, as long as they are all present in the same sequence

However:

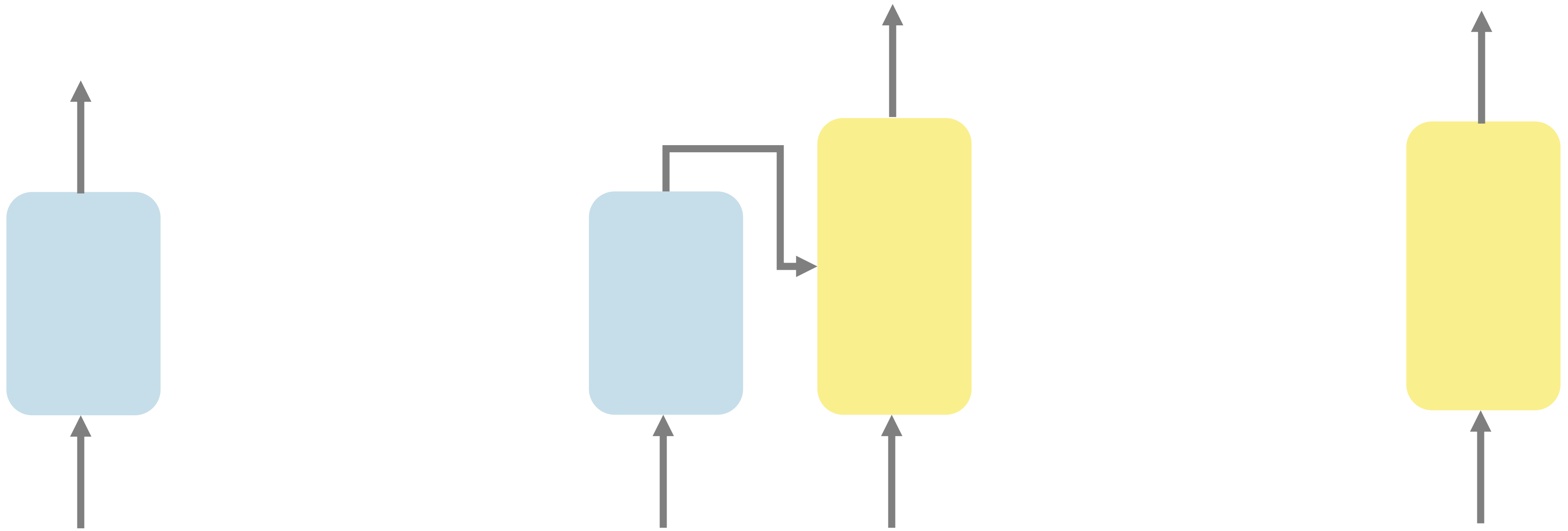
- **Quadratic time complexity** of attention mechanism

Deep Learning for Time Series – Attention-based Models

Transformer Applications

Transformer Architectures

Since the original Transformer model, **three** main types of transformer architectures have been used for different applications.



Encoder only

Encoder-decoder

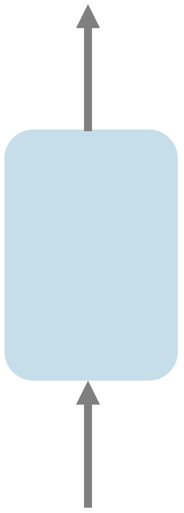
Decoder only

Encoder-Only Models

Encoder-only models are used for tasks based on **encoding** (i.e. understanding) the input sequence. They produce a **fixed-length contextual representation** for each input token (or for the entire sequence).

Some examples are:

- Sentiment analysis and other **text classification** tasks
- Classifying **similarity** between two sequences
- Time series **anomaly detection**



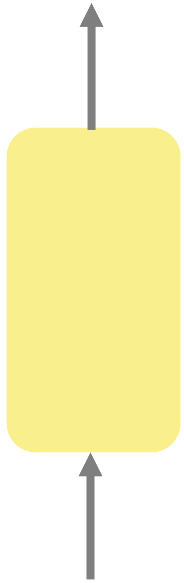
Encoder only

Decoder-Only Models

Decoder-only models progressively **generate** a sequence of output tokens until an end condition is met (for instance a specific end token) using **masked self-attention**.

Some examples are:

- Story writing and other **text generation** tasks
- Time series **forecasting**

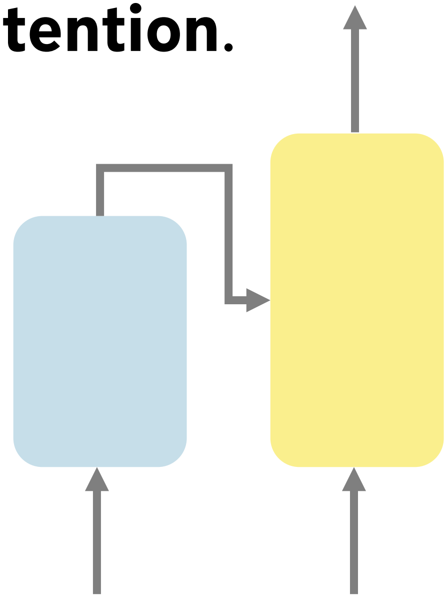


Decoder only

Encoder-Decoder Models

Similar to the original Transformer architecture, these models both **encode** the input and **generate** sequences of output tokens. They progressively generate output tokens until end condition is met.

These models usually use **cross-attention** and **masked-self attention**.



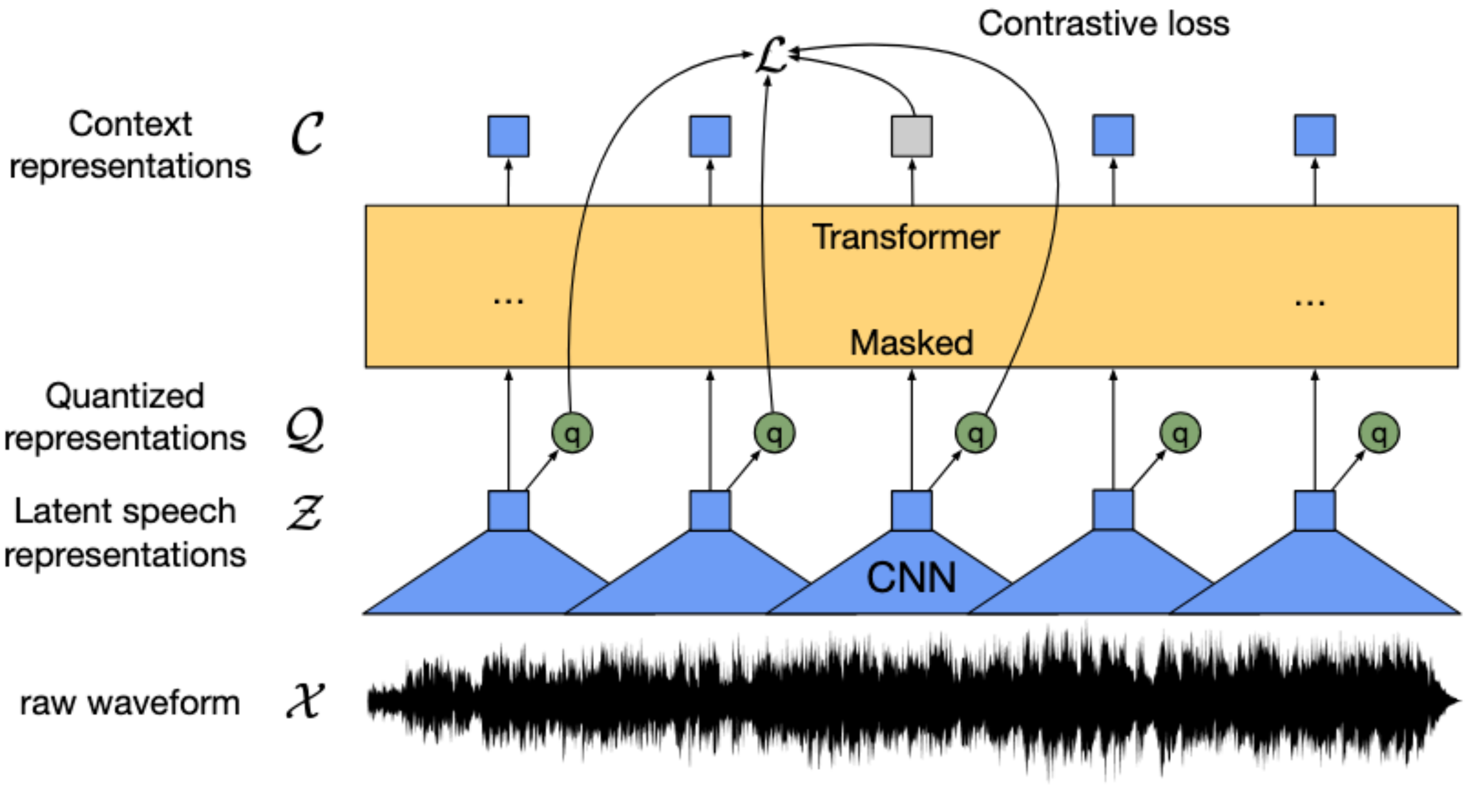
Encoder-decoder

Some examples are:

- **Machine translation**
- **Multivariate** time series **forecasting**

Transformers and Time Series: the Example of Speech

Although the Transformer has first been applied to **text**, it can also be applied to **time series**, for instance to **speech** data.

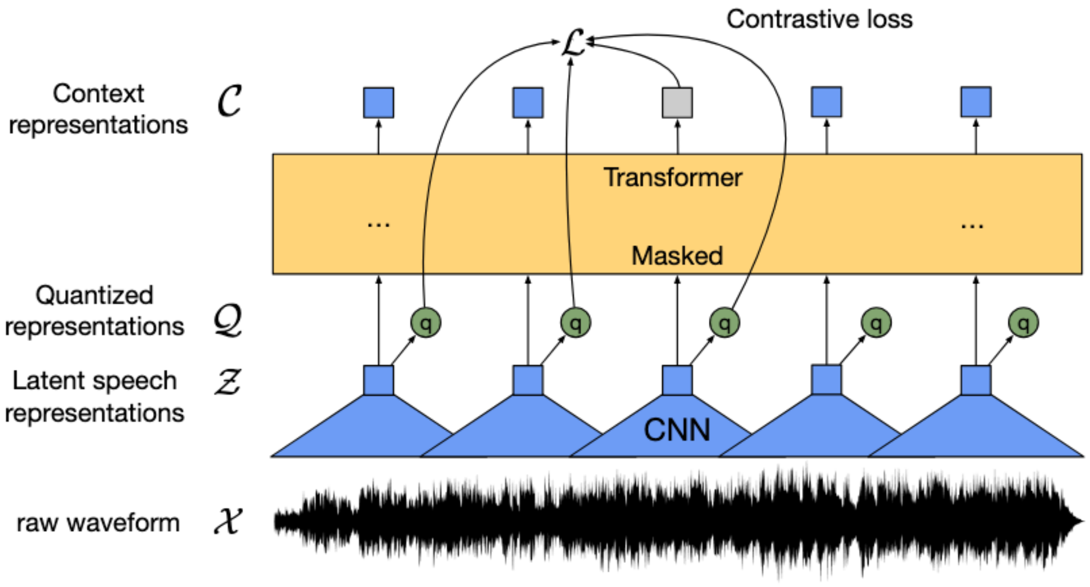


Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." *Advances in neural information processing systems* 33 (2020): 12449-12460.

Wave2vec

The **Wave2vec** model use both **transformers** and **CNNs**.

- The CNN is used to generate latent speech representations
- It predicts masked parts of the input using **self-supervision** (can be used on unlabeled datasets)
- Wave2vec can handle missing values



Baevski, Alexei, et al. "wav2vec 2.0: A framework for self-supervised learning of speech representations." *Advances in neural information processing systems* 33 (2020): 12449-12460.

Transformers and Time Series

For time series, transformers usually require additional **preprocessing** (e.g. scaling or normalization of the values).

Various tasks can be used to train self-supervised time series transformer models and leverage non-annotated data:

- **Masked prediction:** Predicting one or more “masked out” elements (tokens, features…) in a sequence: It is a **reconstruction** task

- **Contrastive learning:** Distinguish between **similar** and **dissimilar** representations.

Deep Learning for Time Series – Attention models

Recap

In This Lecture...

- **Introduction to Natural Language Processing**
- **The Attention Mechanism**
- **The Transformers Architecture**
- **Trasformer-based Models and their Applications**

Transformers: pros and cons

Pros:

- Long dependencies
- Multi-head attention can learn complex dependencies

Cons:

- Quadratic time and memory complexity
- Training is insidious
- Still limited research on time series data.