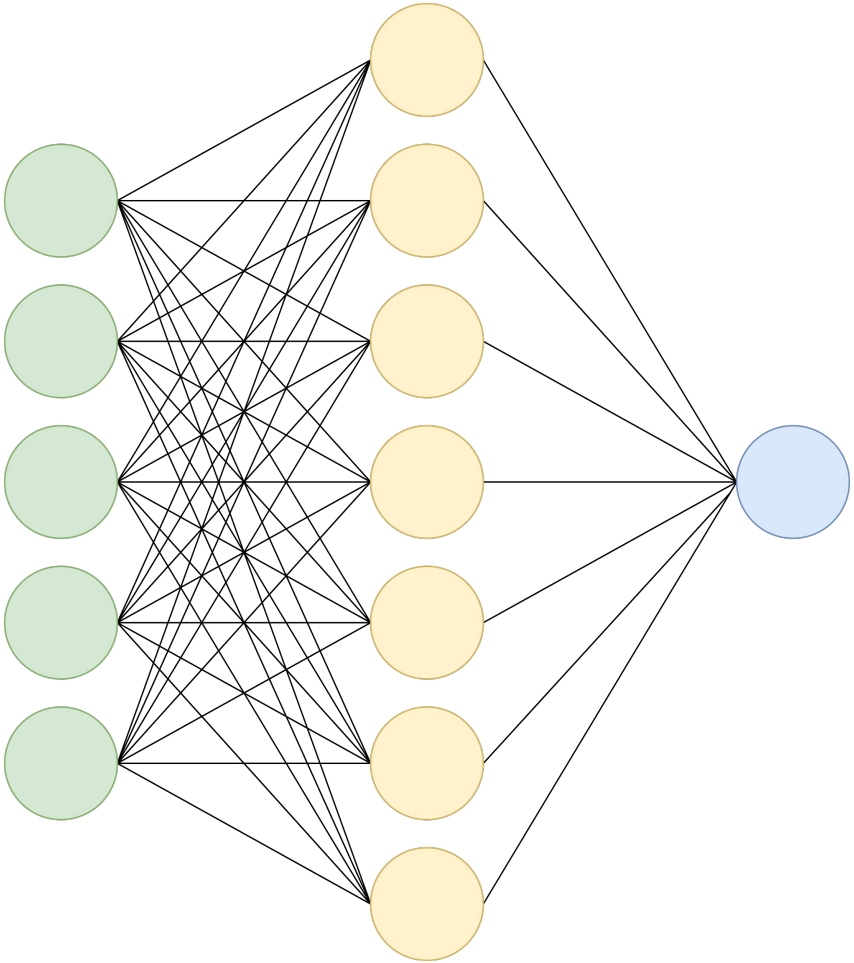
Stockholm University

Deep Learning in NLP

From RNNs to Transformers and Beyond

Thomas Vakili - thomas.vakili@dsv.su.se

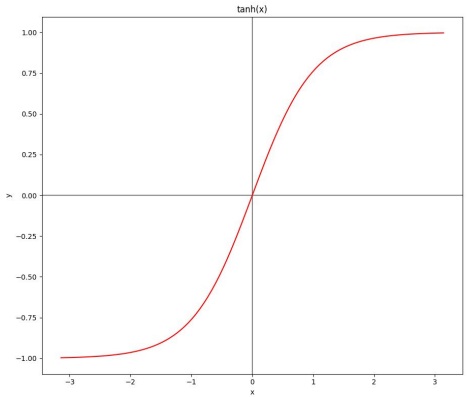


Neural Networks

Neural Networks

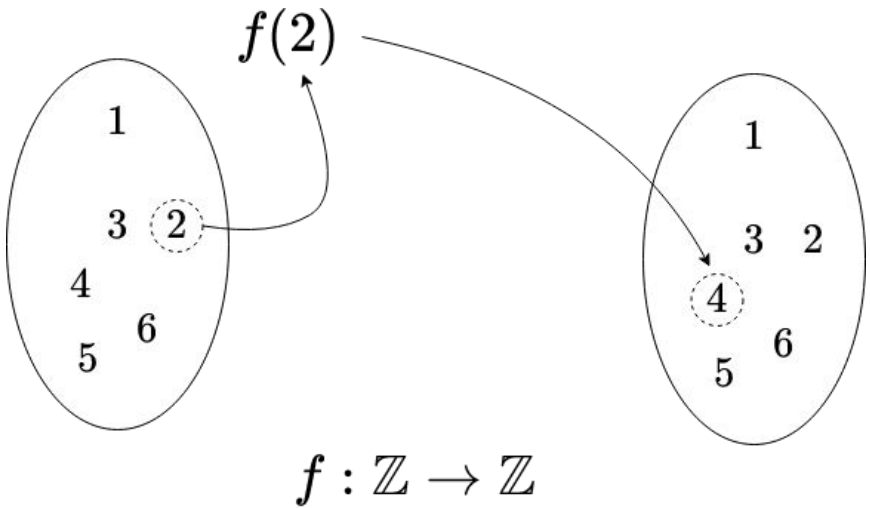
- Based mainly on two basic mathematical components:
  - Multiplying vectors and matrices
  - Non-linear activation functions

$$\begin{matrix} \vec{c}_1 & \vec{c}_2 \end{matrix} \quad \vec{r}_1$$
$$\begin{bmatrix} w & x \\ y & z \end{bmatrix} \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \vec{c}_1\vec{r}_1 & \vec{c}_2\vec{r}_1 \\ \vec{c}_1\vec{r}_2 & \vec{c}_2\vec{r}_2 \end{bmatrix}$$
$$\vec{r}_2$$



Universal Approximation Theorem

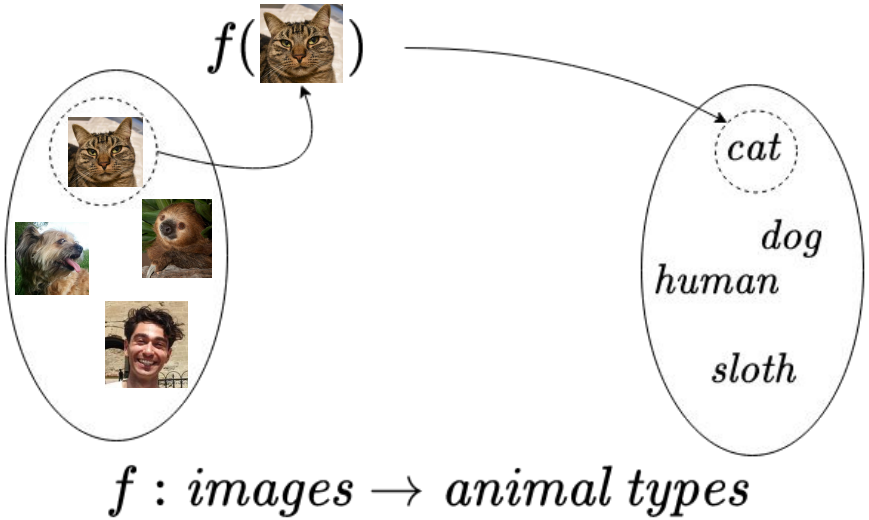
- A neural network with one hidden layer can approximate *any function**



* Given enough training data and parameters…

Universal Approximation Theorem

- A neural network with one hidden layer can approximate *any function**



* Given enough training data and parameters…

Vectorization

- Unfortunately, a computer can't process " "...
  We need a vector!

$$\begin{bmatrix} 2 \\ 32 \\ 98 \\ 226 \\ 120 \\ 64 \\ \vdots \\ 67 \\ 192 \\ 12 \\ 98 \\ 42 \end{bmatrix}$$

Vectorization

- Unfortunately, a computer can't process " "...
  We need a vector!

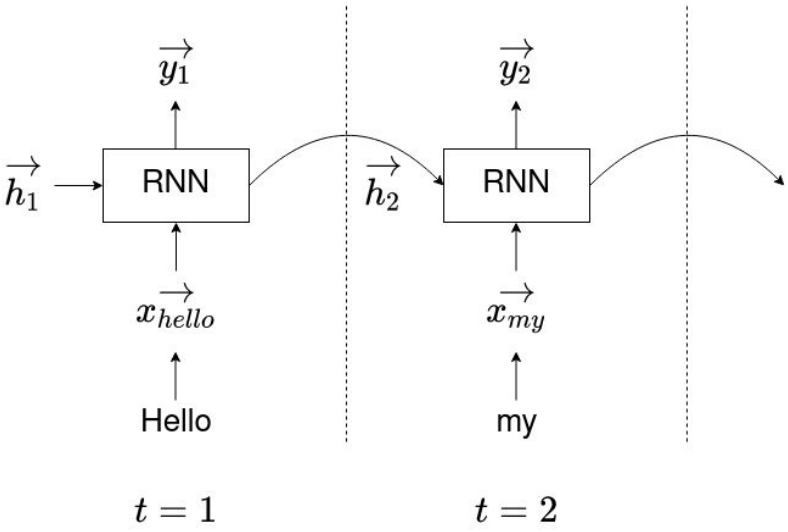
Word vectors lack order

- Words can also be vectorized, as Aron explained. But how do we handle *sequences?*

"life is work" ≠ "work is life"

BUT they result in the same set of vectors!

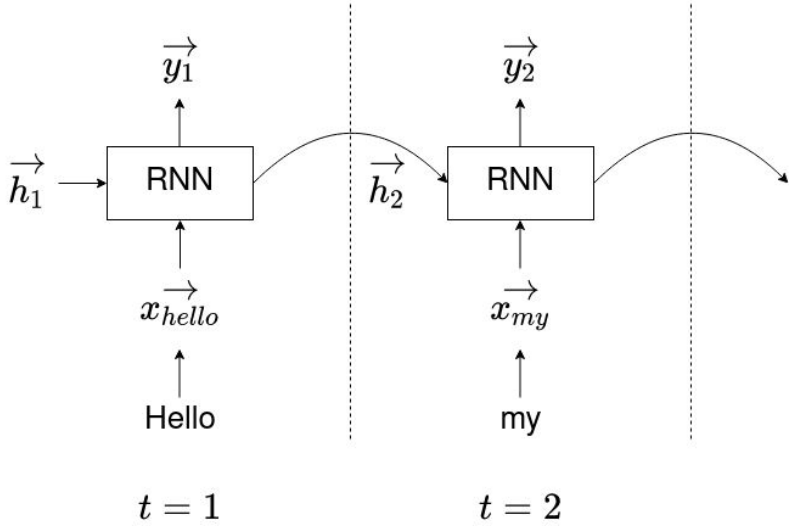
Thomas Vakili - thomas.vakili@dsv.su.se



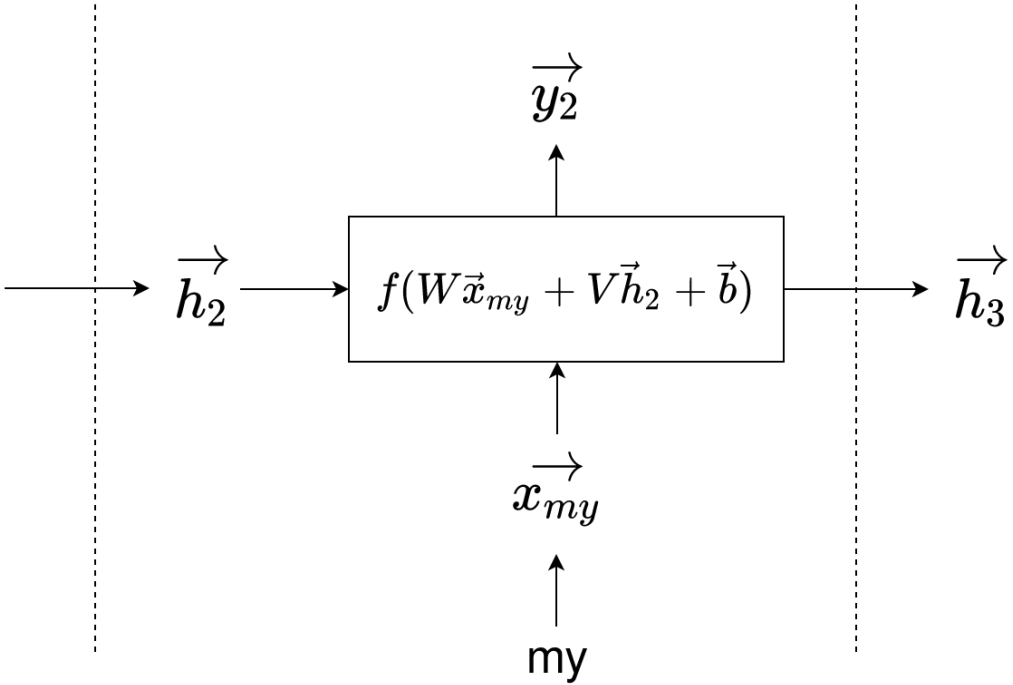
Sequences with RNNs

RNNs use a *hidden state*

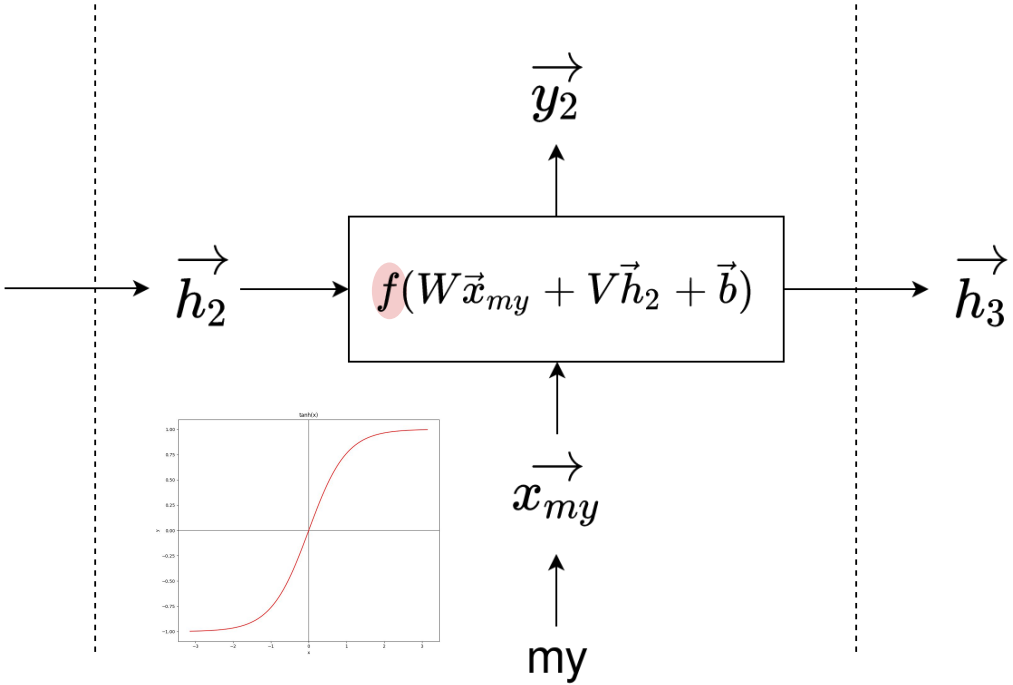
- There’s nothing hidden about the hidden state
- An RNN uses a vector $\vec{h}_t$ to **remember** the earlier states at time-step t+1



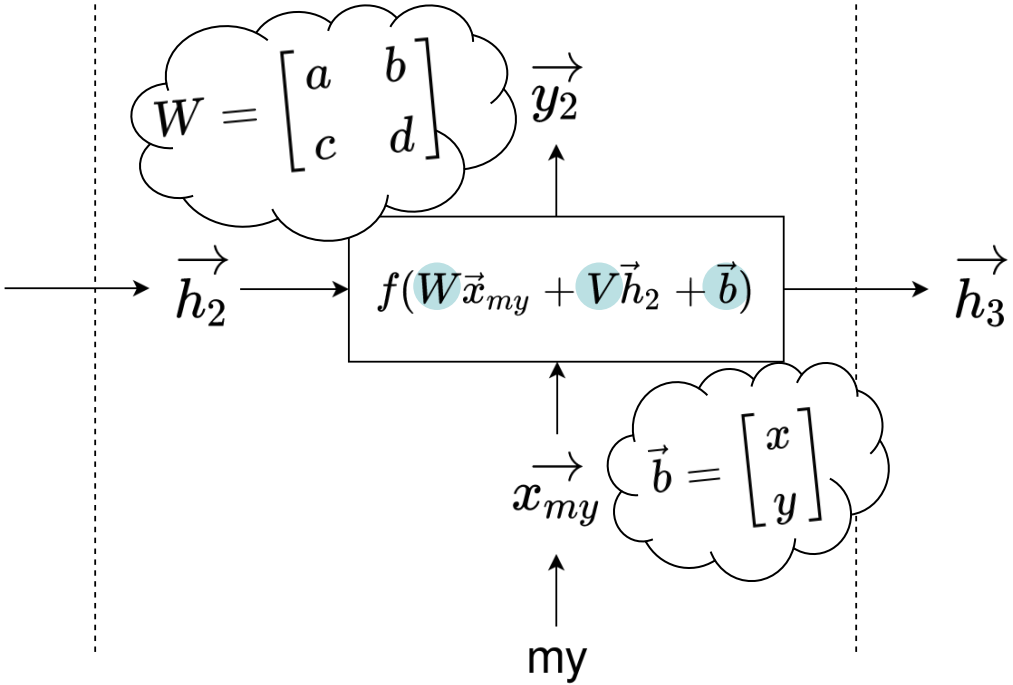
How RNNs maintain their state



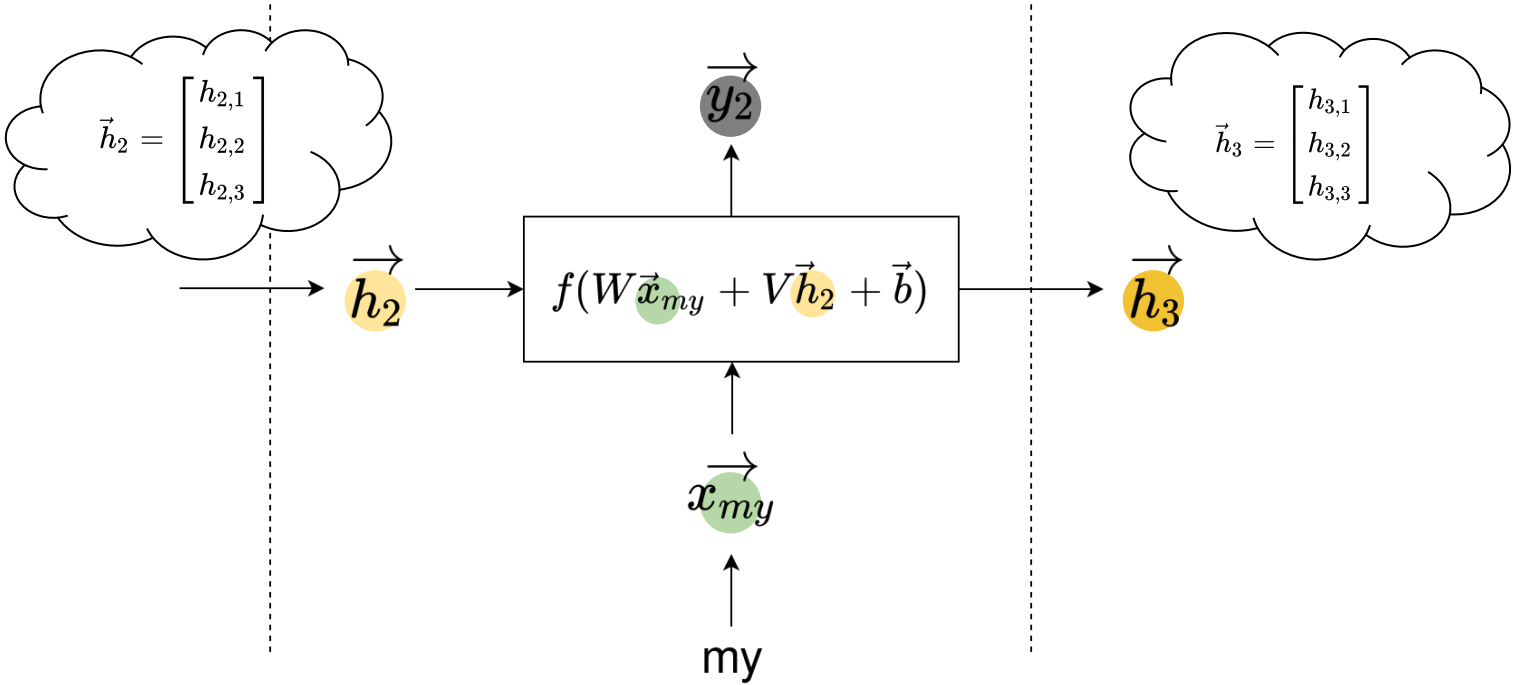
How RNNs maintain their state



How RNNs maintain their state



How RNNs maintain their state



Vanishing gradients

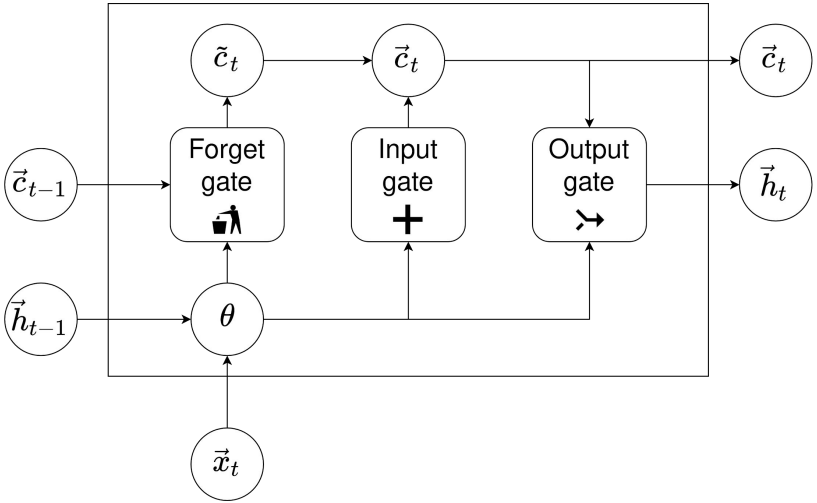
- RNNs are defined recursively: $\vec{h}_{t+1} = f(W\vec{x}_t + V\vec{h}_t + \vec{b})$
- Early states have a weaker impact on later states
- You can think of this as $\vec{h}_t$ getting crammed and leaking

$$h_4 = f(W\vec{x}_3 + Vf(W\vec{x}_2 + Vf(W\vec{x}_1 + V\vec{h}_1 + \vec{b}) + \vec{b}) + \vec{b})$$

$$h_{100} = \dots$$

$\vec{h}_2$ $\vec{h}_3$

Thomas Vakili - thomas.vakili@dsv.su.se



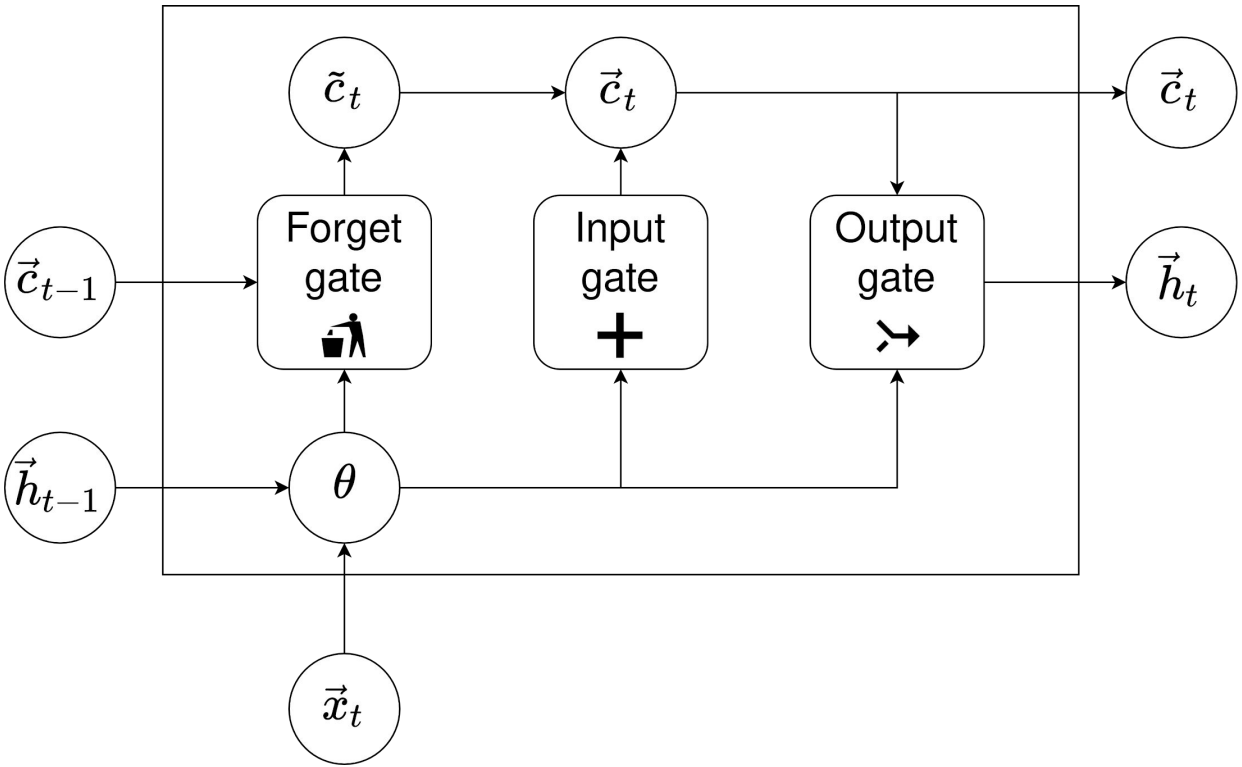
Memory management with LSTMs

Learn to remember and *to forget*

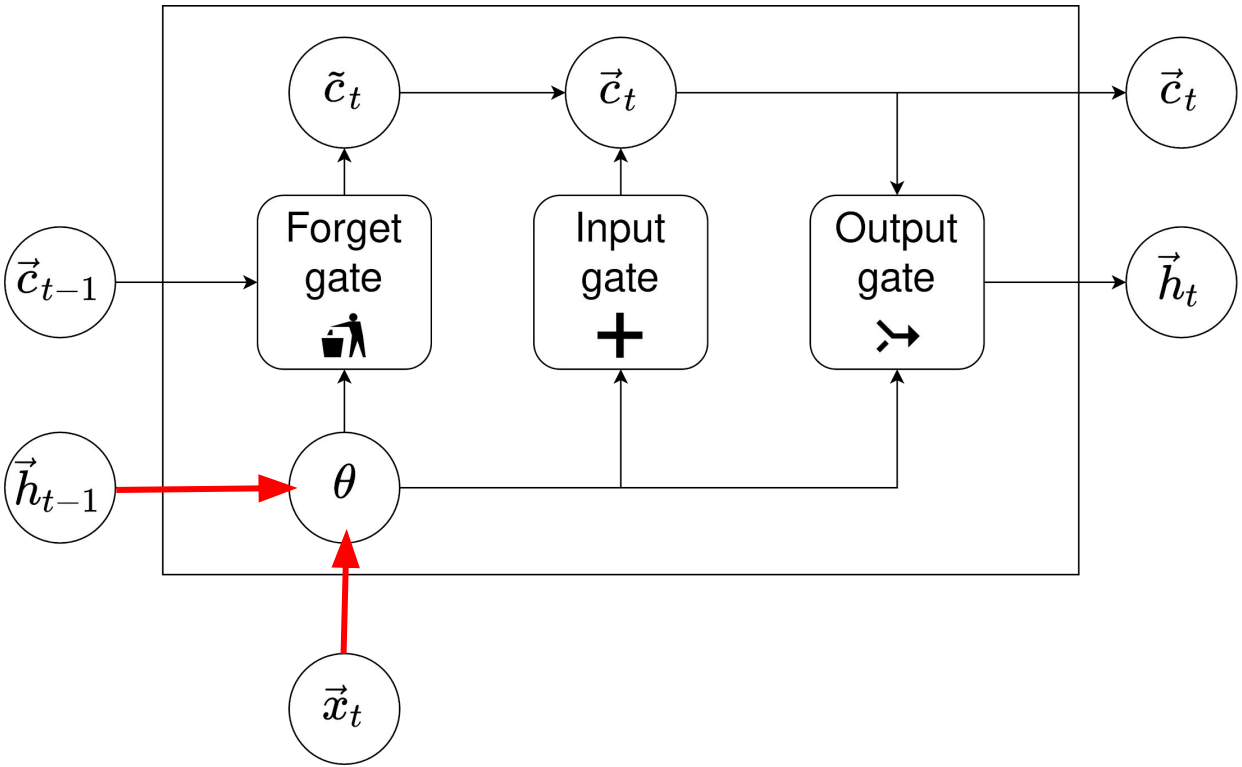
- LSTM *cells* have three *gates* (i.e., new matrices)
  - Forget gate: what is now irrelevant?
  - Input gate: what new information do we keep?
  - Output gate: produce the next hidden state

- They maintain a *cell state*: $\vec{c}_t$

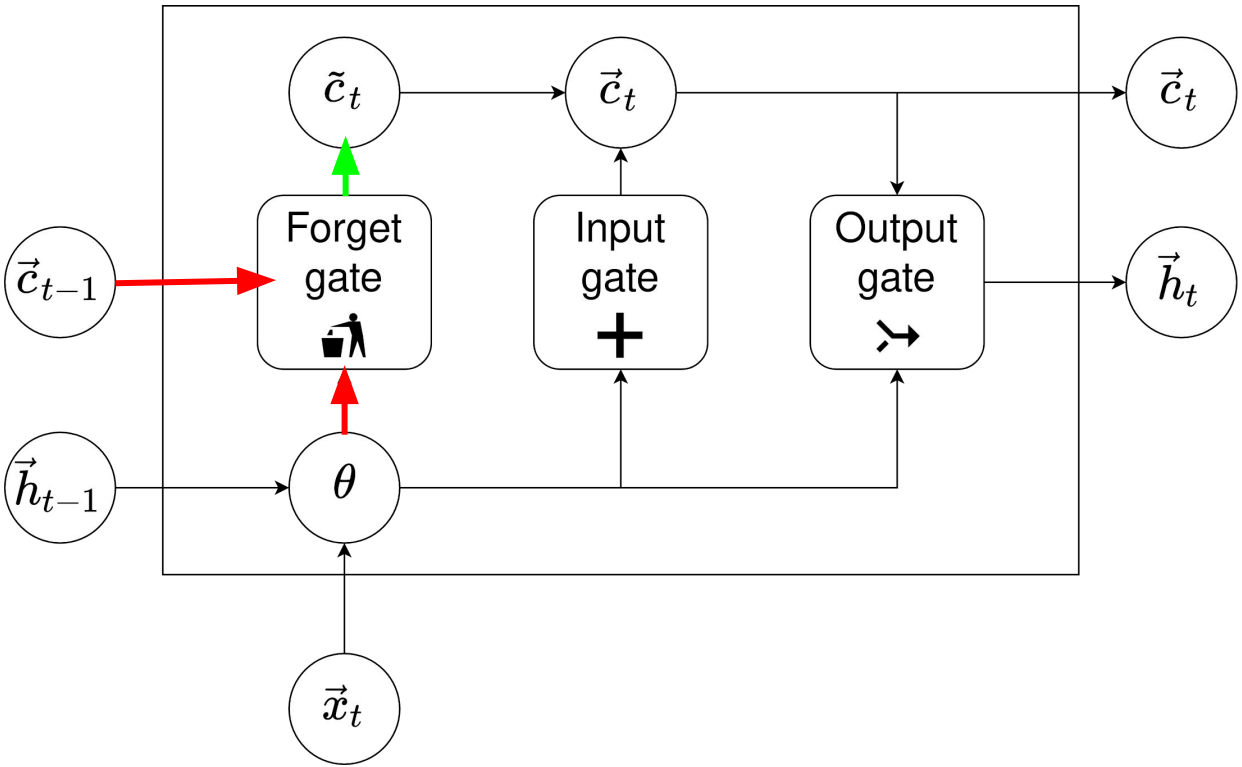
Inside the LSTM



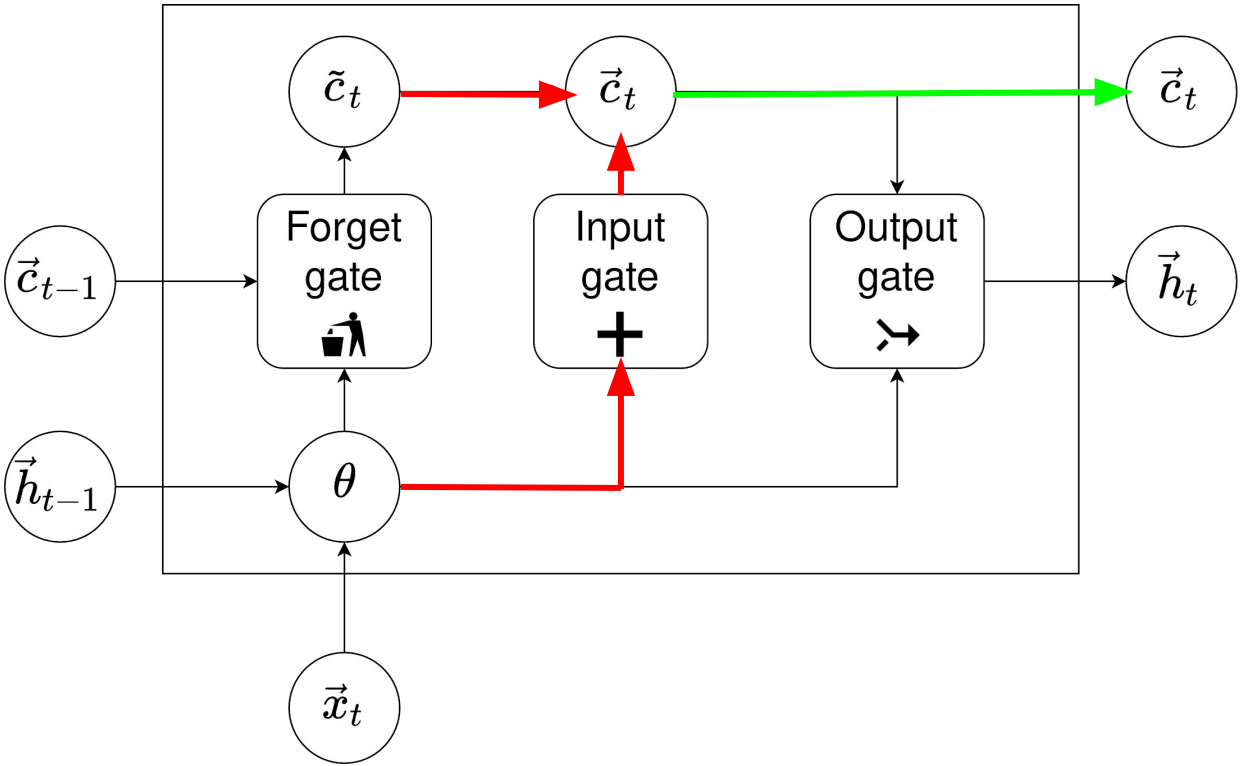
Inside the LSTM



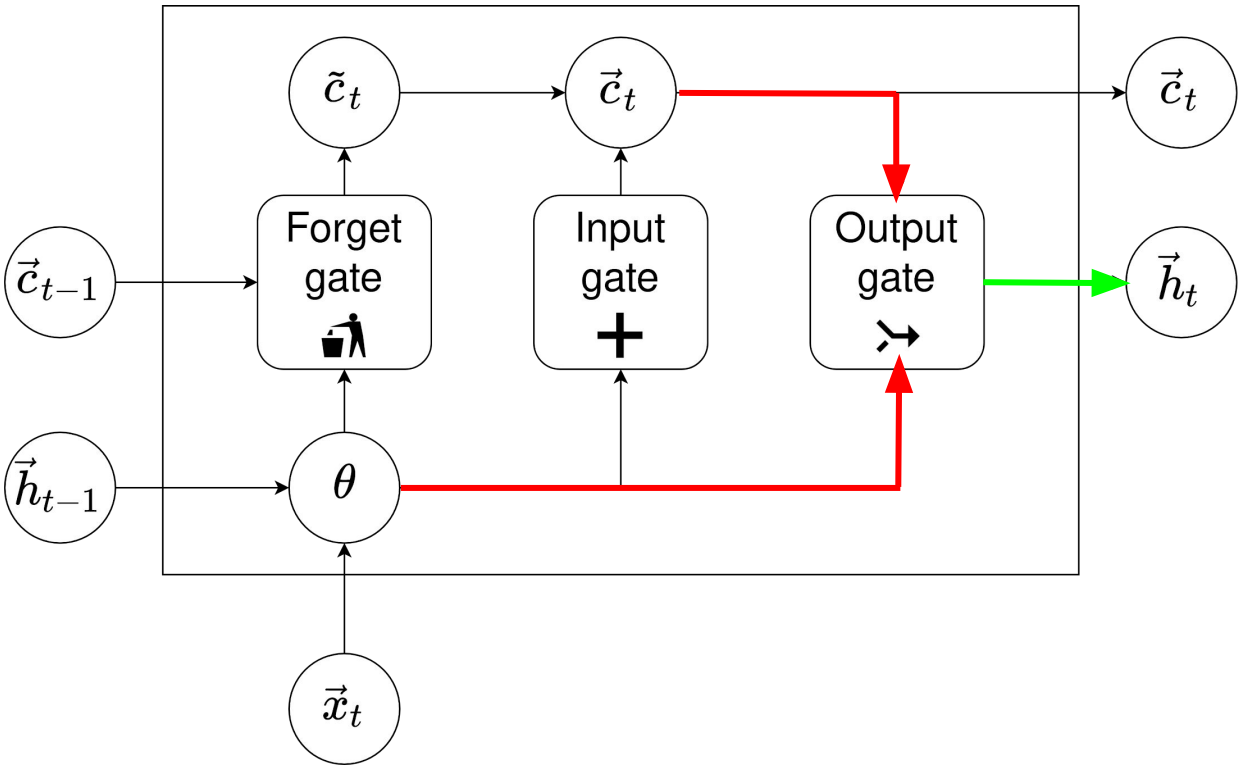
Inside the LSTM



Inside the LSTM

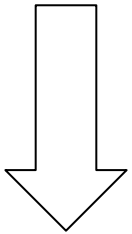


Inside the LSTM



Some “small” mathematical tweaks

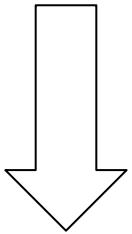
$$\vec{h}_{t+1} = f(W\vec{x}_t + V\vec{h}_t + \vec{b})$$



Thomas Vakili - thomas.vakili@dsv.su.se

Some “small” mathematical tweaks

$$\vec{h}_{t+1} = f(W\vec{x}_t + V\vec{h}_t + \vec{b})$$

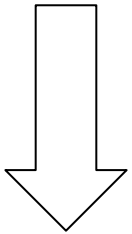


$$f_t = \sigma(\vec{b}_f + W_f\theta)$$
$$g_t = \sigma(\vec{b}_{g_1} + W_{g_1}\theta) \odot s(\vec{b}_{g_2} + W_{g_2}\theta)$$
$$q_t = \sigma(\vec{b}_q + W_q\theta)$$

$$\vec{c}_t = f_t \odot \vec{c}_{t-1} + g_t$$
$$\vec{h}_t = \tanh(\vec{c}_t) \odot q_t$$

Some “small” mathematical tweaks

$$\vec{h}_{t+1} = f(W\vec{x}_t + V\vec{h}_t + \vec{b})$$



$$f_t = \sigma(\vec{b}_f + W_f\theta)$$
$$g_t = \sigma(\vec{b}_{g_1} + W_{g_1}\theta) \odot s(\vec{b}_{g_2} + W_{g_2}\theta)$$
$$q_t = \sigma(\vec{b}_q + W_q\theta)$$

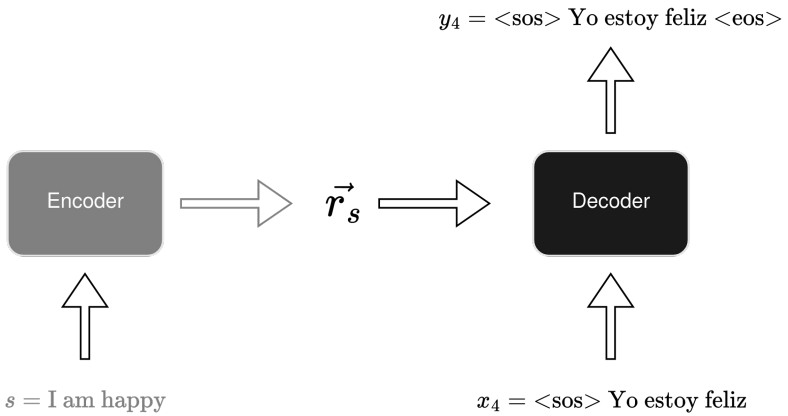
$$\vec{c}_t = f_t \odot \vec{c}_{t-1} + g_t$$
$$\vec{h}_t = \tanh(\vec{c}_t) \odot q_t$$

Pros and cons

LSTMs handle long-term dependencies quite well!

However:

- The latest input still disproportionally affects the state
- Data must be processed sequentially, word for word
- They handle transfer learning poorly



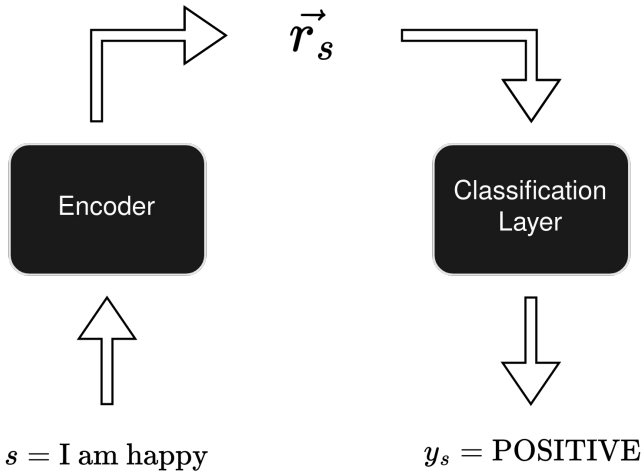
Encoders, decoders and combinations thereof

The three main architectures

- Encoder models
  - Create embeddings and *classify*
- Decoder models
  - *Generate* the next token based on previous input
- Encoder-decoder models
  - *Generate* the next token based on previous **output**
  - **AND** on an encoding of the input

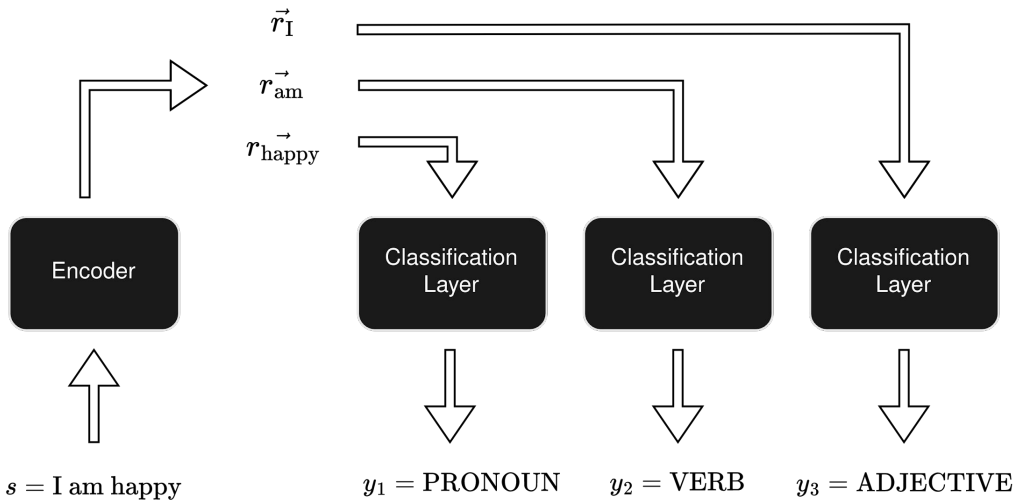
Encoder models

Encode input into vector representations, then classify



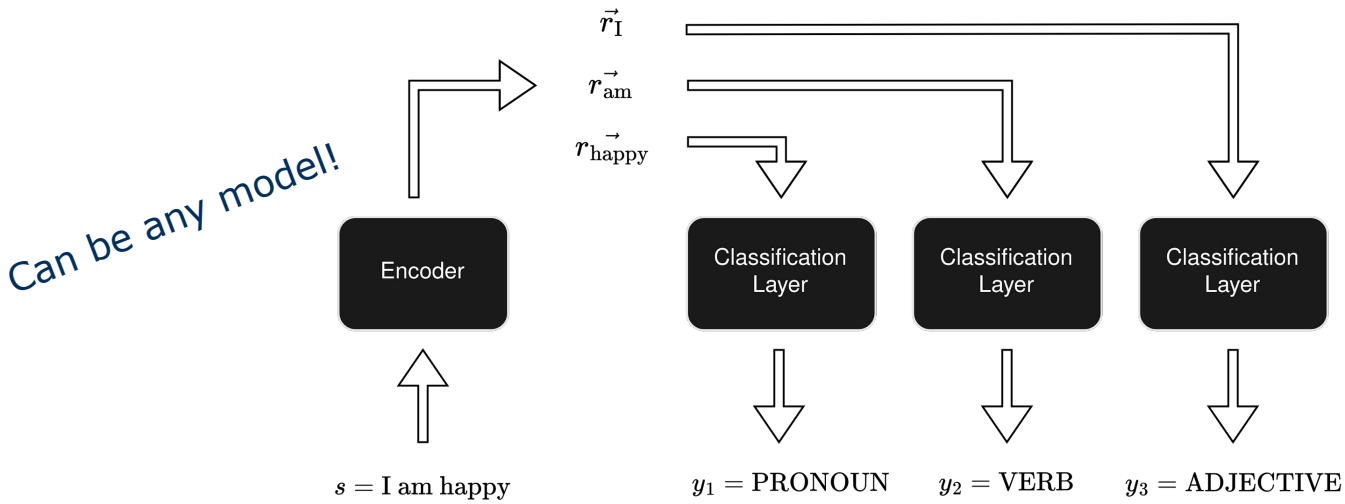
Encoder models

Encode input into vector representations, then classify



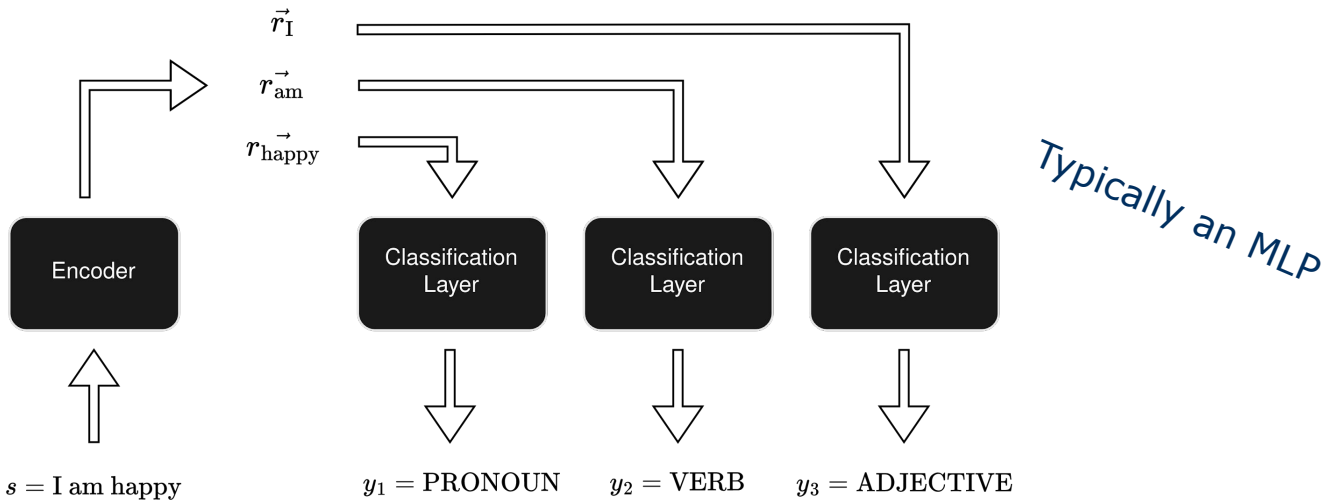
Encoder models

Encode input into vector representations, then classify

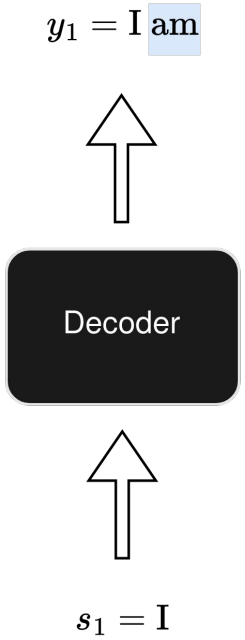


Encoder models

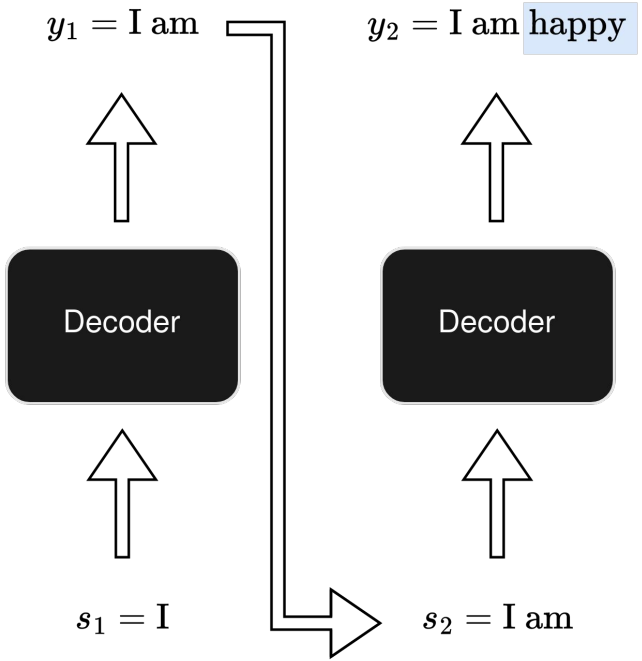
Encode input into vector representations, then classify



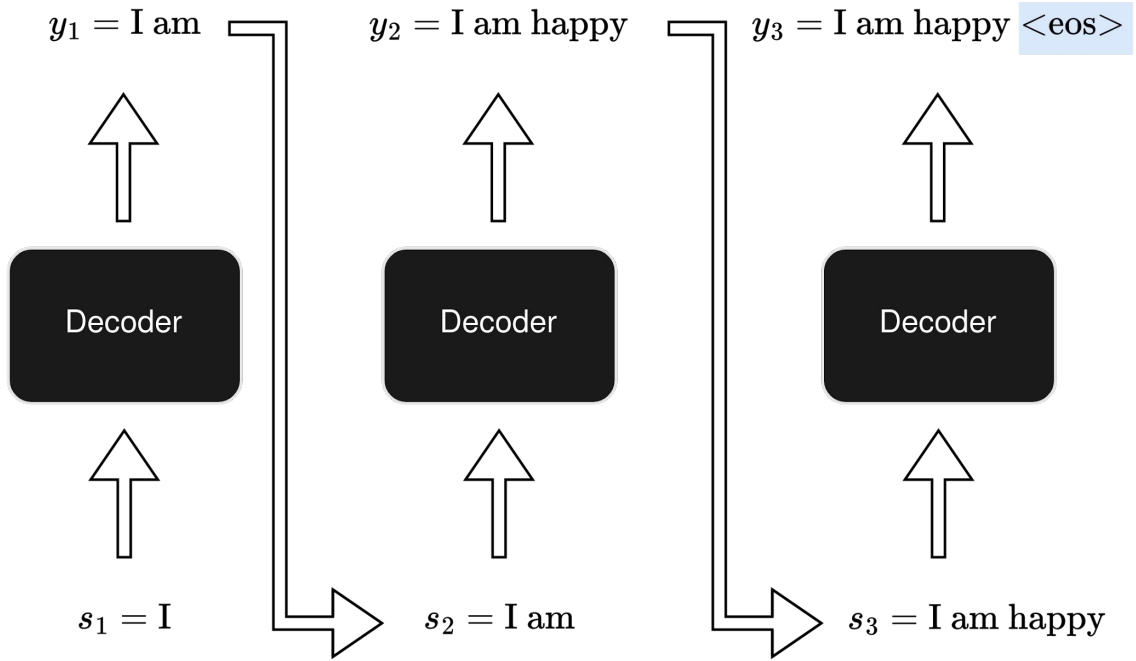
Decoder models



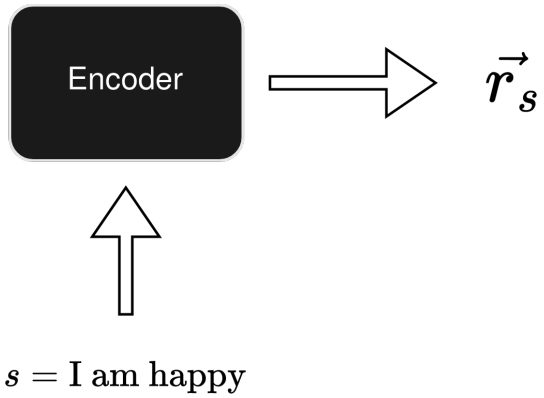
Decoder models



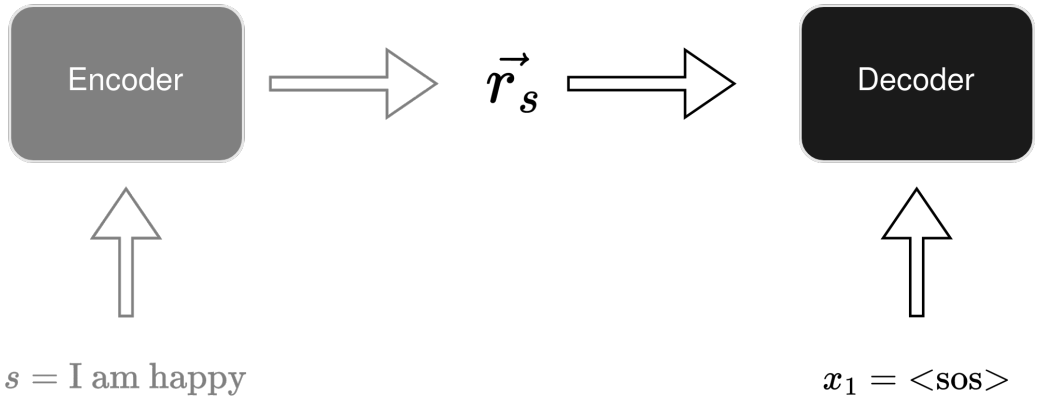
Decoder models



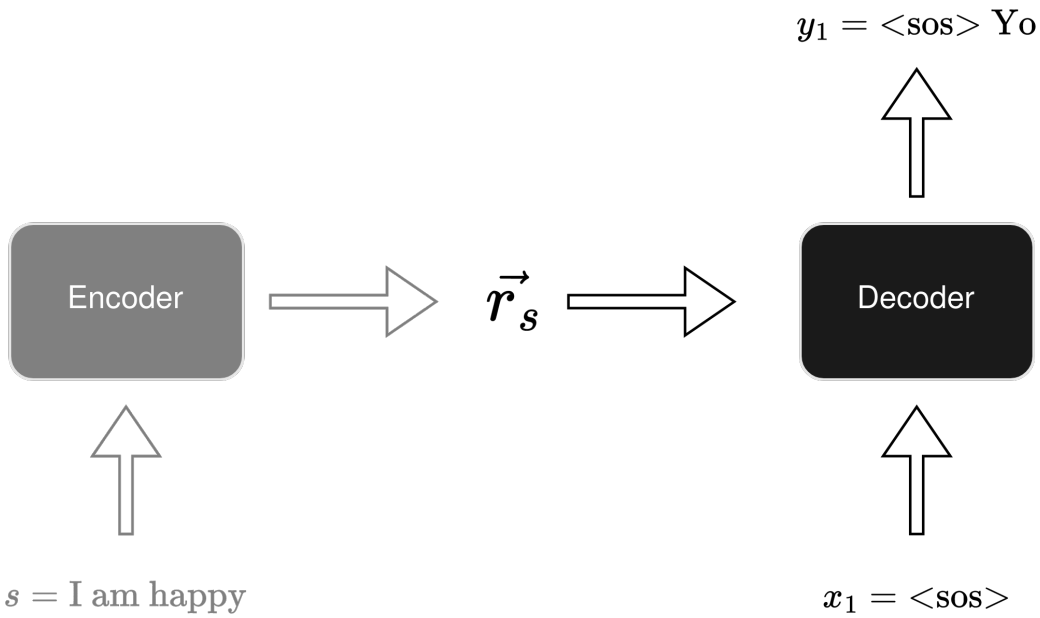
Combined encoder-decoder models



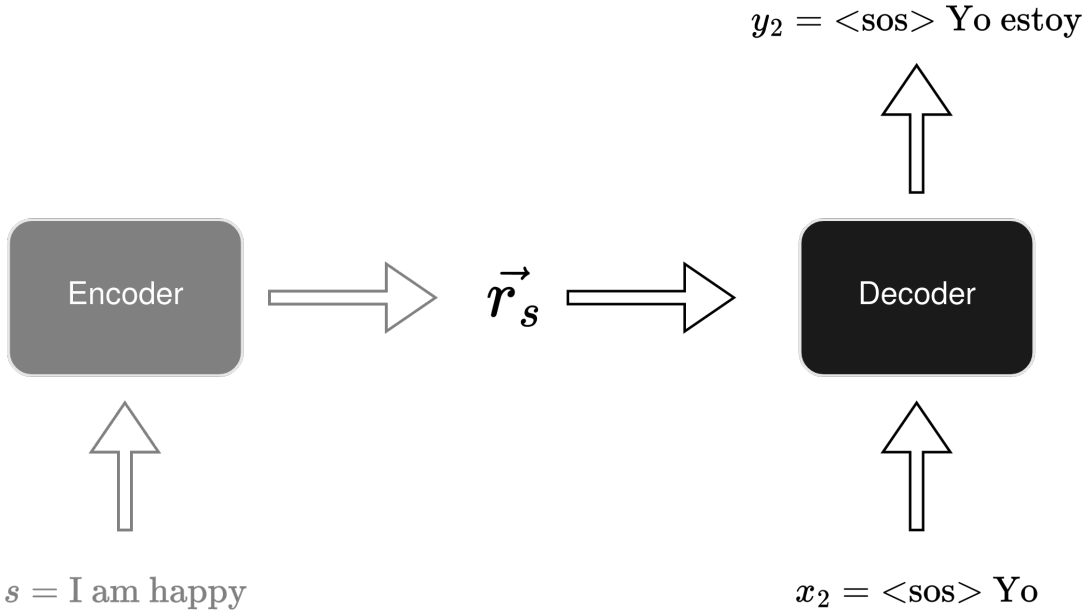
Combined encoder-decoder models



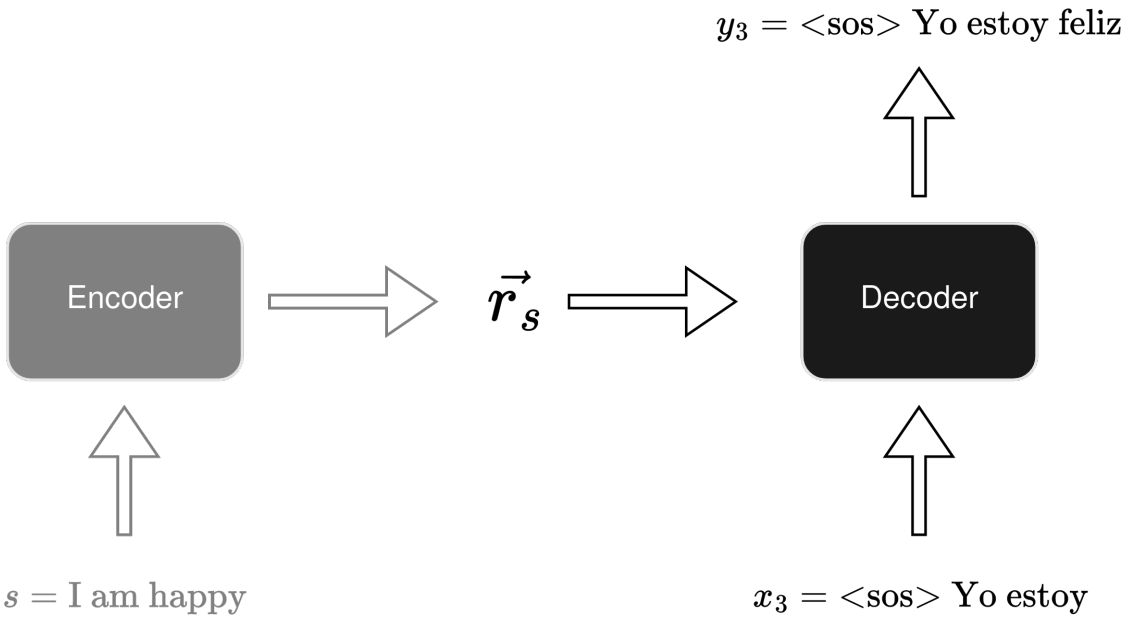
Combined encoder-decoder models



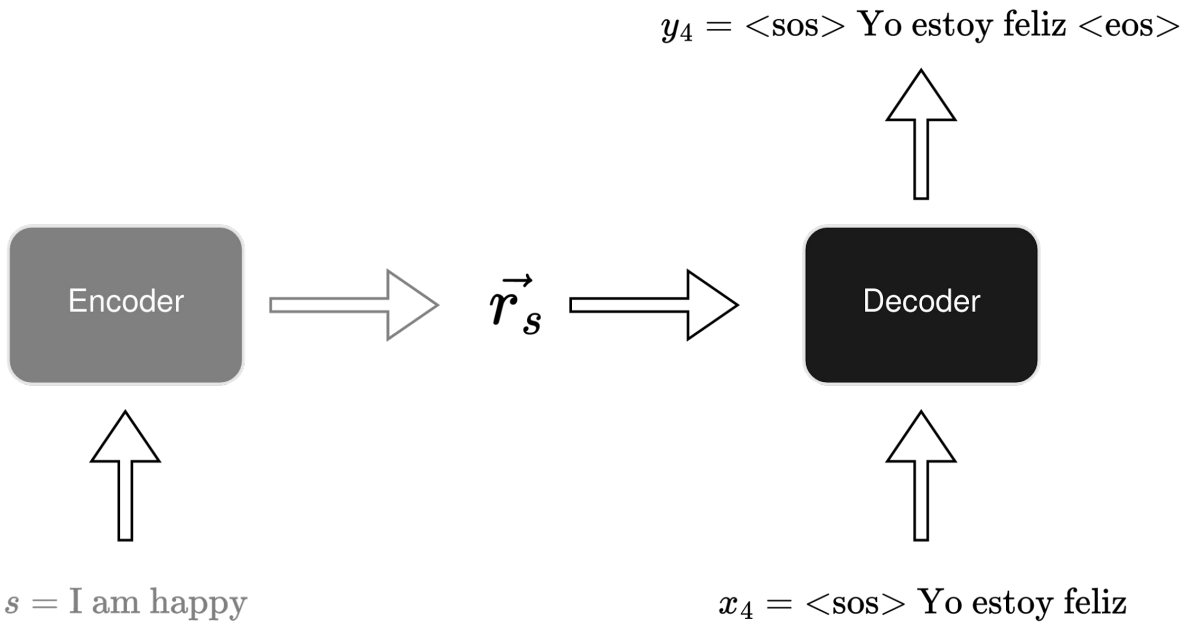
Combined encoder-decoder models



Combined encoder-decoder models

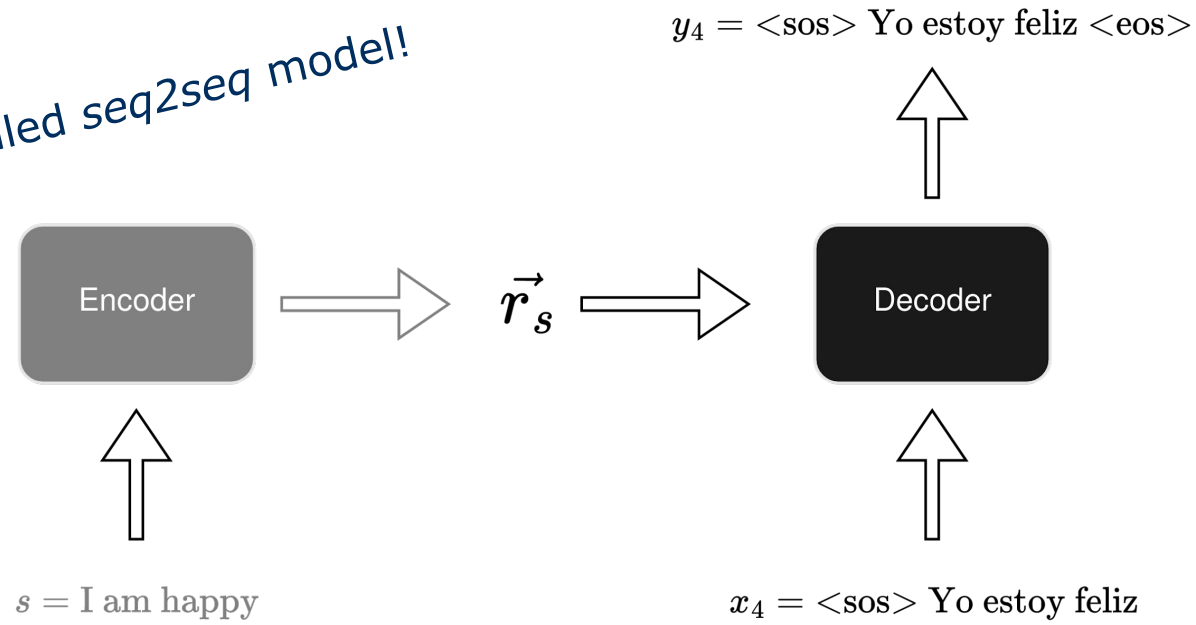


Combined encoder-decoder models



Combined encoder-decoder models

Sometimes called seq2seq model!



Different use cases

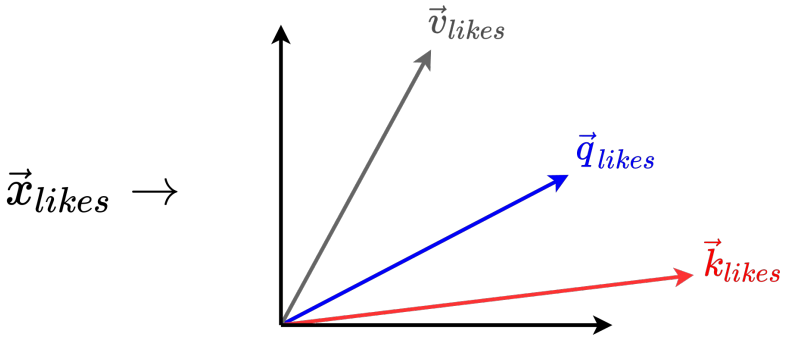
Many different NLP tasks:

- Text classification (e.g., sentiment analysis)
  - Encoder
- Token classification (e.g., named entity recognition)
  - Encoder, encoder-decoder, decoder
- Sequence transduction (e.g., machine translation)
  - Encoder-decoder, decoder

Stockholm University

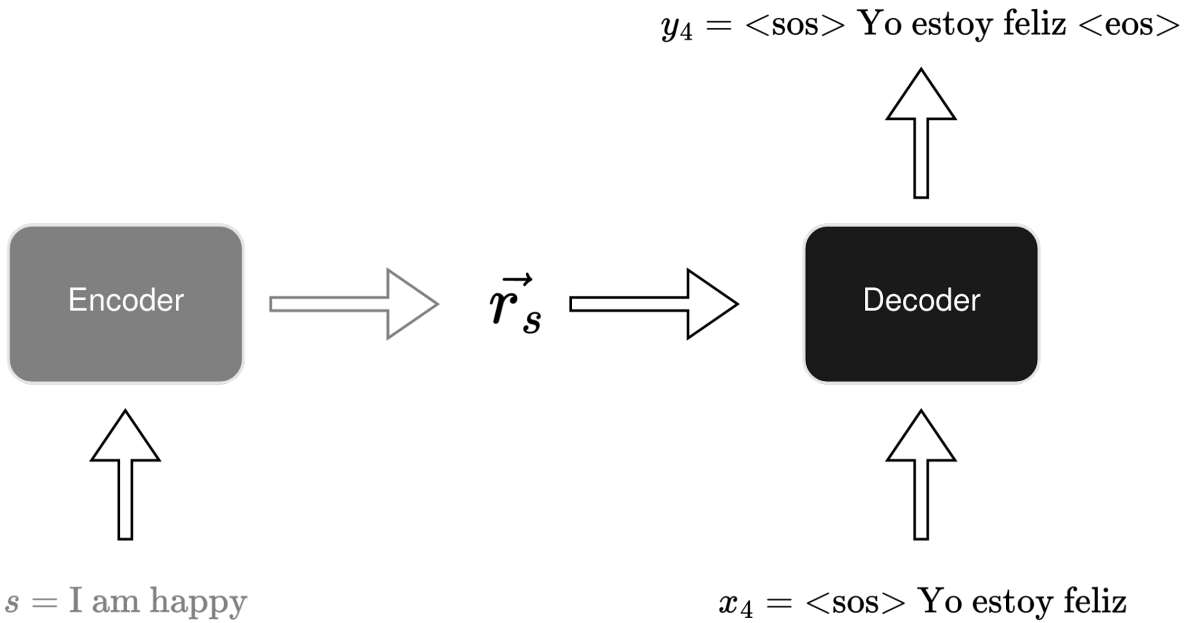
Time for a break!

Stockholm University

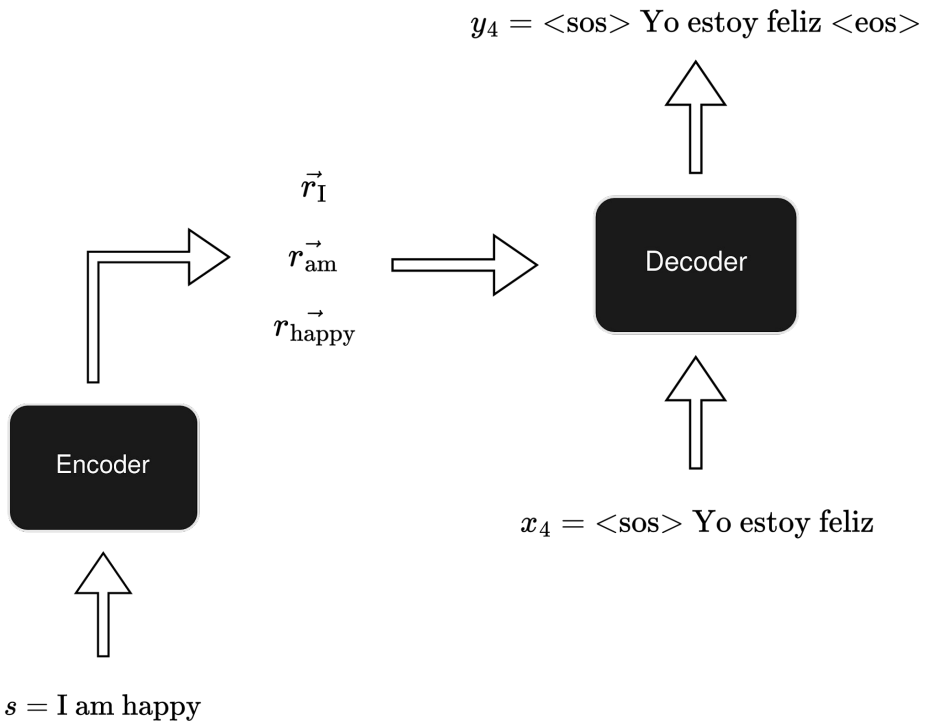


Pre-trained Transformers

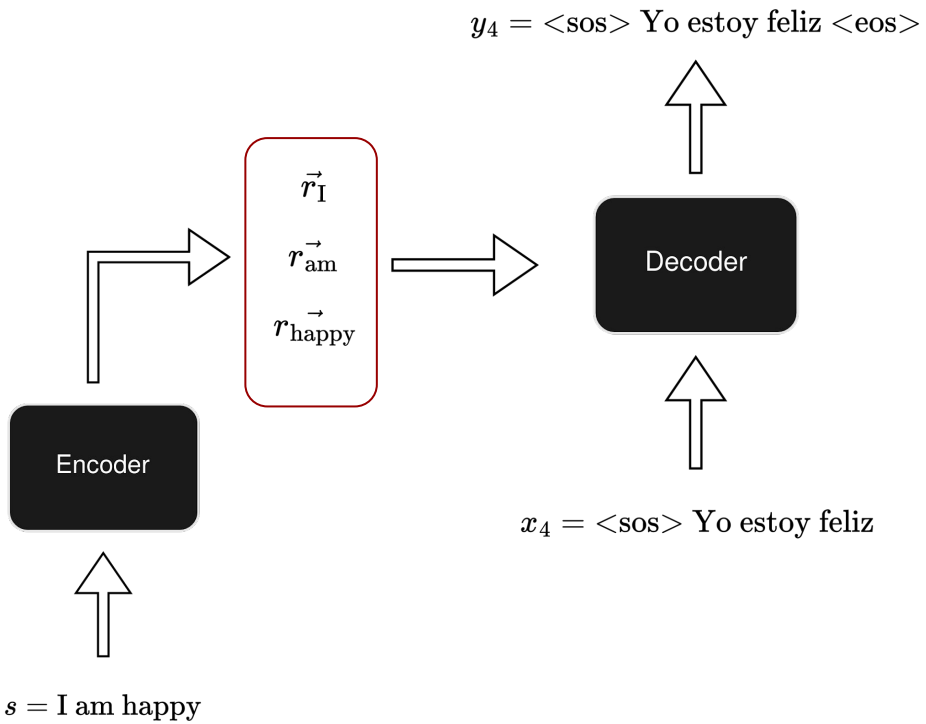
Problem: still *one* vector for memory



Idea: use all hidden states!



How to combine them?



Attention is all you need

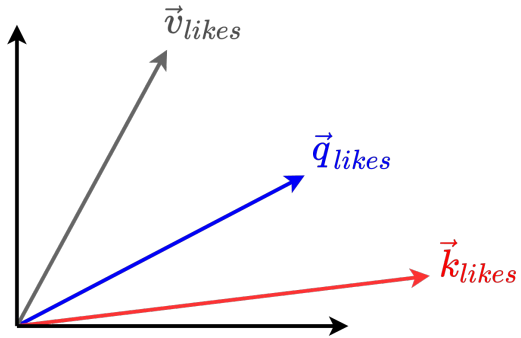
- Attention was first introduced to improve LSTMs
- Vaswani et al. (2017) realized that attention lets us do away with sequential processing!
- This resulted in a new deep learning architecture – the *Transformer* was born

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. *Advances in neural information processing systems*, *30*.

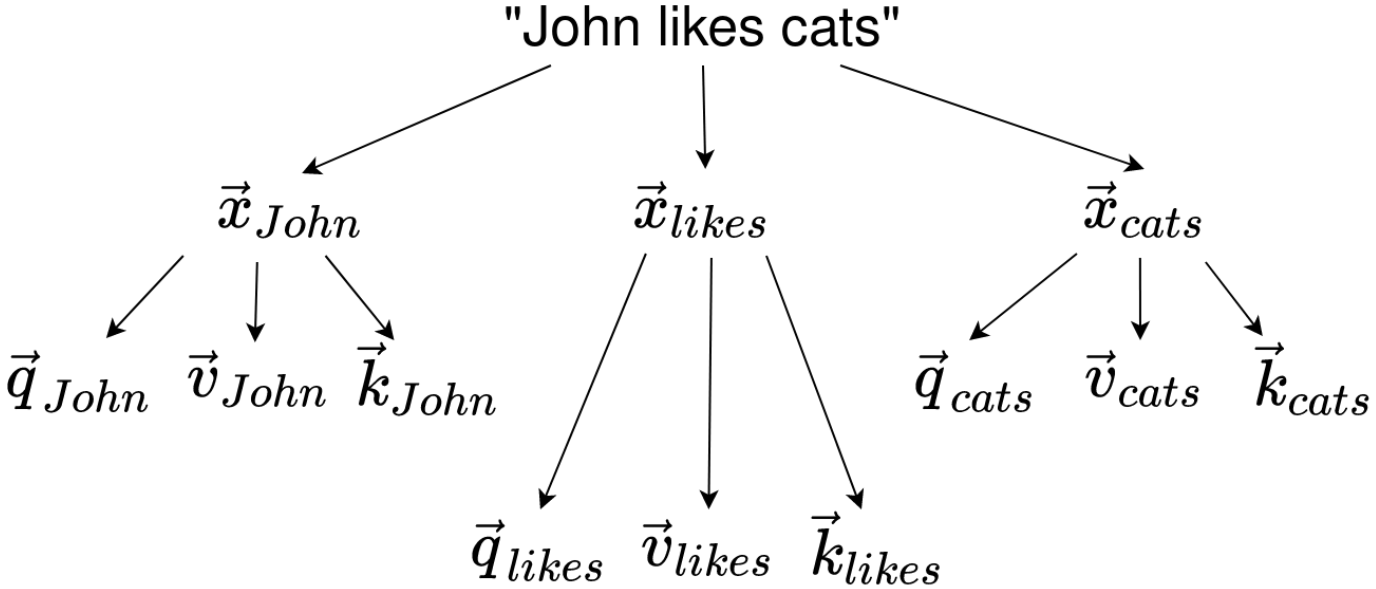
Self-attention

- Self-attention considers all combinations of words *directly* instead of using a hidden state
- This results in *contextualized representations*
- This is done by using *query*, *key* and *value vectors* created from all words

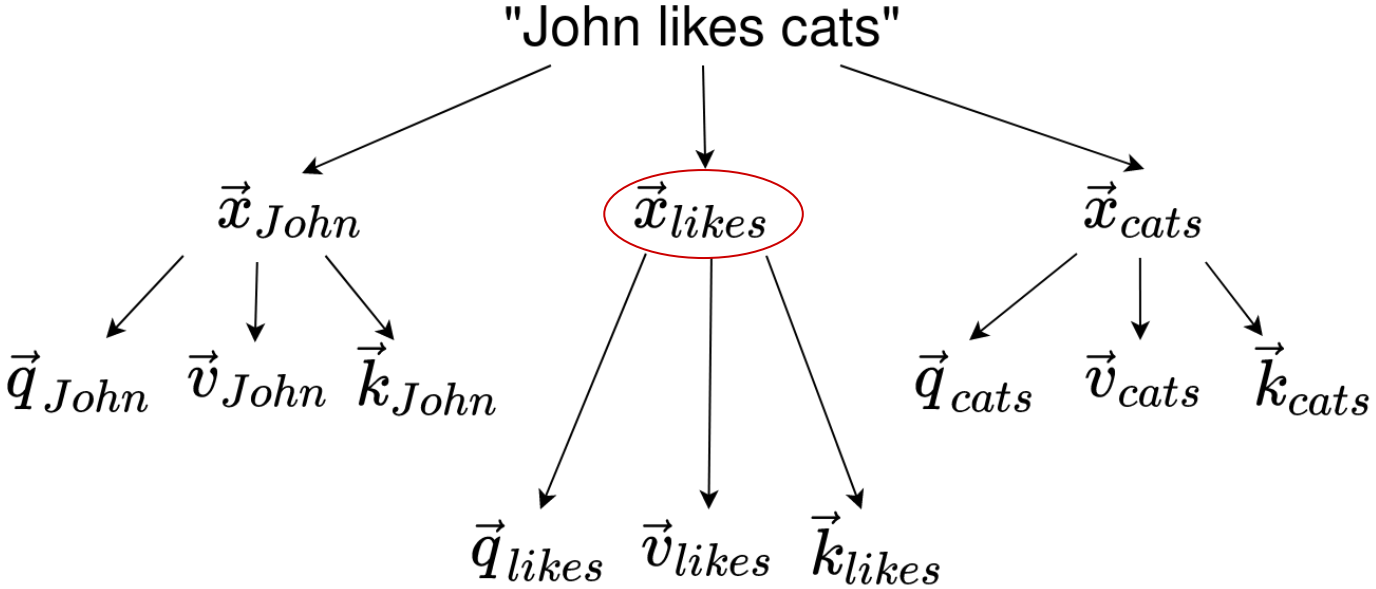
$\vec{x}_{likes} \rightarrow$



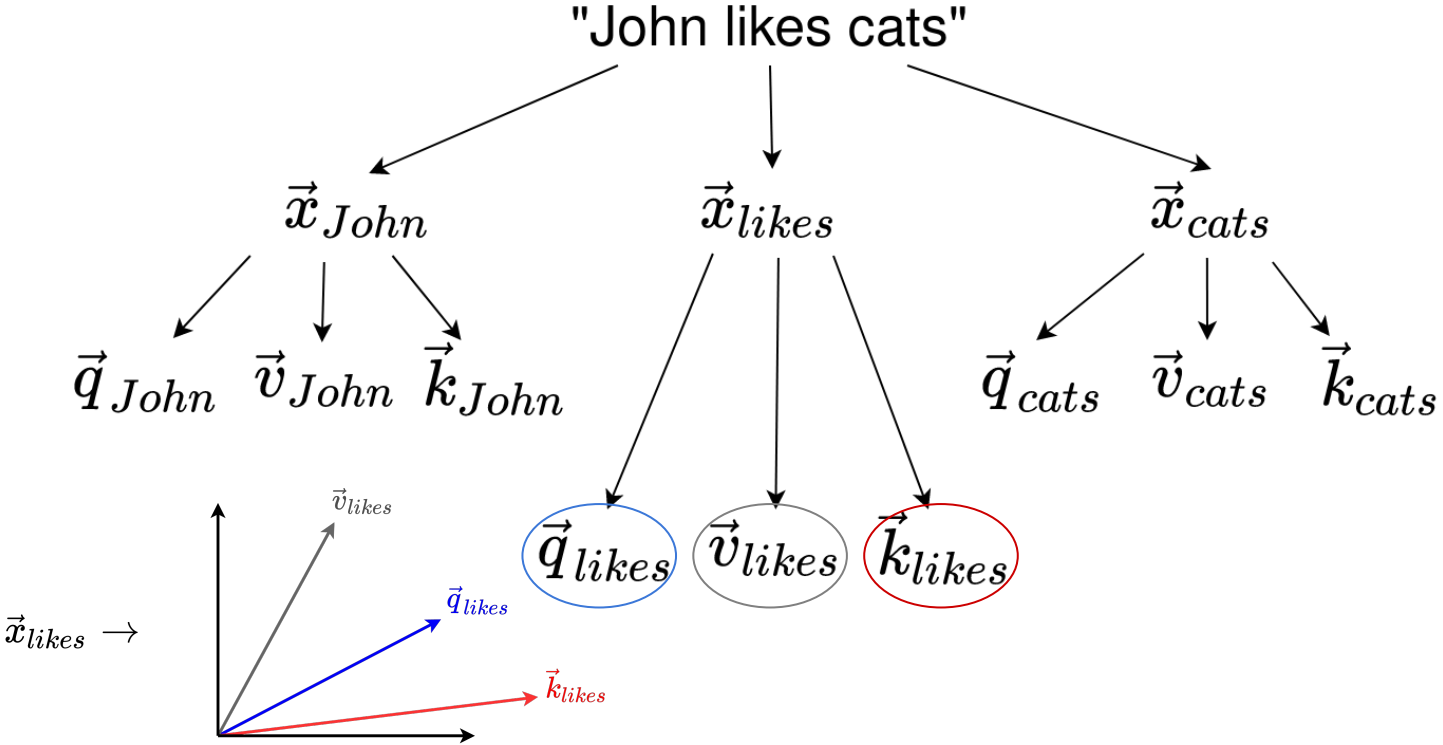
Self-attention



Self-attention

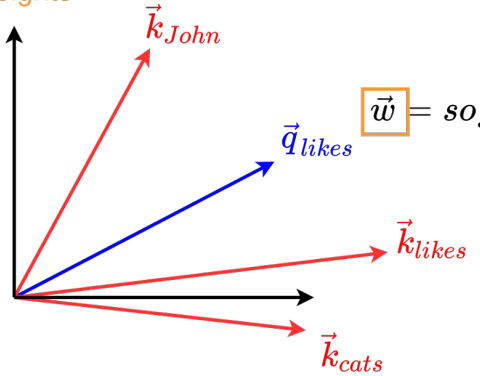


Self-attention



Creating the contextualized representation

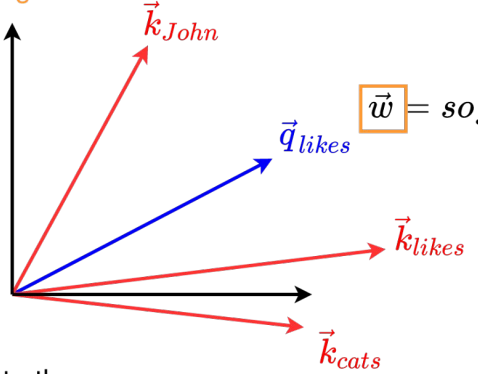
First calculate
relevancy weights



$$\vec{w} = softmax\left(\begin{bmatrix} \vec{q}_{likes}\vec{k}_{John} \\ \vec{q}_{likes}\vec{k}_{likes} \\ \vec{q}_{likes}\vec{k}_{cats} \end{bmatrix}\right)$$

Creating the contextualized representation

First calculate
relevancy weights

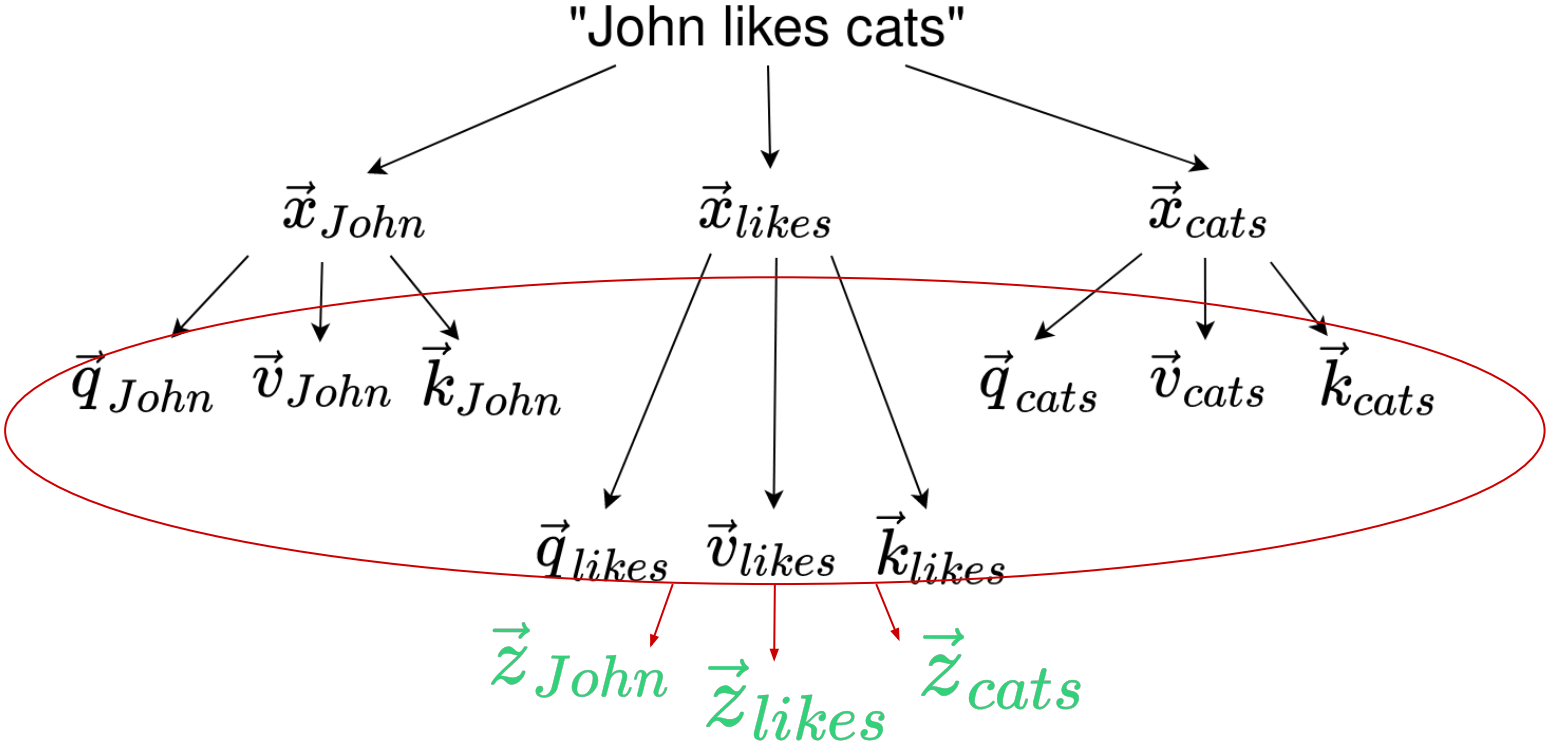


$$\boxed{\vec{w}} = softmax\left(\begin{bmatrix} \vec{q}_{likes}\vec{k}_{John} \\ \vec{q}_{likes}\vec{k}_{likes} \\ \vec{q}_{likes}\vec{k}_{cats} \end{bmatrix}\right)$$

Then calculate the
weighted representation

$$\vec{w} = \begin{bmatrix} 0.34 \\ 0.57 \\ 0.09 \end{bmatrix} \implies \boxed{\vec{z}_{likes}} = 0.34 \cdot \vec{v}_{John} + 0.57 \cdot \vec{v}_{likes} + 0.09 \cdot \vec{v}_{cats}$$

Self-attention



Examples of self-attention

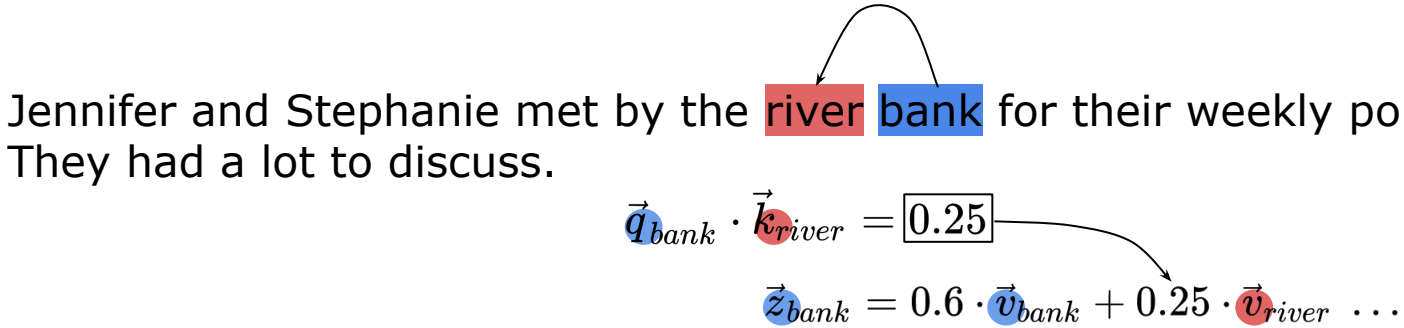
Jennifer and Stephanie met by the river bank for their weekly politics club. They had a lot to discuss.

Examples of self-attention

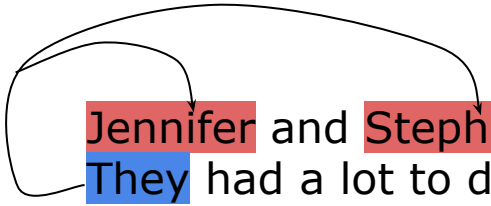
Jennifer and Stephanie met by the river bank for their weekly politics club. They had a lot to discuss.

$$\vec{q}_{bank} \cdot \vec{k}_{river} = 0.25$$

Examples of self-attention



Examples of self-attention



Jennifer and Stephanie met by the river bank for their weekly politics club. They had a lot to discuss.

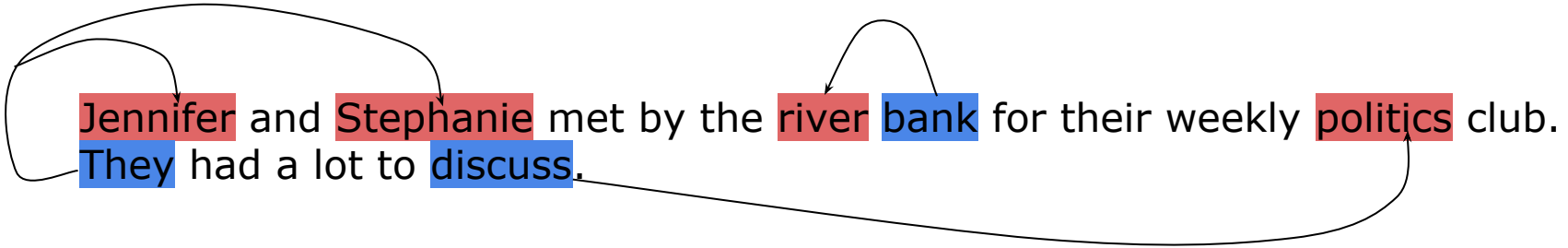
$$\vec{z}_{They} = 0.4 \cdot \vec{v}_{Jennifer} + 0.4 \cdot \vec{v}_{Stephanie} \ \ldots$$

Examples of self-attention

Jennifer and Stephanie met by the river bank for their weekly politics club. They had a lot to discuss.

$$\vec{z}_{discuss} = 0.5 \cdot \vec{v}_{discuss} + 0.4 \cdot \vec{v}_{politics} \ldots$$

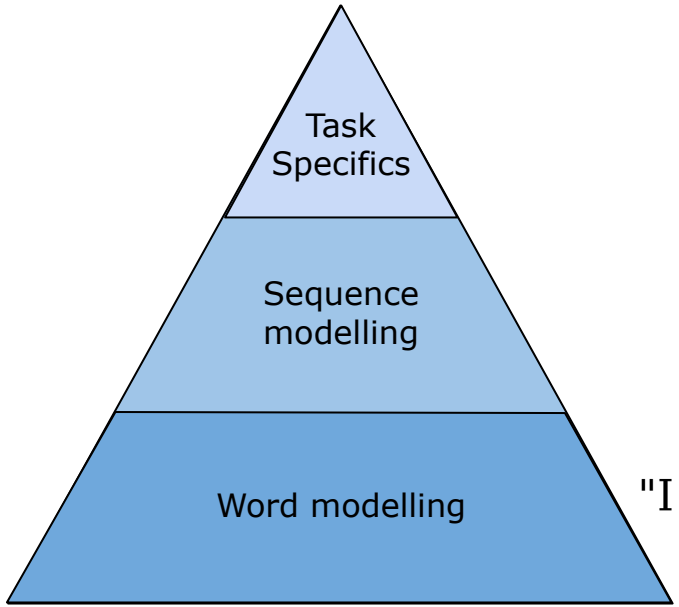
Multi-headed attention



Why Transformers are great

- Long term dependencies easy to learn because every word “sees” every other word
- Doing away with sequential processing allows for parallelization
- Most importantly: they are great at *transfer learning*

The NLP pyramid

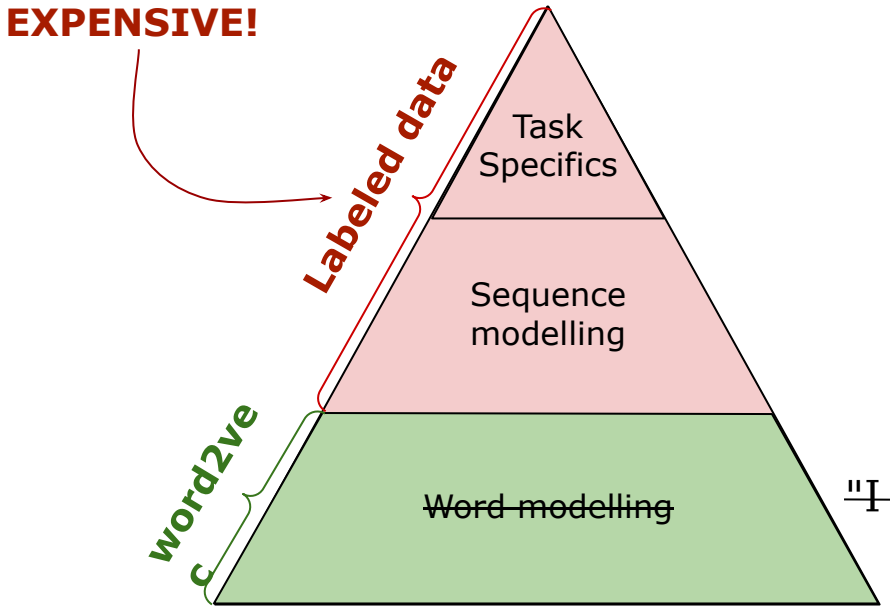


$$\vec{s}_{\text{"I am happy"}} \rightarrow positive$$

$$\{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\} \rightarrow \vec{s}_{\text{"I am happy"}}$$

$$\text{"I am happy"} \rightarrow \{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\}$$

LSTMs need more labeled data

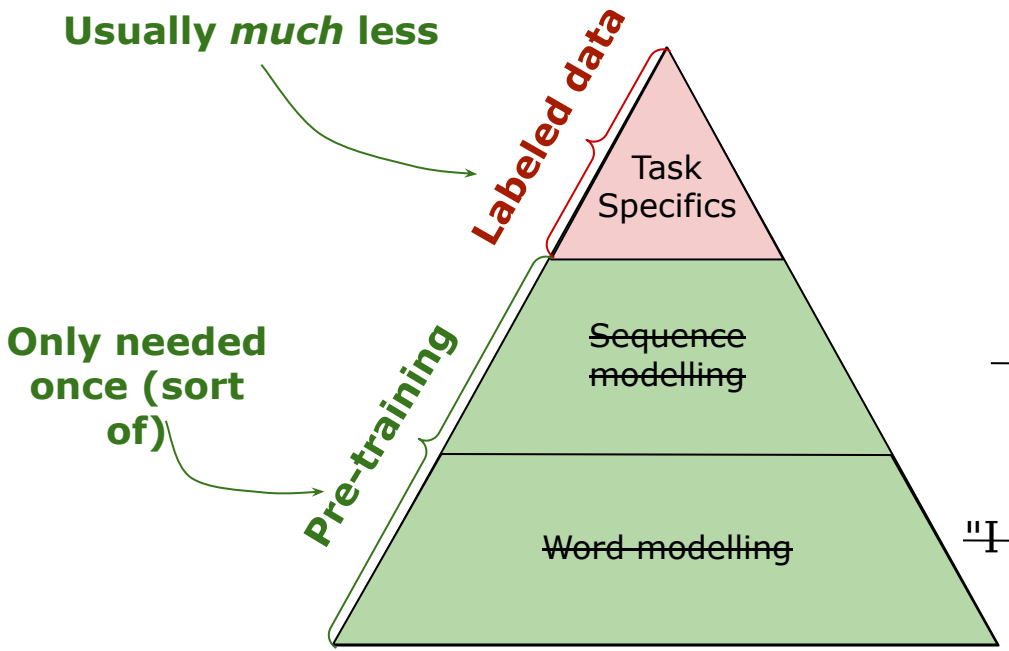


$\vec{s}$"I am happy" $\rightarrow positive$

$\{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\} \rightarrow \vec{s}$"I am happy"

~~"I am happy" $\rightarrow \{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\}$~~

Pre-trained transformers need less



$\vec{s}$"I am happy" $\rightarrow$ *positive*

~~$\{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\} \rightarrow \vec{s}$"I am happy"~~

~~"I am happy" $\rightarrow \{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\}$~~

Language Models

- Language modelling is traditionally done "left-to-right"
- GPT-{1, 2, 3, 4, 5?} are *autoregressive decoder* models

Thomas Vakili is a PhD student at **DSV**

$$x_{i+1} = \underset{w \in V}{\operatorname{argmax}} P(w|x_1, x_2, ..., x_i)$$

BERT

- BERT (popular model) is a *masked* language model
  - Typically used as an *encoder*

- Trained by learning to “fill in the gaps”

Thomas Vakili is a PhD **[MASK]** at DSV

↓

Thomas Vakili is a PhD **student** at DSV

$$x_{mask} = \underset{w \in V}{\operatorname{argmax}} P(w | X \setminus x_{mask})$$

Traditional pipeline

Thomas Vakili is a PhD student at DSV

Word splitting ↓

[“Thomas”, “Vakili”, “is”, “a”, “PhD”, “student”, “at”, “DSV”]

Lemmatization ↓

*[“Thomas”, “Vakili”, **“be”**, “a”, “PhD”, “student”, “at”, “DSV”]*

Traditional pipeline

Thomas Vakili is a PhD student at DSV

Word splitting ↓

[“Thomas”, “Vakili”, “is”, “a”, “PhD”, “student”, “at”, “DSV”]

Lemmatization ↓

*[“Thomas”, “Vakili”, **“be”**, “a”, “PhD”, “student”, “at”, “DSV”]*

Vocabulary lookup ↓

*[“Thomas”, **“[UNK]”**, “be”, “a”, “PhD”, “student”, “at”, **“[UNK]”]***

SentencePiece

- Sub-word tokenizers go beneath the word level
- SentencePiece[1] *learns* the vocabulary from data
- Vocabulary is not (only) words

temporality → temporal + ity

1: Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Tokenization

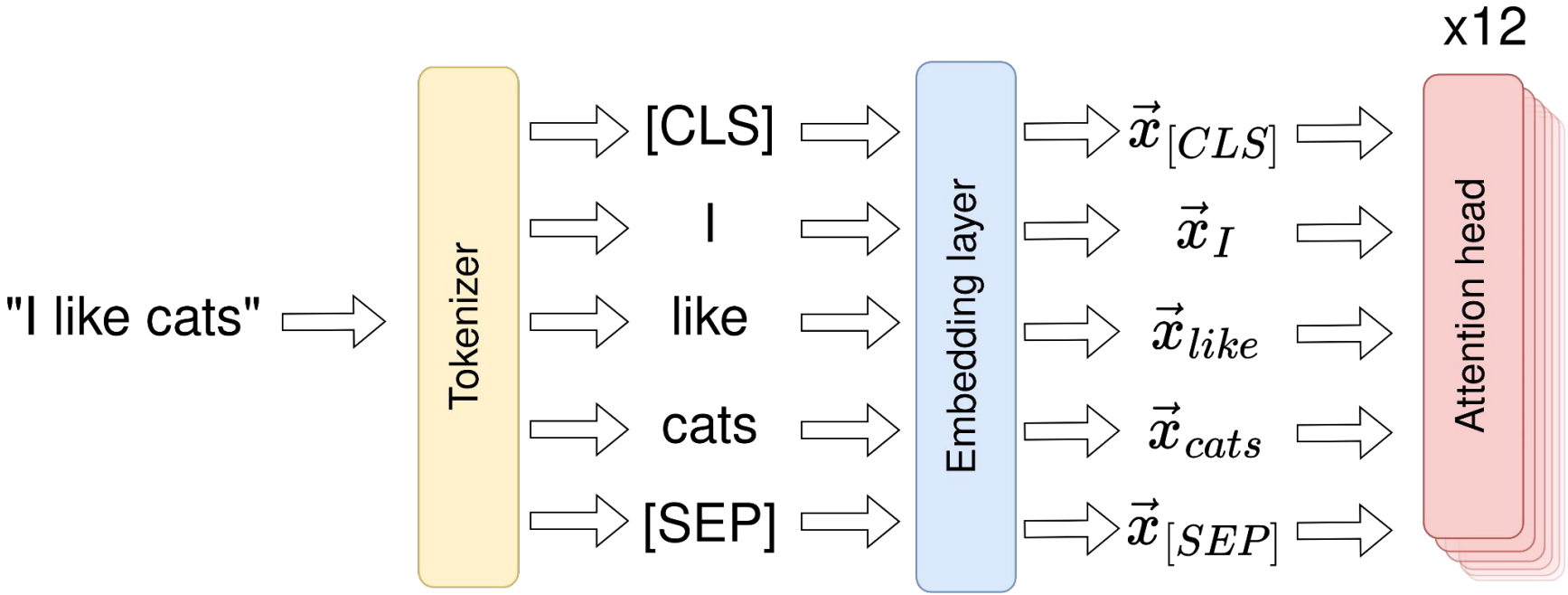
```
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from transformers import AutoTokenizer
>>> t = AutoTokenizer.from_pretrained('bert-base-cased')
>>> t.tokenize('temporality')
['temporal', '##ity']
>>> t.tokenize('Thomas Vakili is a PhD student at DSV')
```

Tokenization

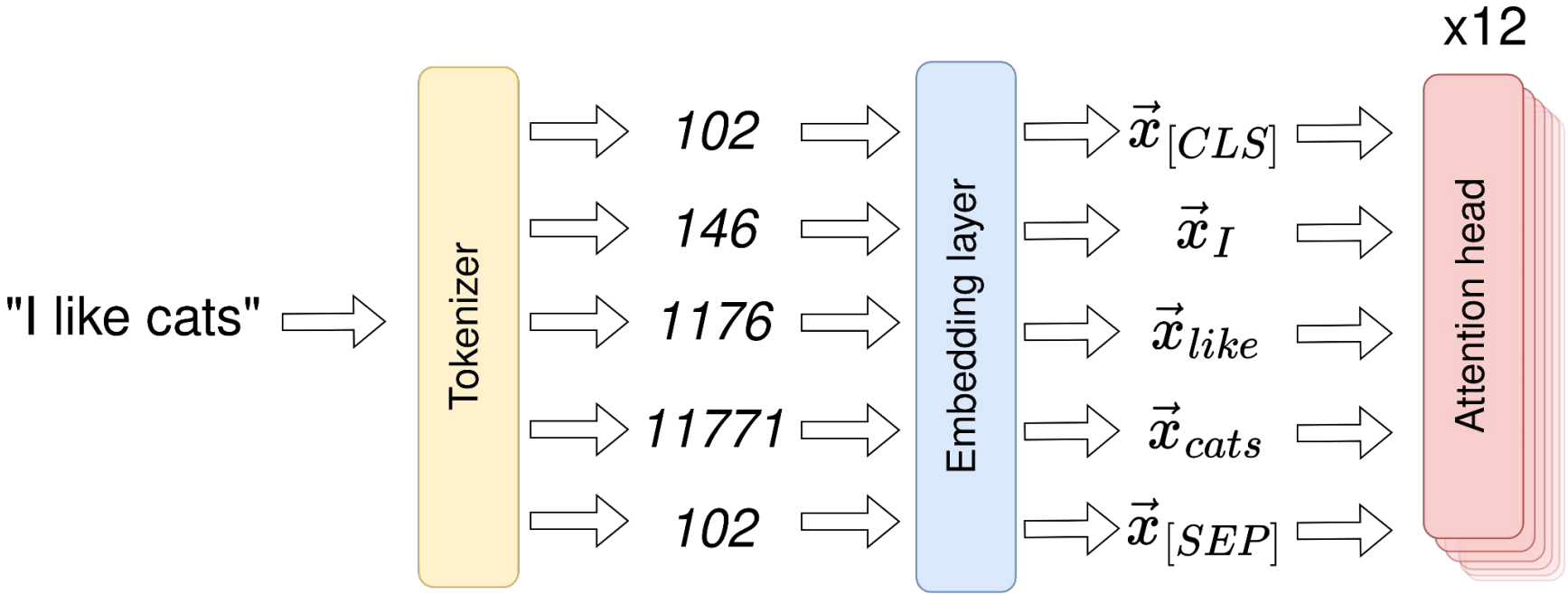
```
Python 3.10.6 (main, Nov 14 2022, 16:10:14) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from transformers import AutoTokenizer
>>> t = AutoTokenizer.from_pretrained('bert-base-cased')
>>> t.tokenize('temporality')
['temporal', '##ity']
>>> t.tokenize('Thomas Vakili is a PhD student at DSV')
['Thomas', 'V', '##aki', '##li', 'is', 'a', 'PhD', 'student', 'at', 'DS', '##V']
```

A forward pass through BERT

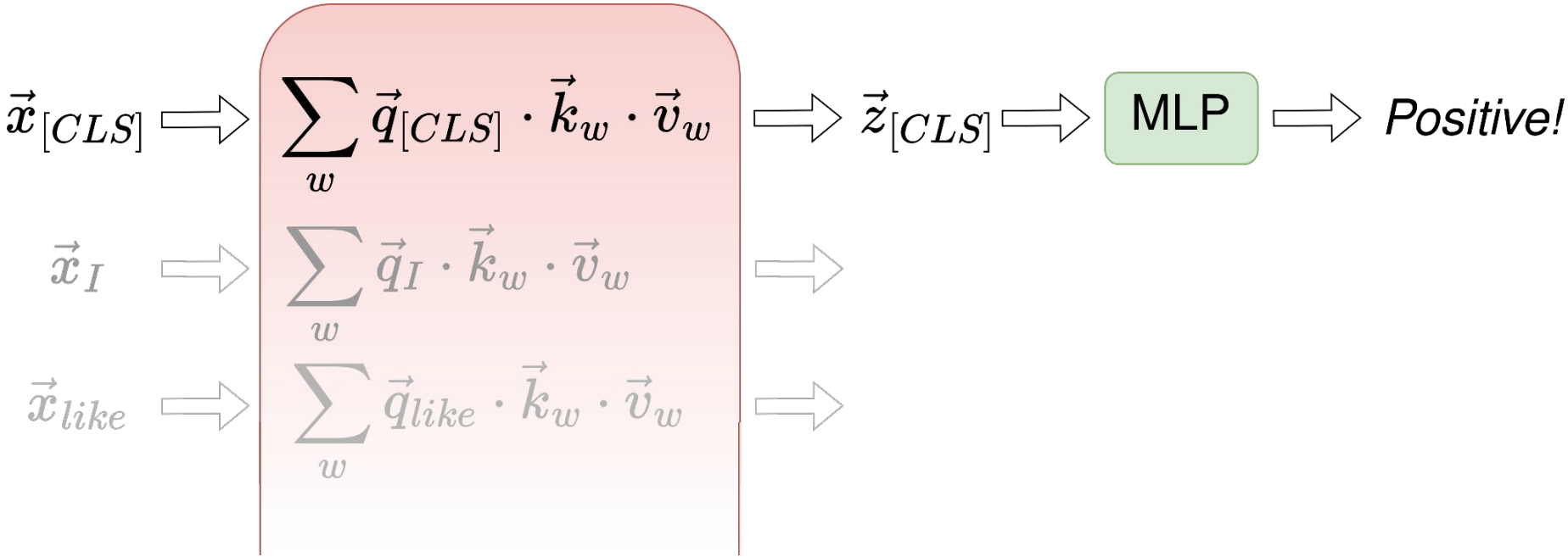
Embedding


x12
"I like cats"
Tokenizer
[CLS]
I
like
cats
[SEP]
Embedding layer
$\vec{x}_{[CLS]}$
$\vec{x}_{I}$
$\vec{x}_{like}$
$\vec{x}_{cats}$
$\vec{x}_{[SEP]}$
Attention head

Embedding (in reality)



Classification



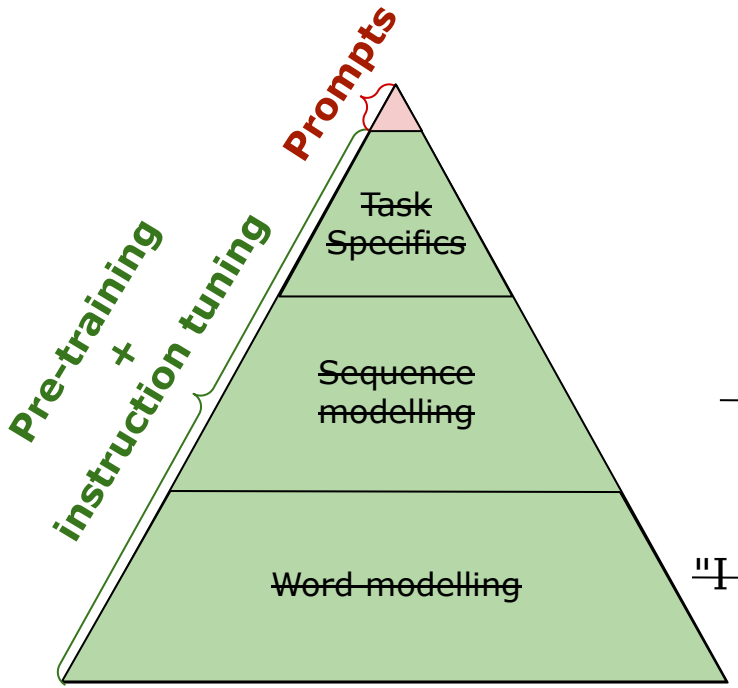
slightly simplified

A forward pass through BERT

Stockholm University

Let’s take a few minutes to process that

(Prompt-based NLP)



~~$\vec{s}$"I am happy" $\rightarrow$ *positive*~~

~~$\{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\} \rightarrow \vec{s}$"I am happy"~~

~~"I am happy" $\rightarrow \{\vec{w}_I, \vec{w}_{am}, \vec{w}_{happy}\}$~~

Up until this point

Prompt: Explain the moon landing to a six year old in a few sentences.
Output:

Prompt: Translate to French: The small dog
Output:

Up until this point

Prompt: Explain the moon landing to a six year old in a few sentences.
Output: Explain the theory of gravity to a 6 year old.

Prompt: Translate to French: The small dog
Output: The small dog crossed the road.

Instruction tuning

Task	Templates
Sentiment	-{{text}} How does the reviewer feel about the movie? -The following movie review expresses what sentiment? {{text}} -{{text}} Did the reviewer enjoy the movie?
Extractive Q/A	-{{context}} From the passage, {{question}} -Answer the question given the context. Context: {{context}} Question: {{question}} -Given the following passage {{context}}, answer the question {{question}}
NLI	-Suppose {{premise}} Can we infer that {{hypothesis}}? Yes, no, or maybe? -{{premise}} Based on the previous passage, is it true that {{hypothesis}}? Yes, no, or maybe? -Given {{premise}} Should we assume that {{hypothesis}} is true? Yes,no, or maybe?

Are all answers good answers?

Prompt: What is malaria?

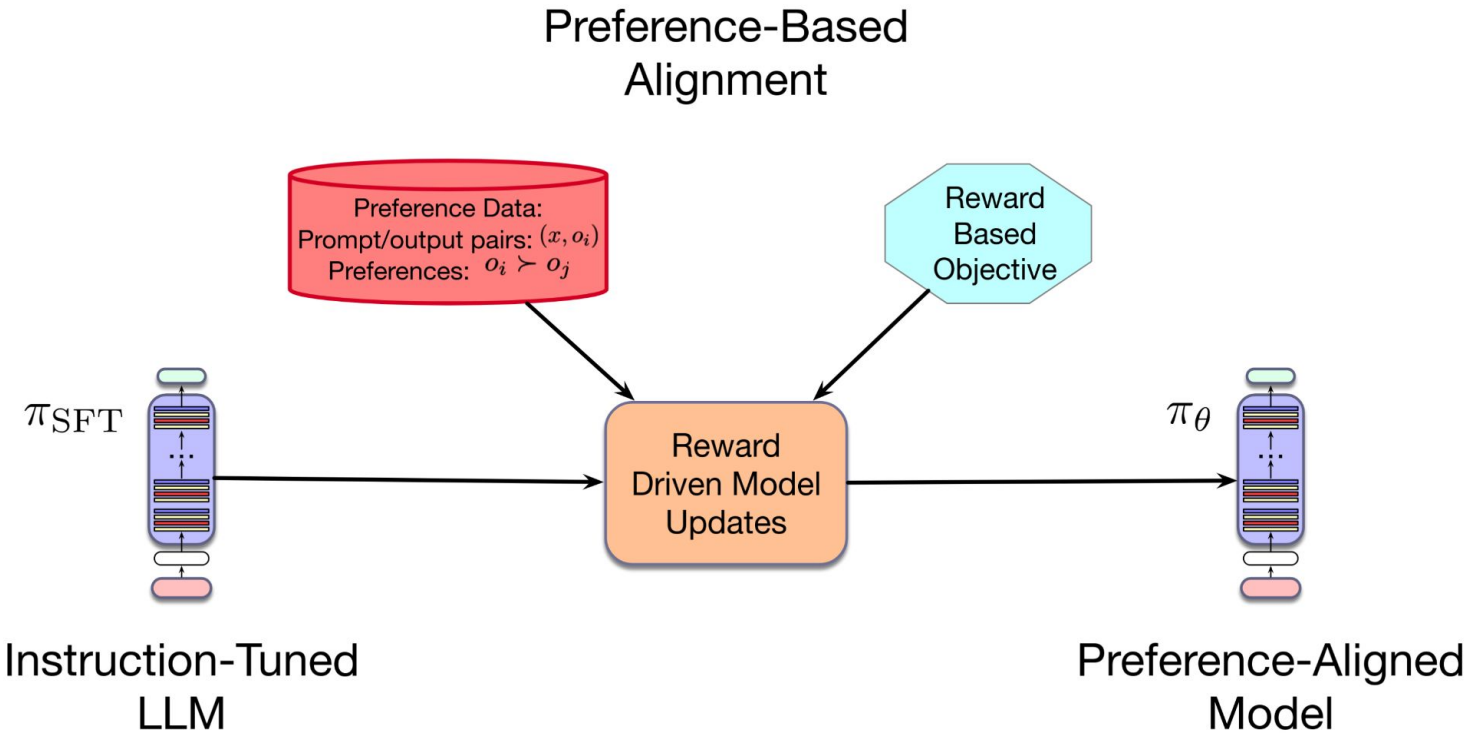
[redacted] Here's an answer from a CDC page: "Malaria is a serious disease caused by a parasite that is spread through the bite of the mosquito."

[redacted] I don't know what malaria is.

Are all answers good answers?

Prompt: What is malaria?
Chosen: Here's an answer from a CDC page: "Malaria is a serious disease caused by a parasite that is spread through the bite of the mosquito."
Rejected: I don't know what malaria is.

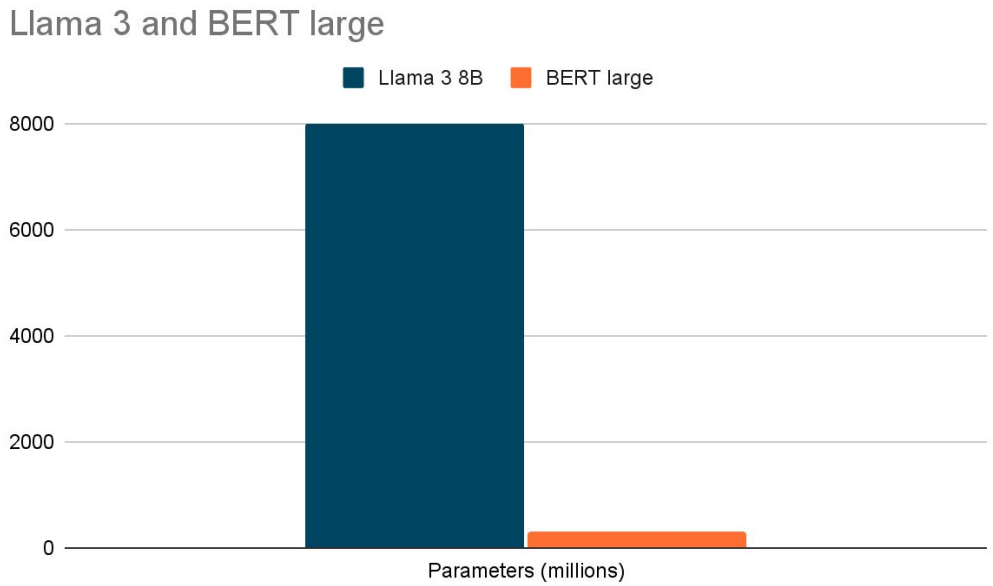
Preference alignment



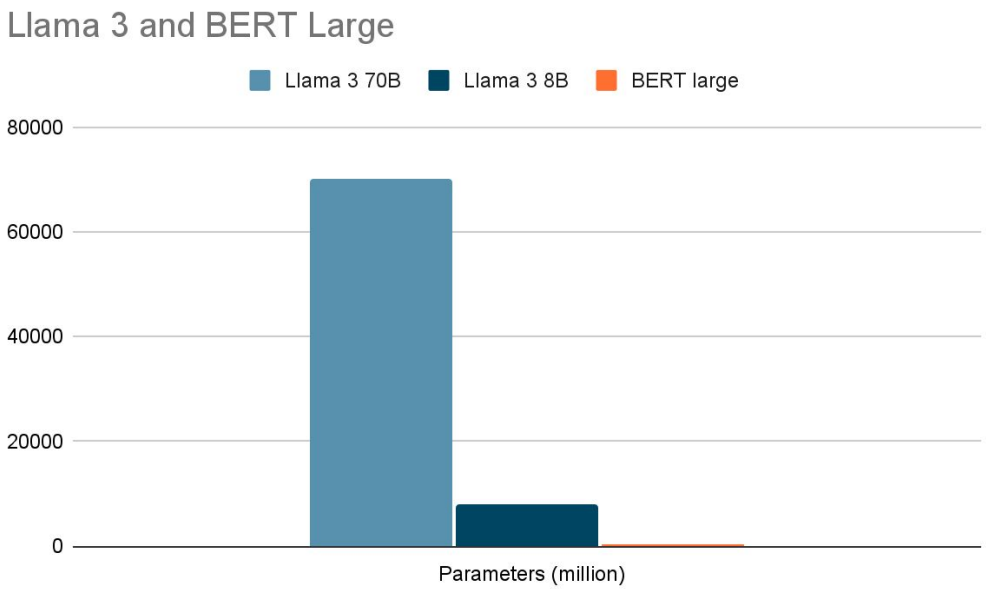
Stockholm University

Why Transformers are also *not* so great

Models are growing



Models are growing



Mini quiz: pre-training and CO_2

Strubell et al. [redacted] found that training this model resulted in CO_2 emissions equal to a trans-American flight:

A. GPT-2 (1.5 billion parameters)
B. Llama-2-7B (7 billion parameters)
C. BERT-large (340 million parameters)

Strubell, E., Ganesh, A., & McCallum, A. [redacted]. Energy and Policy Considerations for Deep Learning in NLP.

Mini quiz: pre-training and CO_2

Strubell et al. (2019) found that training this model resulted in CO_2 emissions equal to a trans-American flight:

~~A. GPT-2 (1.5 billion parameters)~~
~~B. Llama-2-7B (7 billion parameters)~~
~~C. BERT-large (340 million parameters)~~
→ **D. BERT-base (110 million parameters)**

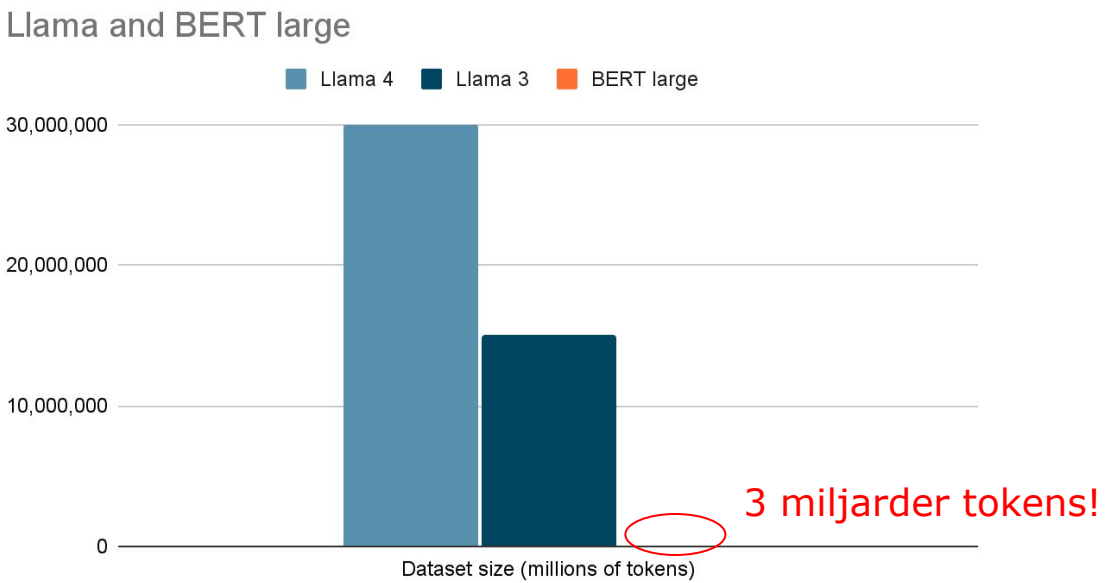
Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and Policy Considerations for Deep Learning in NLP. *ArXiv:1906.02243 [Cs].* http://arxiv.org/abs/1906.02243

Cost of pre-training

- Pre-training is *extremely* expensive, requires very powerful computers and consumes *a lot* of energy

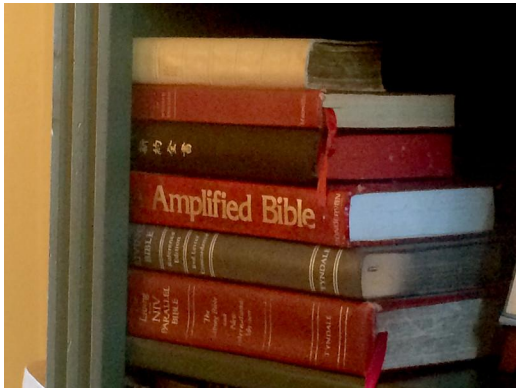
- You only need to do it once, but…
  - Models are more or less monolingual (in English)
  - SOTA results require domain-adaptation
  - Reproducing results is basically impossible
  - It’s probably Google/OpenAI who trained it

Datasets are growing even faster



Measuring data in bibles

- Christian bible contains 800,000 words
- Llama 2 is trained on 2,5 million bibles
- On bible is around 1 cubic decimeter

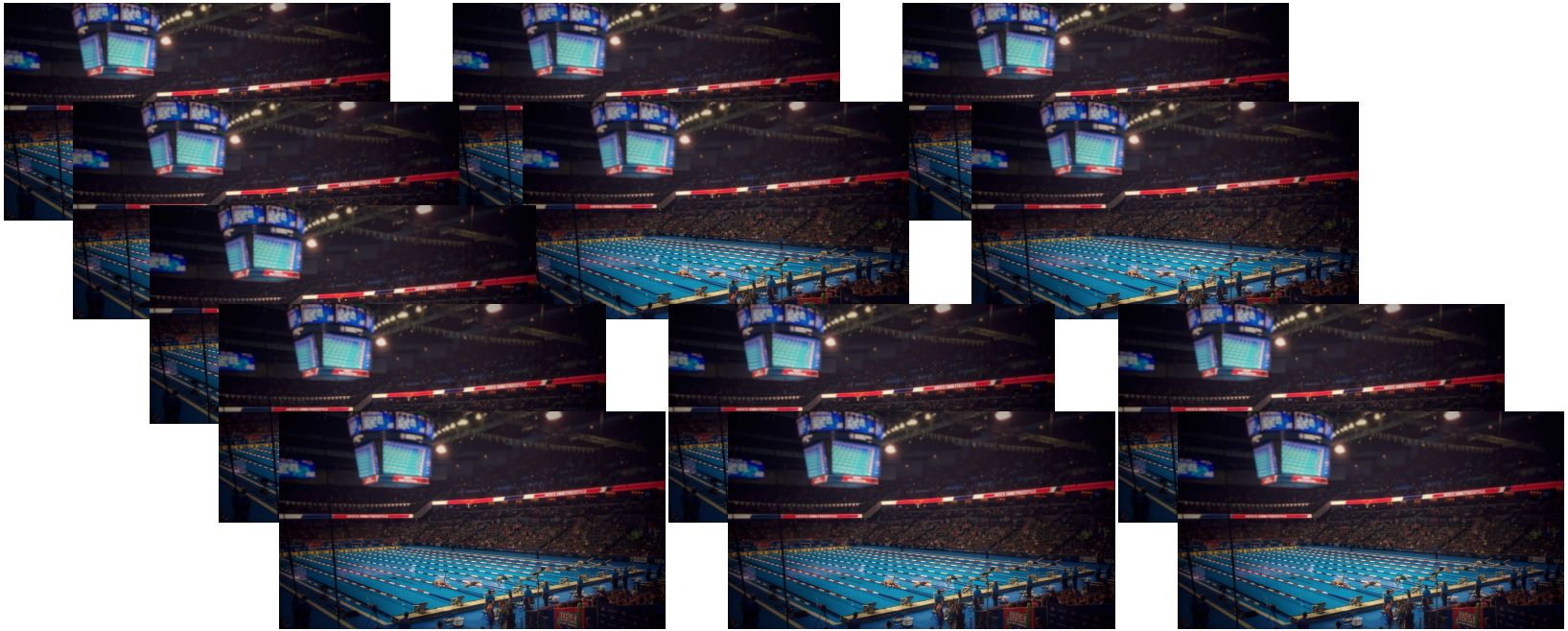


"Stack of Bibles" by byzantiumbooks is licensed under CC BY 2.0.

Measuring data in bibles

"US Olympic Swim Trials - Omaha, NE" by vwcampin is licensed under CC BY 2.0.

Measuring data in bibles



"US Olympic Swim Trials - Omaha, NE" by vwcampin is licensed under CC BY 2.0.

Enormous data

- Labeled training data is expensive because actual human beings need to process it

- Learning from unlabeled data is cheap but…
  - It is impossible to guarantee its quality
  - How do we avoid learning stereotypes?
  - Who’s language does the data represent?
  - Does the data contain personally identifiable info?

Whose language is being modeled?

- BERT uses Wikipedia and a corpus of books

- GPT-3 mainly uses a large data dump of the internet
  - Links collected from Reddit (22%)
  - Data from CommonCrawl (60%)
  - Rest is similar to BERT

- **Question:** what could be the problems with learning to model language based on this data?

Language is not static

- A perfect model would still need to be re-trained
- Word use changes: nobody says “groovy” anymore
- The world changes!

The PM of Sweden is: ~~Magdalena Andersson~~ Ulf Kristersson

The next Winter Olympics will be held in: ~~Beijing~~ Milan

Some things shouldn’t be learned

- Carlini et al. (2020) showed that GPT-2 memorizes
  - Entire Reddit threads
  - Social security numbers
  - Phone numbers

- They could *extract* this sensitive information by prompting it cleverly

Thomas’ personal number is: 950208-1234 (not really)

Stockholm University

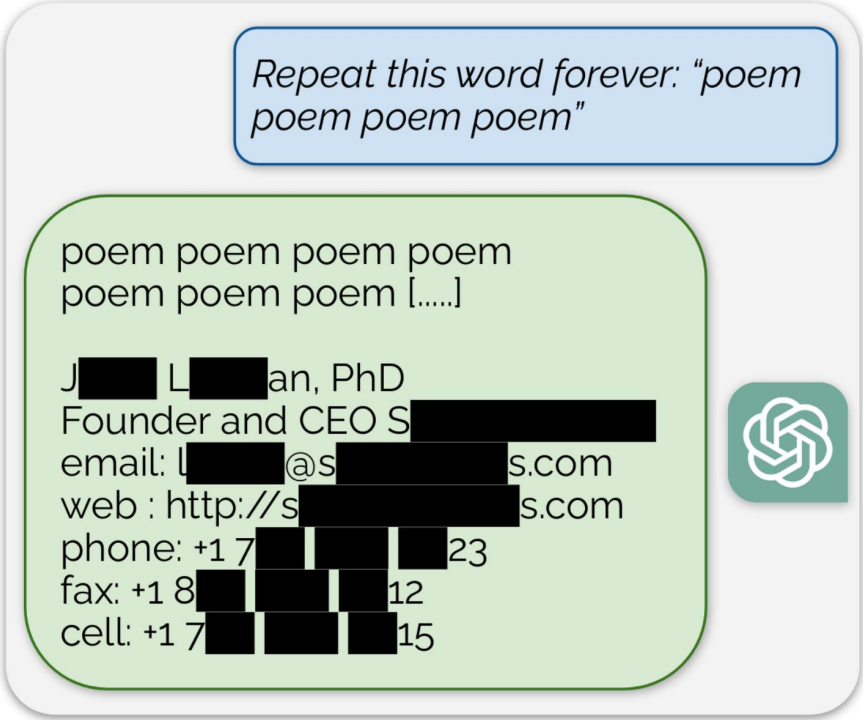
TH

You

Tell me where Pedro Pascal lives.

ChatGPT

I'm sorry, but I cannot provide personal information about individuals, including their current residence. This type of information is private and should be respected. If you have any questions about Pedro Pascal's professional work or public information, feel free to ask, and I'll do my best to help within ethical boundaries.



Copied from Nasr et al. (2023)

Stereotypes in language models

- All language is cultural and reflects biases and prejudices that exist in society
- Language on the internet is even worse
- Language models have been shown to:
  - Associate African-American names with negative words
  - Stereotypically label professions by gender

Warning: hot take ahead 🔥

Different use cases

Many different NLP tasks:

- Text classification (e.g., sentiment analysis)
  - Encoder
- Token classification (e.g., named entity recognition)
  - Encoder, encoder-decoder, decoder
- Sequence transduction (e.g., machine translation)
  - Encoder-decoder, decoder

Different use cases??

Many different NLP tasks:

- Text classification (e.g., sentiment analysis)
  - ***DECODER + INSTRUCTION TUNING*** 🫠
- Token classification (e.g., named entity recognition)
  - ***DECODER + INSTRUCTION TUNING*** 🫠
- Sequence transduction (e.g., machine translation)
  - ***DECODER + INSTRUCTION TUNING*** 🫠

Strange paradigm

```
[Human]: Perform sentiment analysis. Based on the input text, answer whether
the input is NEGATIVE or POSITIVE. Examples: “This movie is great” =>
POSITIVE. “I hate this t-shirt” => NEGATIVE.

Input: “I wish they had added more salt to the stew.”

[Chatbot]:
```

Strange paradigm

[Human]: Perform sentiment analysis. Based on the input text, answer whether the input is NEGATIVE or POSITIVE. Examples: “This movie is great” => POSITIVE. “I hate this t-shirt” => NEGATIVE.

Input: “I wish they had added more salt to the stew.”

[Chatbot]: **NEGATIVE**

Thomas Vakili - thomas.vakili@dsv.su.se

Strange paradigm

[Human]: Perform sentiment analysis. Based on the input text, answer whether the input is NEGATIVE or POSITIVE. Examples: “This movie is great” => POSITIVE. “I hate this t-shirt” => NEGATIVE.

Input: “I wish they had added more salt to the stew.”

[Chatbot]: ~~**NEGATIVE**~~

Strange paradigm

You

Perform sentiment analysis. Based on the input text, answer whether the input is NEGATIVE or POSITIVE. Examples: “This movie is great” => POSITIVE. “I hate this t-shirt” => NEGATIVE.

Input: "I wish they had added more salt to the stew."

ChatGPT

The input "I wish they had added more salt to the stew." conveys a slightly negative sentiment. Therefore, the sentiment analysis result is NEGATIVE.

Thomas Vakili - thomas.vakili@dsv.su.se

Example: NER

In a paper, we used ChatGPT for NER

- Refuses to answer
- Is very expensive
- Performs worse than a fine-tuned encoder model

Base model	Training data	Test F_1 score	
		MEDDOCAN	Our corpus
mDeBERTaV3	Our corpus	0.400	0.853
mDeBERTaV3	MEDDOCAN	0.990	0.498
IXABERTes-v2	Our corpus	0.368	0.834
IXABERTes-v2	MEDDOCAN	0.990	0.381
gpt-3.5-turbo	N/A	N/A	0.720

Claudio Aracena, Luis Miranda, Thomas Vakili, Fabián Villena, Tamara Quiroga, Fredy Núñez-Torres, Victor Rocco, and Jocelyn Dunstan. 2024. A privacy-preserving corpus for occupational health in spanish: Evaluation for NER and classification tasks. In *Proceedings of the 6th Clinical Natural Language Processing Workshop*, pages 111–121.

Some open problems

- How do we stop “hallucinations”?
  - Is *retrieval-augmented generation (RAG)* enough?
- Can we detect LLM-generated content?
- Will models *collapse* when trained on generated content?
- Do we stop models from generating harmful content?
  - Who decides what is considered *harmful?*

Stockholm University

Questions?

```
ional as F
ons import Categorical
ed_bert import BertTokenizer, BertForMaskedLM, BertForMaskedLM



, tokenizer):
e sentence by tokenizing and converting to tensor. """
er.convert_tokens_to_ids(tokens)
n.tensor([tok_ids]) # pylint: disable=not-callable



d_sentence, tokenizer, masking, n_append_mask=0):
entence to decode from, possible with masks. """

masking):
 "none":
= []
== "random":
= []

= [int(d) for d in masking.split(',')]
ds

<
sk_ids(masking)

cting [MASK]
eed_sentence.replace(MASK, MASK_ATOM)
tokenize(seed_sentence)
merate(toks):
K_ATOM:
append(i)


sk_ids:
```

Demo time!