Data Mining and Language Technology

08 - Transformers

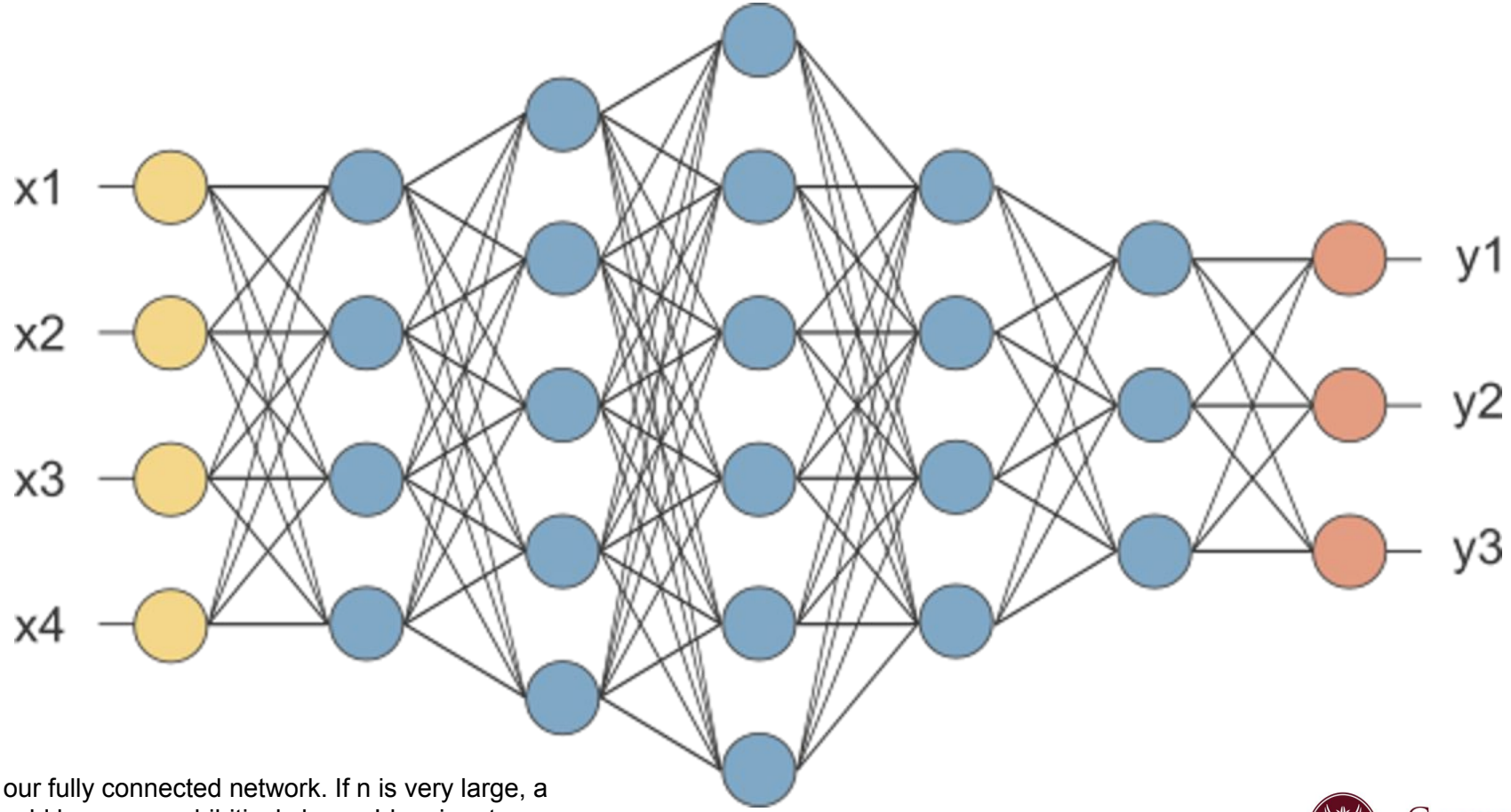
SAPIENZA
UNIVERSITÀ DI ROMA

Fabrizio Silvestri

Modelling Sequence Data

SAPIENZA
UNIVERSITÀ DI ROMA

Examples of Sequence data	Input Data	Output
Speech Recognition		*This is RNN*
Machine Translation	Hello, I am Pankaj.	Hallo, ich bin Pankaj. हैलो, मैं पंकज हूं।
Language Modeling	Recurrent neural ? based ? model	**network** **language**
Named Entity Recognition	Pankaj lives in Munich	**Pankaj** lives in **Munich** *person* *location*
Sentiment Classification	There is nothing to like in this movie.	★☆☆☆☆
Video Activity Analysis		Punching



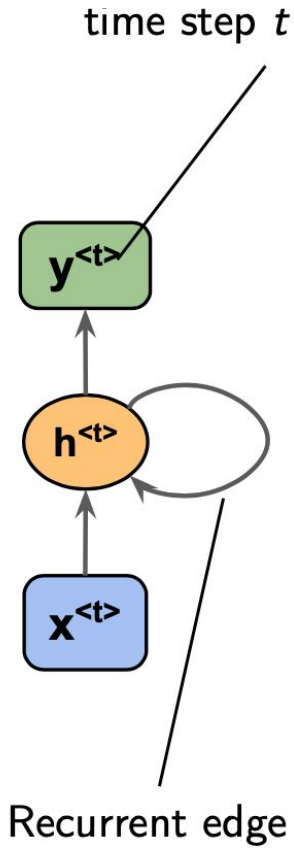
This is our fully connected network. If n is very large, a FCN would become prohibitively large. Idea: input one x_i at a time, and re-use the same edge weights

Recurrent Neural Networks

SAPIENZA
UNIVERSITÀ DI ROMA

The General Idea

- A recurrent neural network is a neural network that is specialized for processing a sequence of values $\mathbf{x}_1, \ldots, \mathbf{x}_T$.
- Parameter sharing enables the model for examples of different lengths and generalize across them.



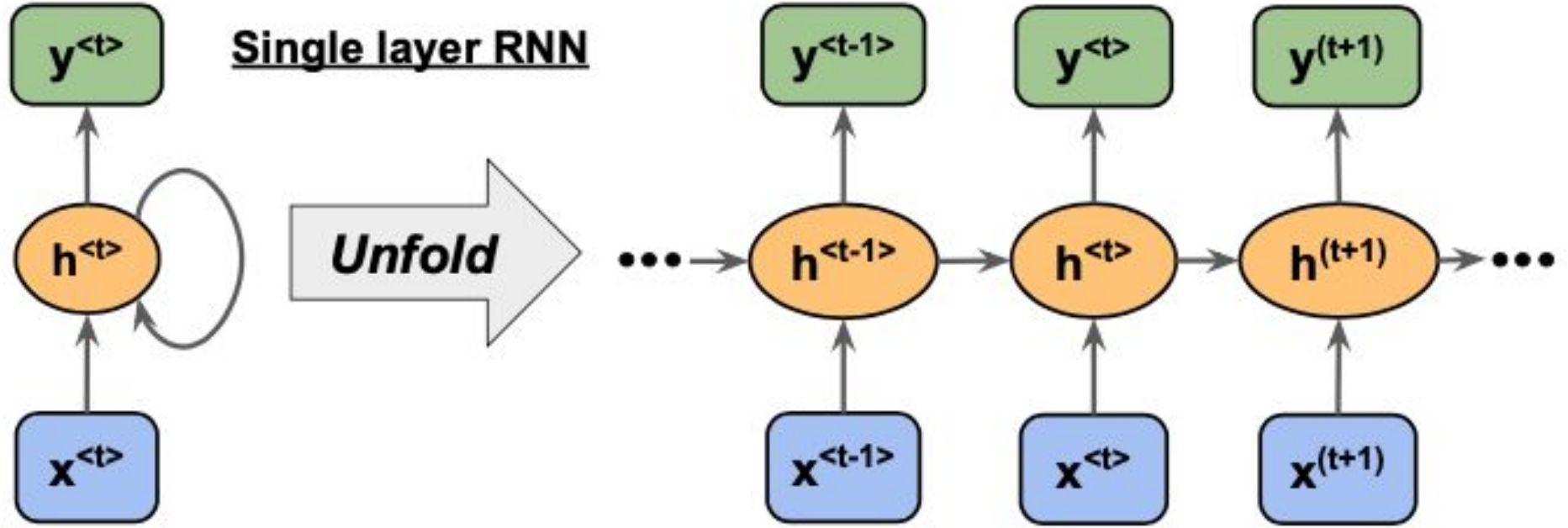
Unfolding

- Consider the classical form of a dynamical system:

$$\underbrace{h^{<t>}}_{\text{cell state}} = f_W(\underbrace{h^{<t-1>}}_{\text{old state}}, \underbrace{x^{<t>}}_{\text{input vector}})$$

  - recurrent because the definition of s at time t refers back to the same definition at time t−1.
- For a finite number of time steps τ, the formula can be unfolded by applying the definition $\tau-1$ times

Unfolding



Weight Matrices

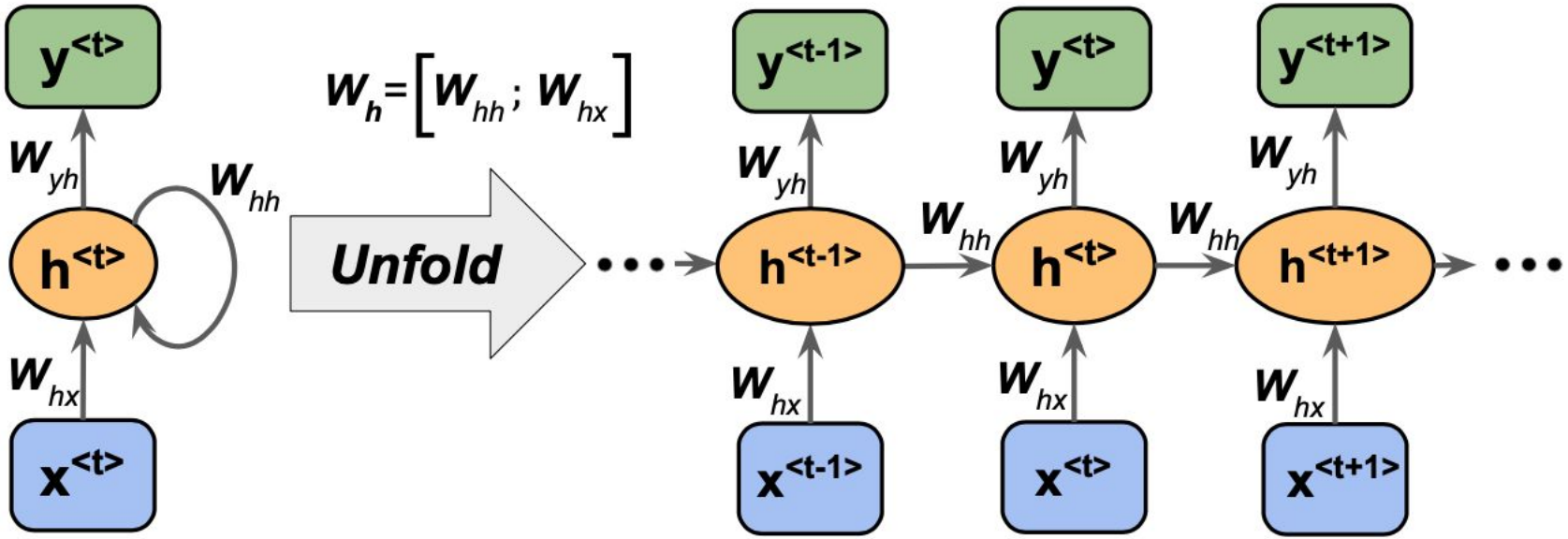


Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition.* Birmingham, UK: Packt Publishing, 2019

Vanilla RNN

- A possible instantiation of:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- Is the following

$$h_t = g_\theta(h_{t-1}, x_t)$$

$$\downarrow$$

$$h_t = \tanh(W_{hh}\, h_{t-1} + W_{xh} x_t)$$

$$o_t = softmax(W_{ho} h_t)$$

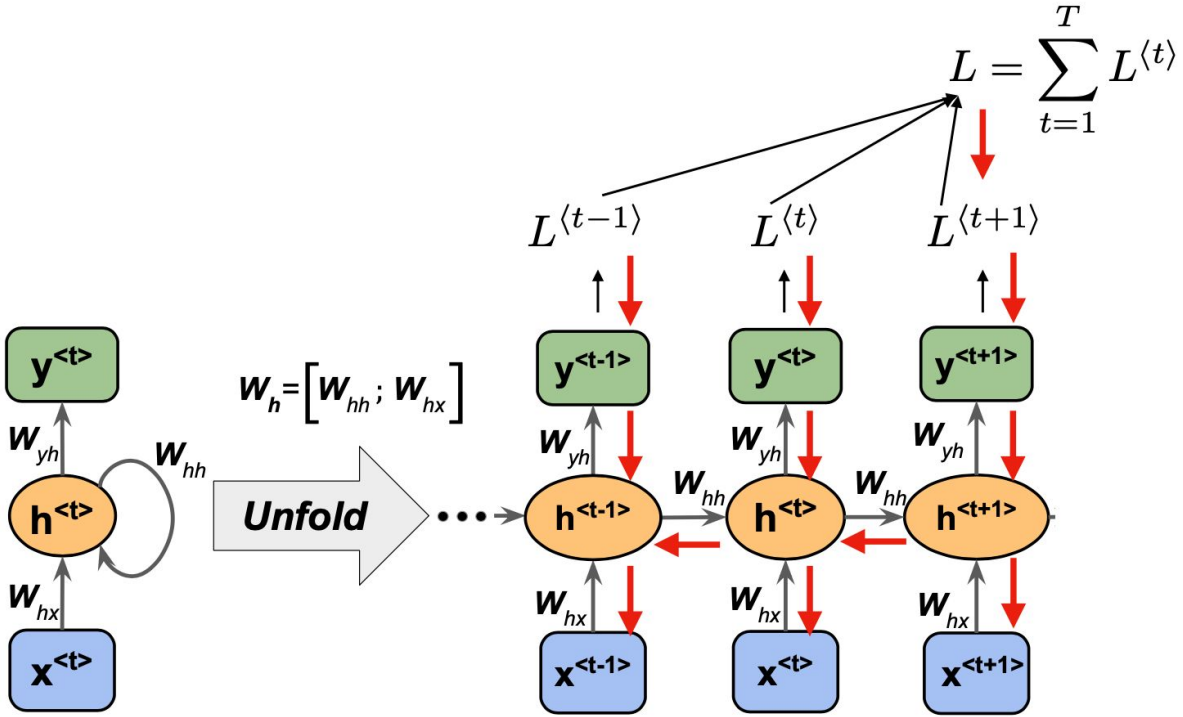
Backpropagation Through Time

How to Train a Recurrent NN?

- The training data for a recurrent neural network is an ordered sequence of k input-output pairs, $\langle x_0,y_0\rangle$, $\langle x_1,y_2\rangle$, …, $\langle x_{k-1},y_{k-1}\rangle$.
  - An initial value must be specified for the hidden state h_0.
  - Typically, a vector of all zeros is used for this purpose.
- We begin by unfolding a recurrent neural network in time.
  - Every copy of the network shares the same parameters.
- We can now use Backprop taking extra care of the shared parameters
  - Backpropagation Through Time (BPTT).

Inspired by: https://deeplearningmath.org/sequence-models.html

Backpropagation Through Time



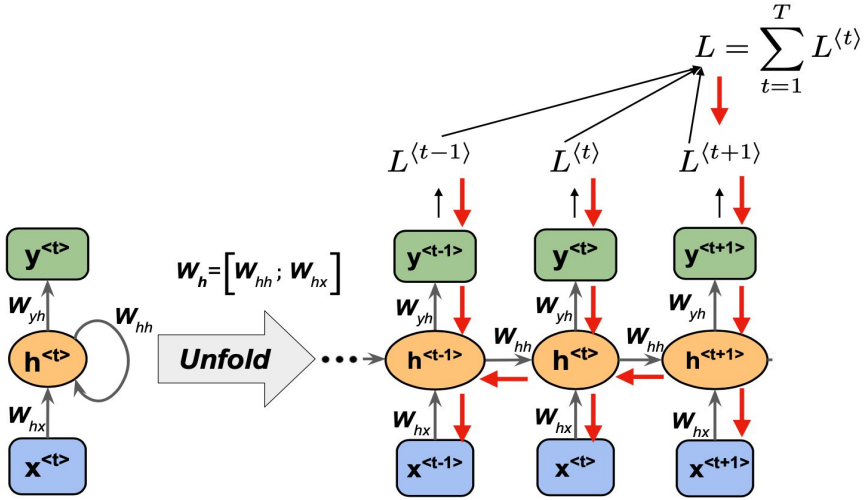
SAPIENZA UNIVERSITÀ DI ROMA

Long-Short Term Memory

SAPIENZA
UNIVERSITÀ DI ROMA

Motivations

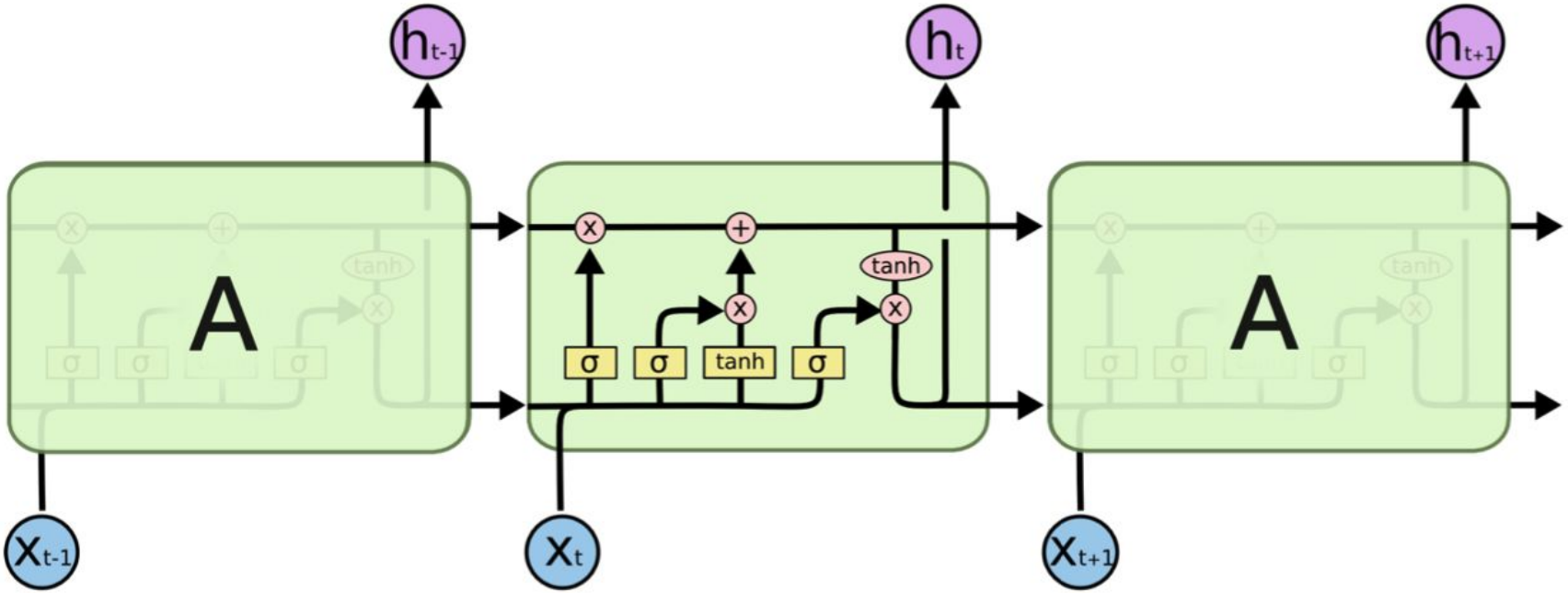
- When backpropagating gradients, the parameters referring to far away in time are (way) less affected by the loss.



Main ideas

- Instead of using just a single hidden state ($\mathbf{h}^{(t)}$) Long-Short Time Memory architectures use also a cell state ($\mathbf{c}^{(t)}$)
- The cell has the ability to store long-term information.
- LSTM models can erase, write and read information from the cell.
- Gates are defined to get the ability to select which information to either erase, write or read.
    - The gates are also vectors of length n.
    - At each step *t* the gates can be open (1), closed (0) or somewhere in-between. Note that the gates are dynamic.

Graphical Representation of an LSTM Module

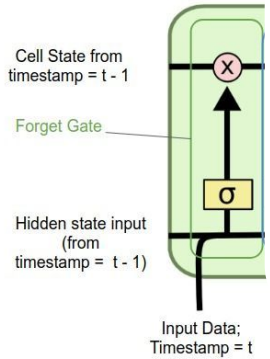


From a sequence of inputs $\mathbf{x}^{(t)}$, LSTM computes a sequence of hidden states $\mathbf{h}^{(t)}$, and cell state $\mathbf{c}^{(t)}$

Forget Gate

- Forget gate: controls what is kept versus forgotten, from previous cell state

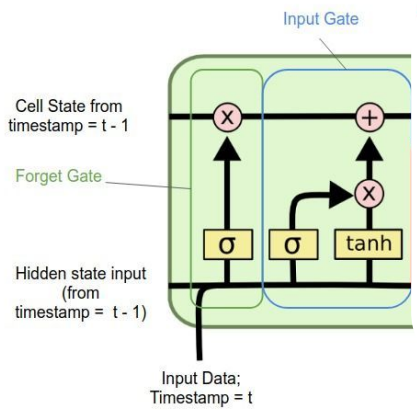
$$f^{(t)} = \sigma\left(W_f h^{(t-1)} + U_f x^{(t)} + b_f\right)$$



Input Gate

- Input gate: controls what parts of the new cell content are written to cell

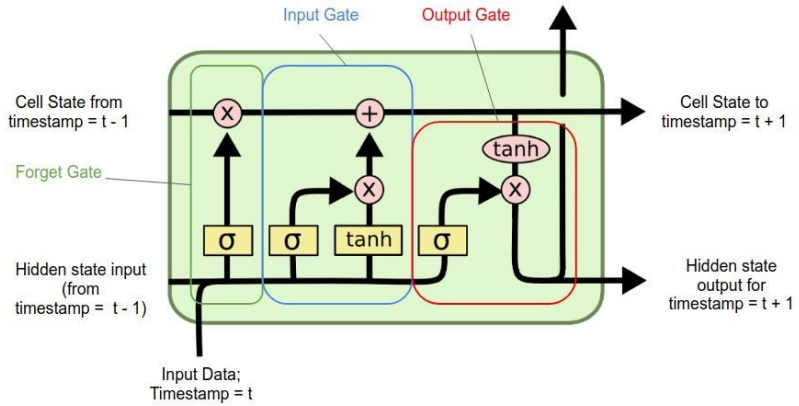
$$i^{(t)} = \sigma\left(W_i h^{(t-1)} + U_i x^{(t)} + b_i\right)$$



Output Gate

- Output gate: controls what parts of cell are transferred to hidden state

$$o^{(t)} = \sigma\left(W_o(r^t \circ h^{(t-1)}) + U_o x^{(t)} + b_o\right)$$



New Cell Content

- After hidden and cell have been used we need to update the cell and then the hidden state
- New content

$$\tilde{c}^{(t)} = \tanh\left(W_c h^{(t-1)} + U_c(x^t + b_c\right)$$

- erase (“forget”) some content from last cell state, and write (“input”) some new cell content

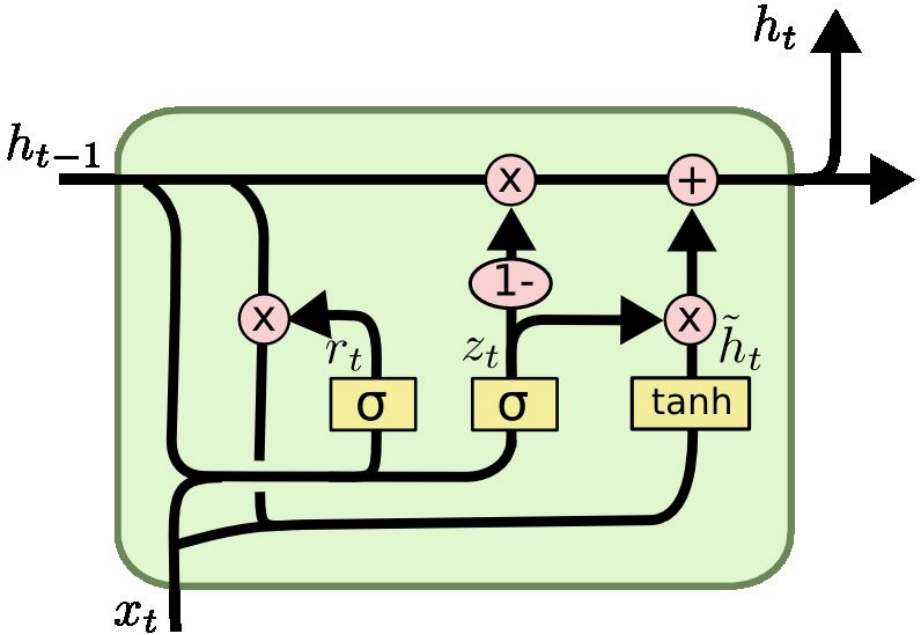
$$c^{(t)} = f^{(t)} \circ c^{(t-1)} + i^{(t)} \circ \tilde{c}^{(t)}$$

- read (“output”) some content from the cell

$$h^{(t)} = o^{(t)} \circ \tanh c^{(t)}$$

Gated Recurrent Units

Graphical Representation of a GRU Module

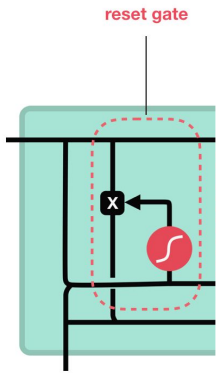


From a sequence of inputs $\mathbf{x}^{(t)}$, GRU does not use any cell state, and at each timestep t only a hidden state $\mathbf{h}^{(t)}$ is maintained.

Reset Gate

- Reset gate: controls what parts of previous hidden state are used to compute new content

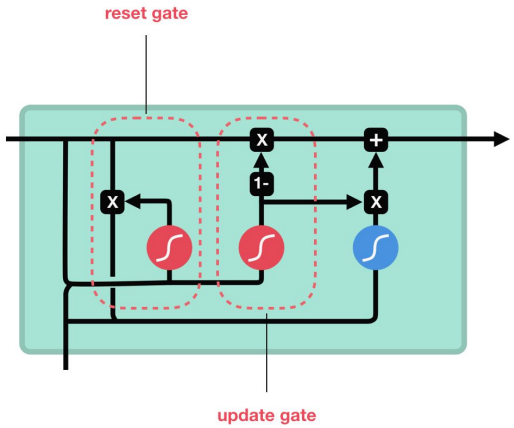
$$r^{(t)} = \sigma\left(W_r h^{(t-1)} + U_r x^{(t)} + b_r\right)$$



Update Gate

- Update gate: controls what parts of hidden state are updated vs preserved

$$u^{(t)} = \sigma\left(W_u h^{(t-1)} + U_u x^{(t)} + b_u\right)$$



Updates to Hidden States

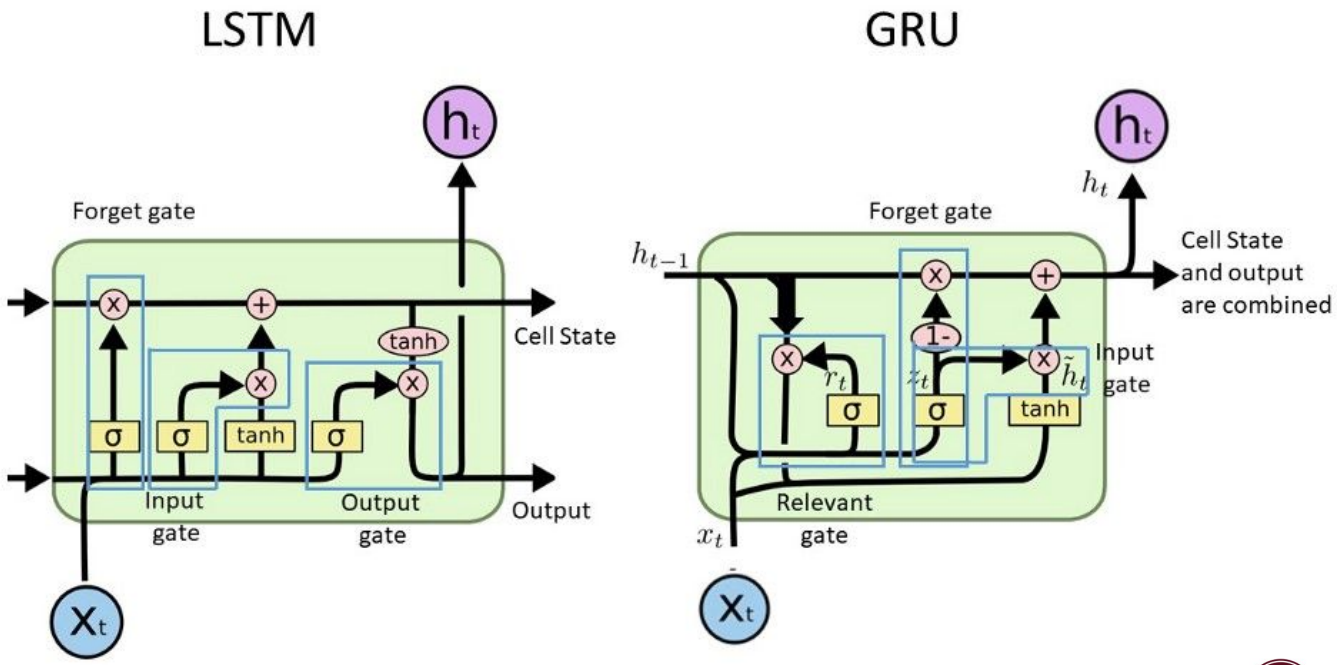
- New hidden state content: reset gate selects useful parts of previous hidden state. Use this and current input to compute new hidden content.

$$\tilde{h}^{(t)} = \tanh\left(W_r(r^t \circ h^{(t-1)}) + U_h x^{(t)} + b_h\right)$$

- Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$h^{(t)} = (1 - u^{(t)}) \circ h^{(t-1)} + u^{(t)} \circ \tilde{h}^{(t)}$$

LSTM vs. GRU

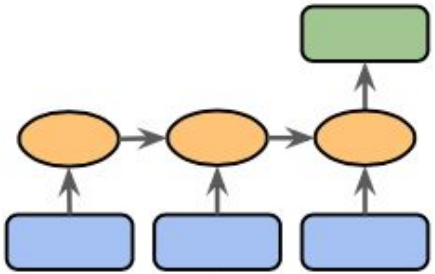


Sequence-to-Sequence (Seq2Seq)

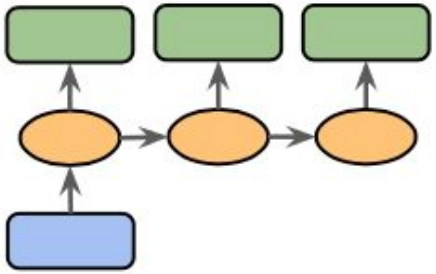
The General Idea

- RNNs are at the basis of a general idea of producing ML models that learn to transform an input sequence into an output sequence
- Examples:
  - Translation: I am Italian → Io sono italiano
  - Captioning:  → Picture of a flying plane
  - Aging:  → 

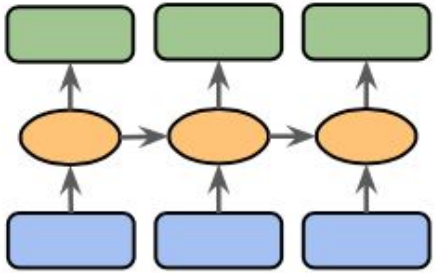
Types of Seq2Seq



many-to-one



one-to-many



many-to-many

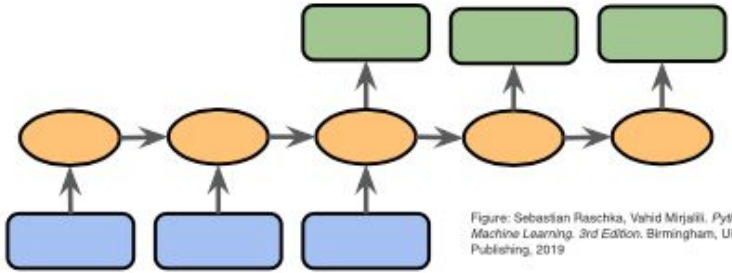
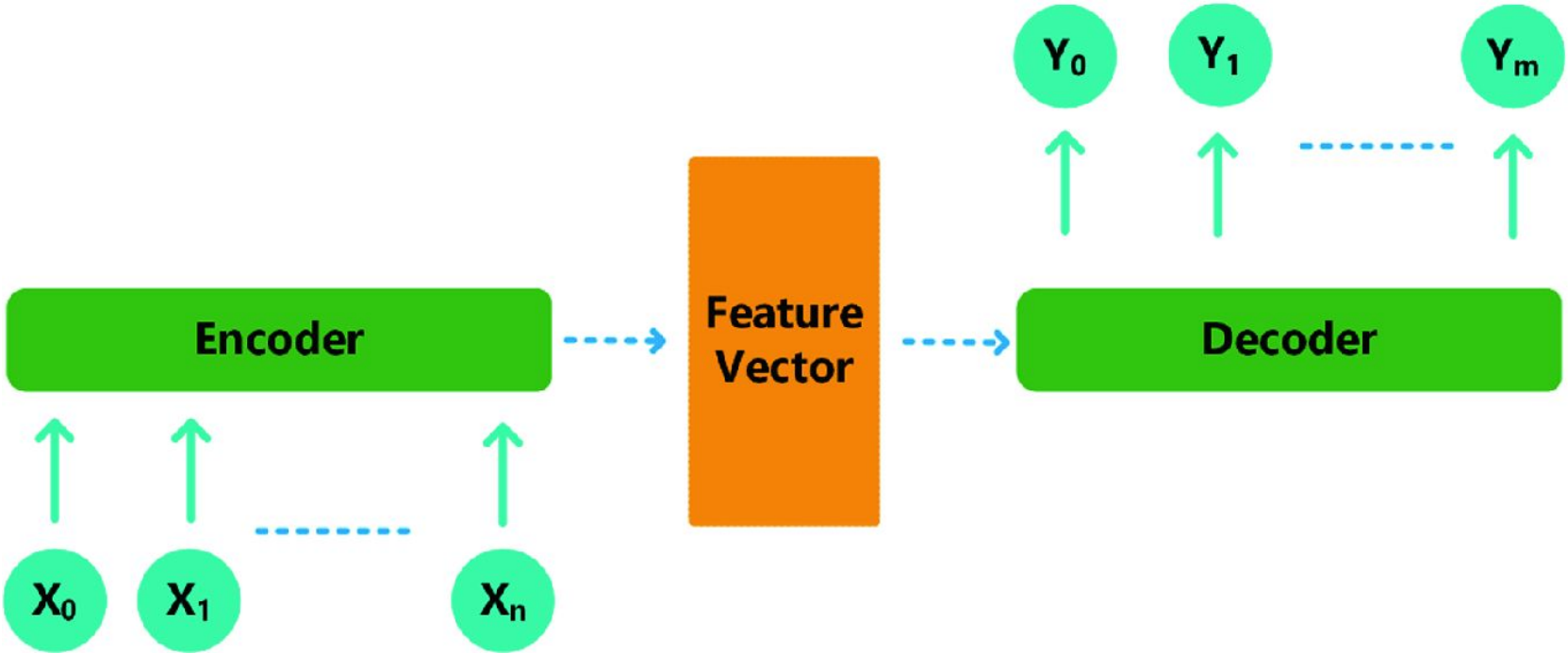


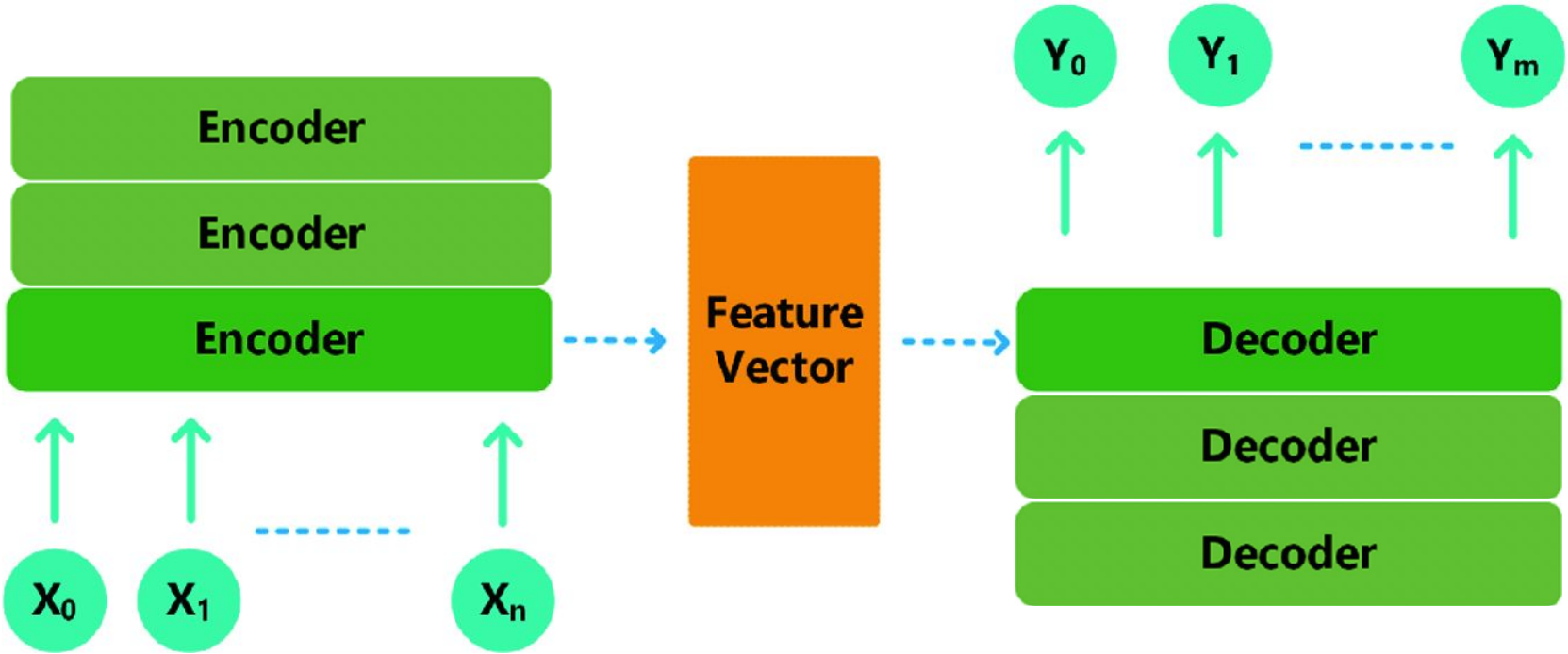
Figure: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning. 3rd Edition*. Birmingham, UK: Packt Publishing, 2019

many-to-many

The Encoder-Decoder Paradigm



The Encoder-Decoder Paradigm (DL Version)

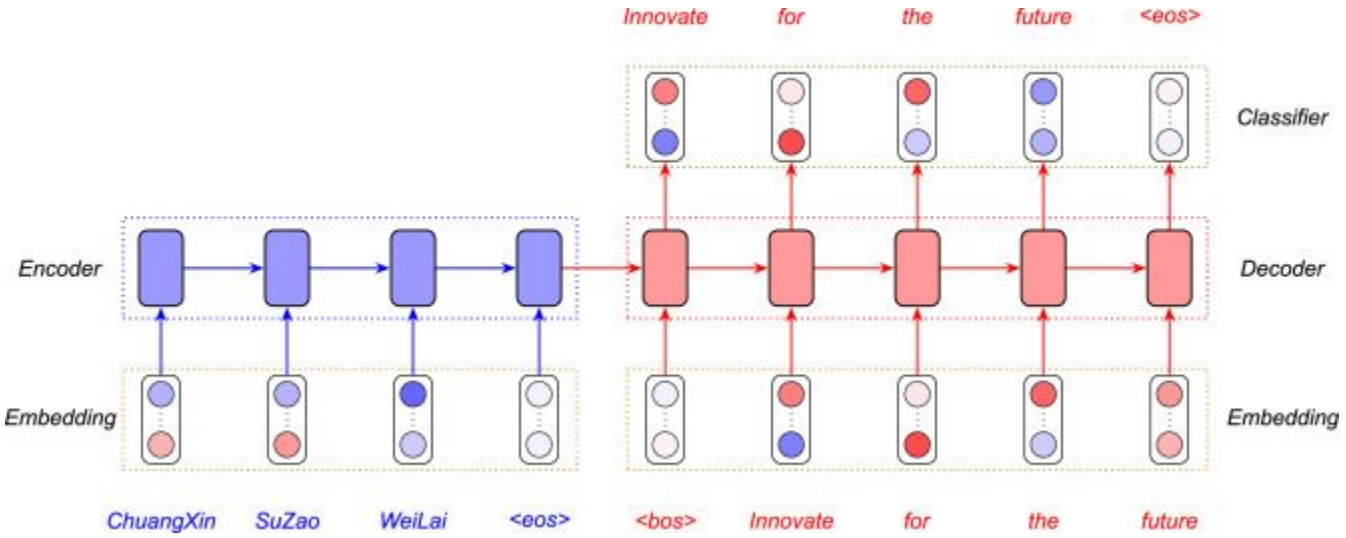


Autoregressive Models

What Happens after the Encoding Stage?

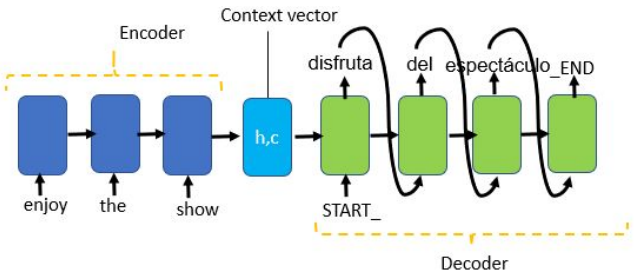
- The Encoder generates a representation for the input sequence.
- The decoder takes this representation and generates the output sequence.

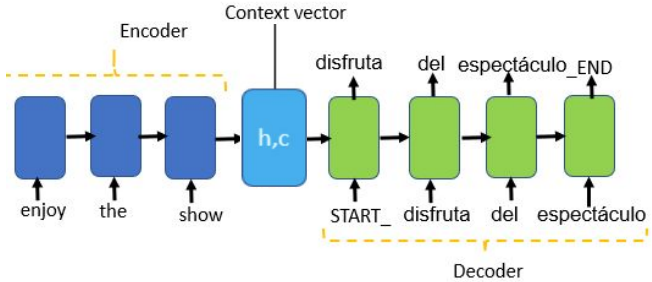
The Example of Language Translation



Teacher Forcing

- At decoding stage, the only input we have is that we generated so far…
  - … not at training time.
- During training, with probability p feed in input the ground truth item of the sequence (with prob. 1-p the generated one)





Teacher forcing -Actual target words are sent as an input to the decoder

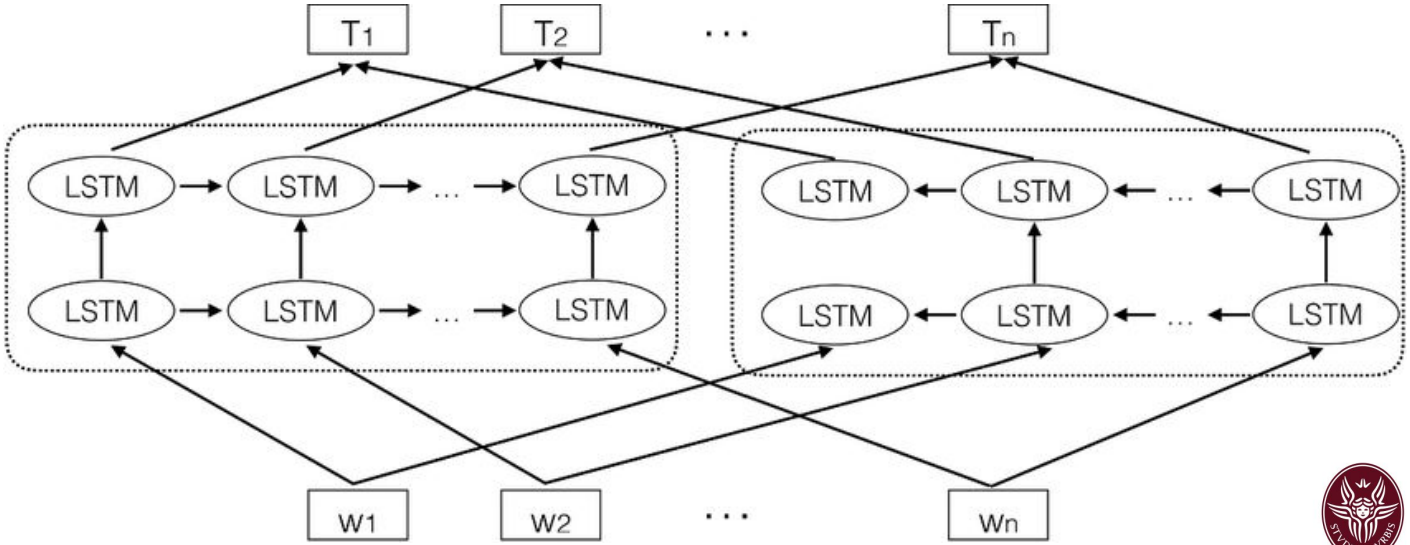
ELMo

A Big Drawback of Word2Vec

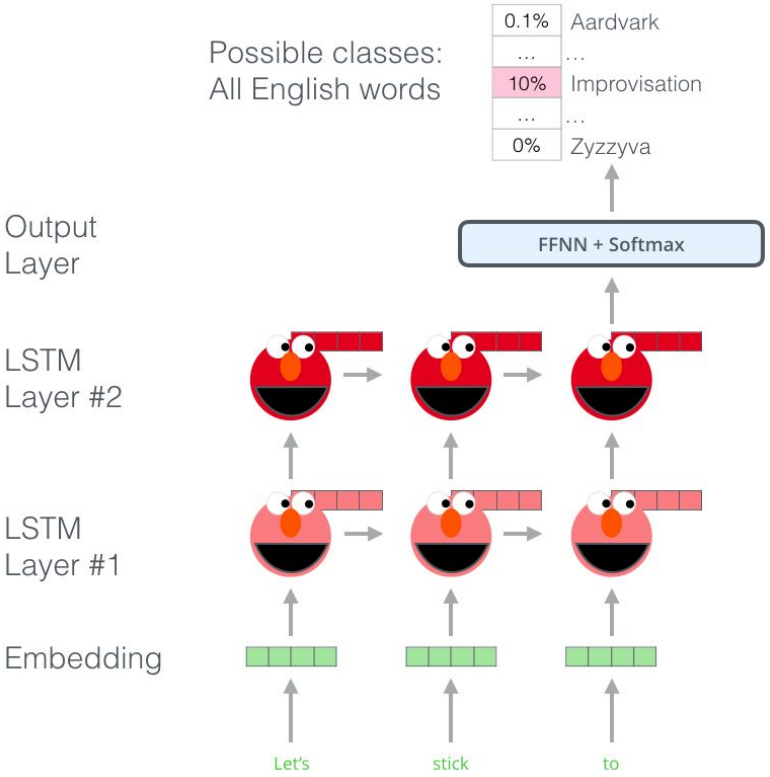
- Word2Vec (or word embeddings in general) associate one vector to each symbol.
- One vector!
- How about:
  - Feet:
    - My feet hurt
    - This tree is 6 feet tall
  - Rock:
    - Rock paper and scissor
    - Rock ‘n’ roll
  - Break (in English) has more than 70 different meanings!!!

Enters ElMo

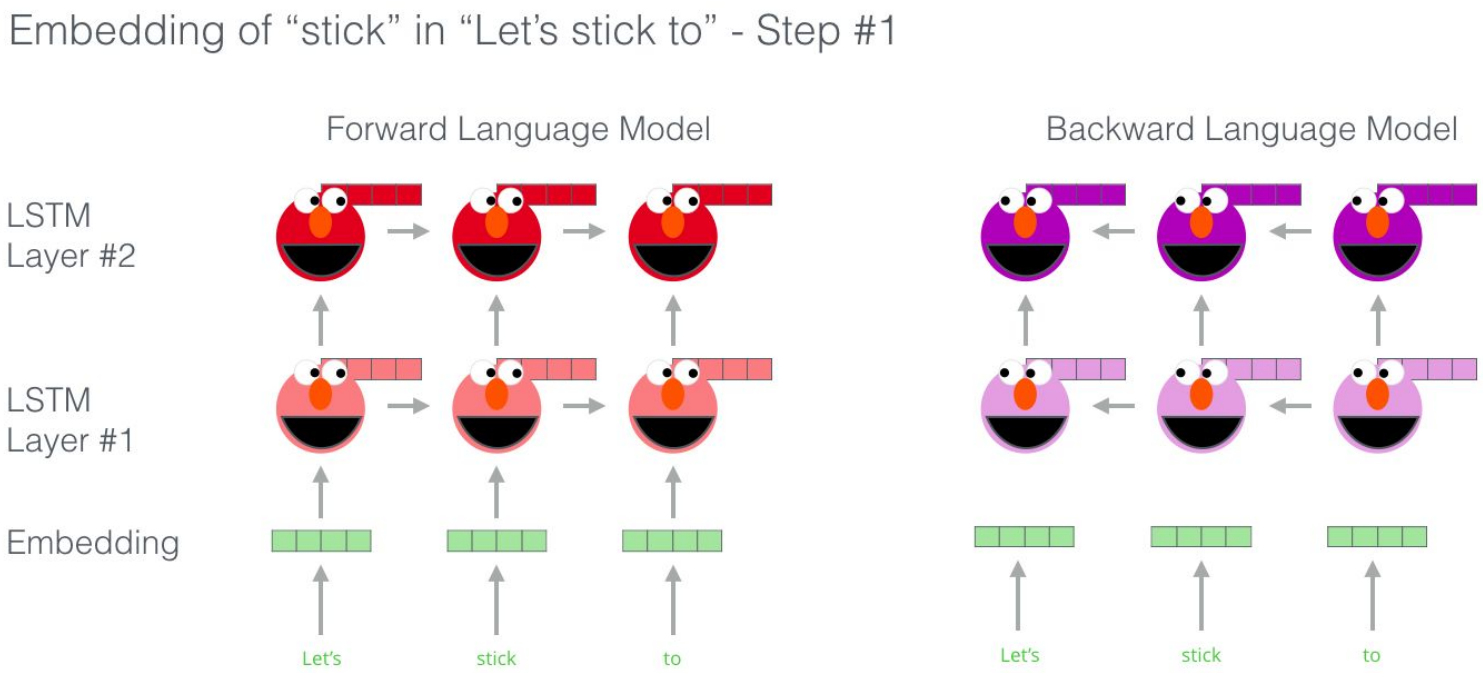
- It trains contextual word embeddings whose value is not static but is computed every time on the basis of the context (i.e, surrounding words)



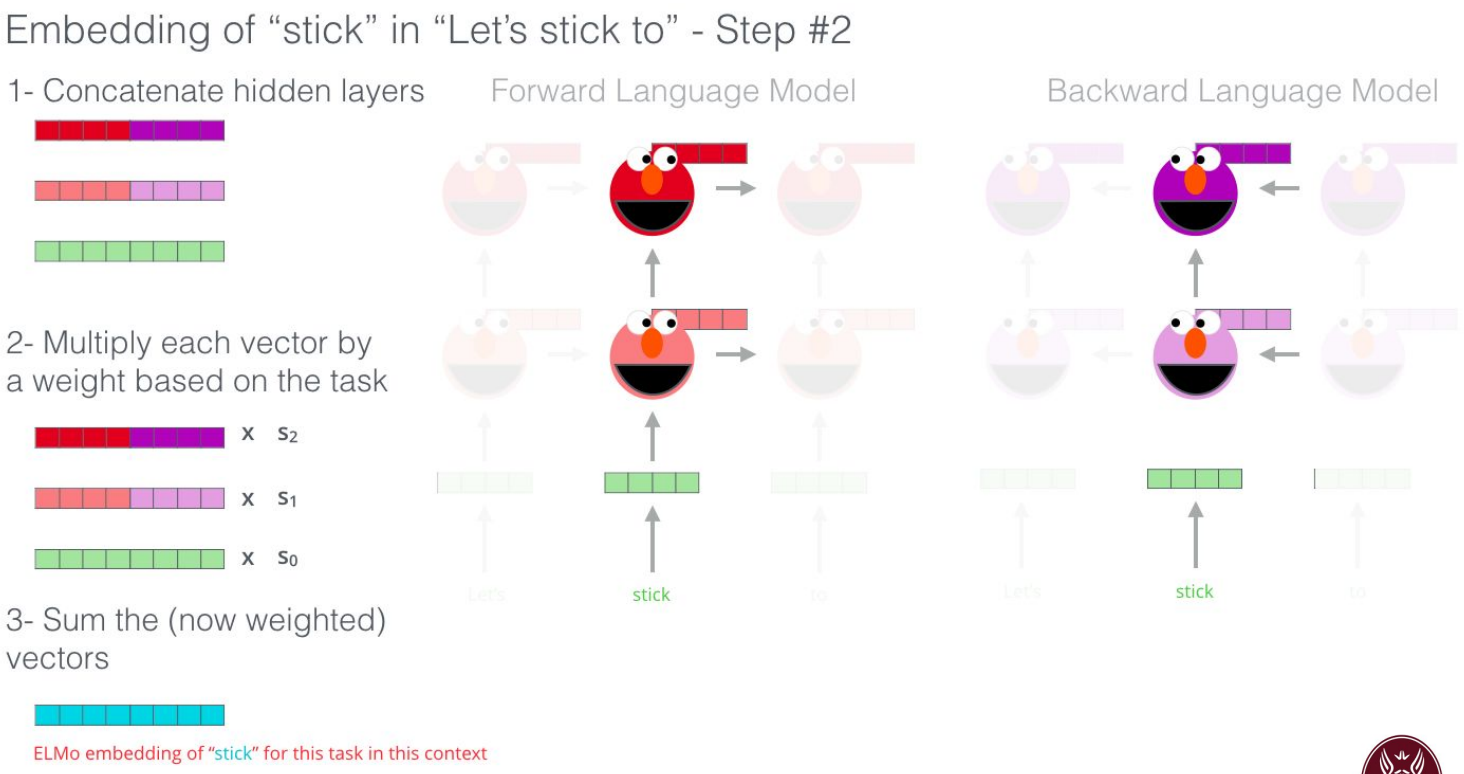
An Example



An Example

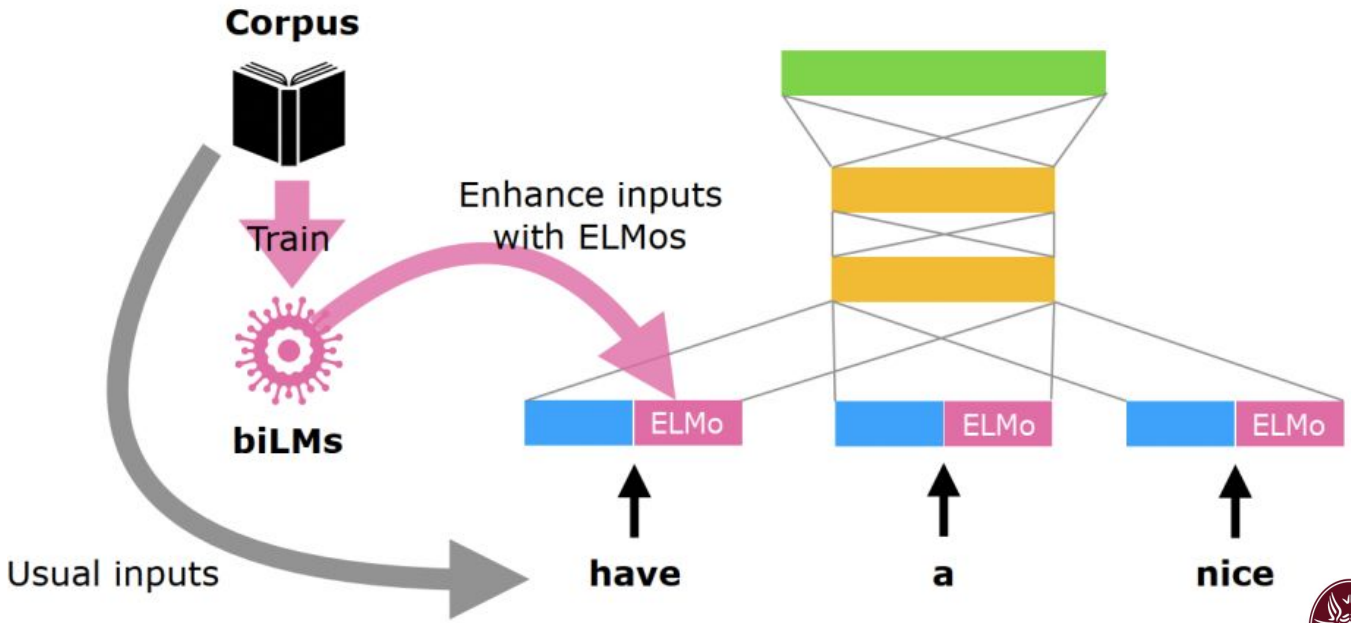


An Example



ELMo as Word Embedding

ELMo can be integrated to almost all neural NLP tasks with simple concatenation to the embedding layer



Experiments

TASK	PREVIOUS SOTA		OUR BASELINE	ELMO + BASELINE	INCREASE (ABSOLUTE/ RELATIVE)
SQuAD	Liu et al. (2017)	84.4	81.1	85.8	4.7 / 24.9%
SNLI	Chen et al. (2017)	88.6	88.0	88.7 ± 0.17	0.7 / 5.8%
SRL	He et al. (2017)	81.7	81.4	84.6	3.2 / 17.2%
Coref	Lee et al. (2017)	67.2	67.2	70.4	3.2 / 9.8%
NER	Peters et al. (2017)	91.93 ± 0.19	90.15	92.22 ± 0.10	2.06 / 21%
SST-5	McCann et al. (2017)	53.7	51.4	54.7 ± 0.5	3.3 / 6.8%

Table 1: Test set comparison of ELMo enhanced neural models with state-of-the-art single model baselines across six benchmark NLP tasks. The performance metric varies across tasks – accuracy for SNLI and SST-5; F_1 for SQuAD, SRL and NER; average F_1 for Coref. Due to the small test sizes for NER and SST-5, we report the mean and standard deviation across five runs with different random seeds. The "increase" column lists both the absolute and relative improvements over our baseline.

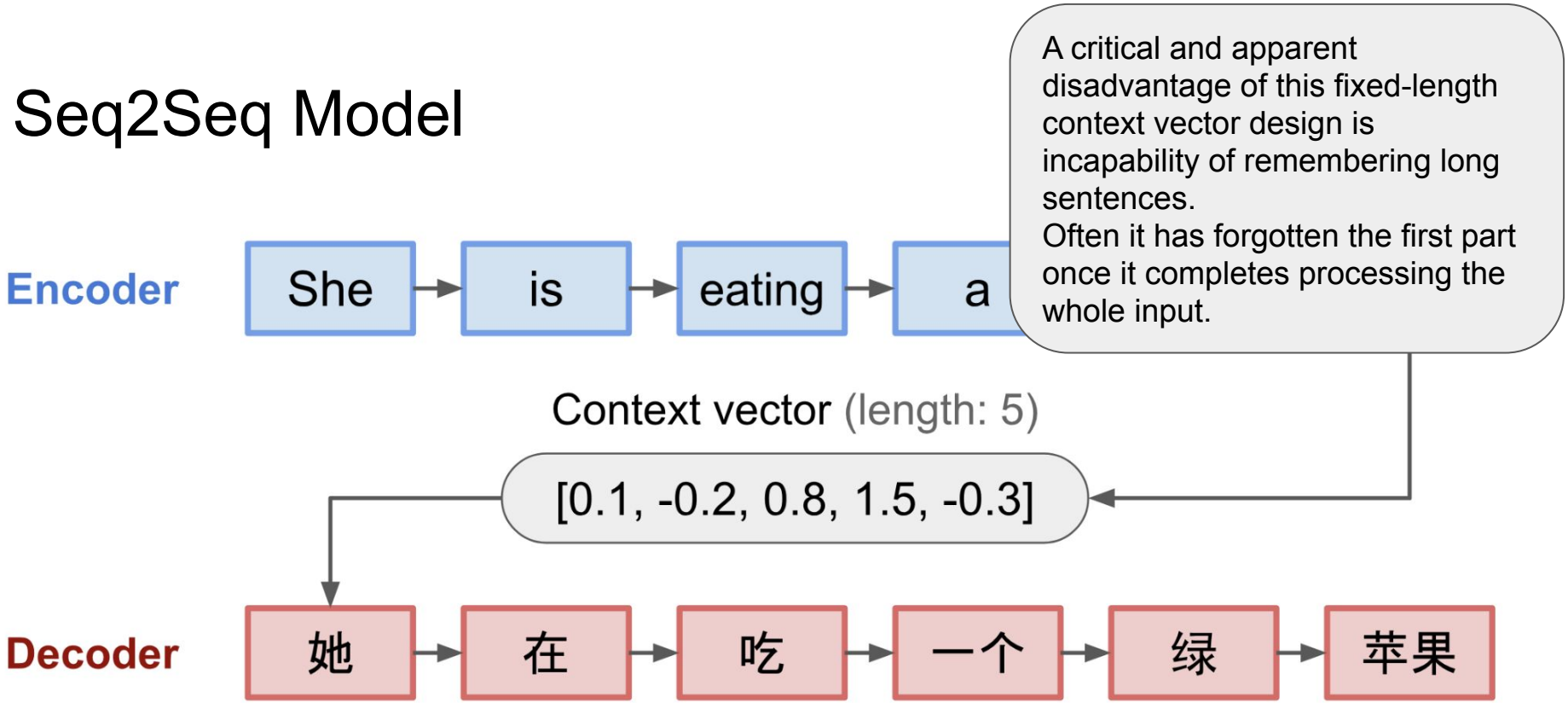
No… Not that kind of Transformers

SAPIENZA
UNIVERSITÀ DI ROMA

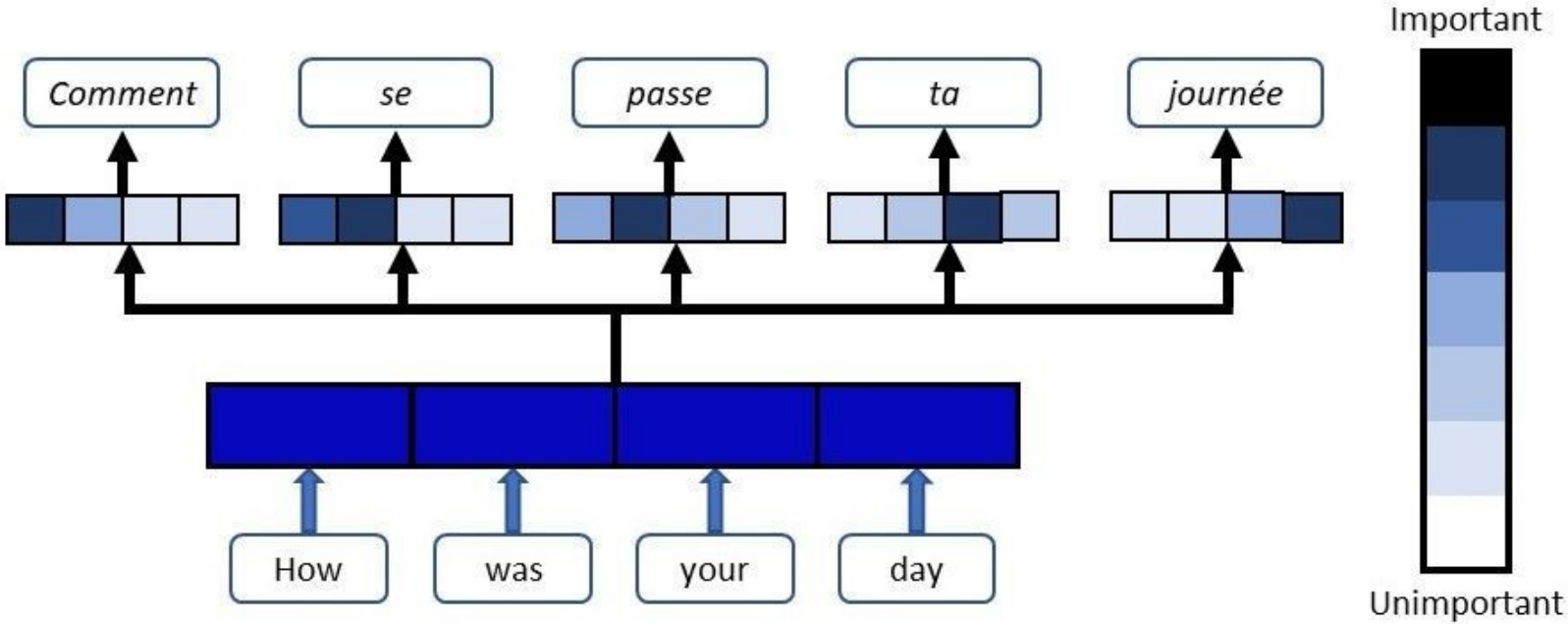
Attention Mechanism

SAPIENZA
UNIVERSITÀ DI ROMA

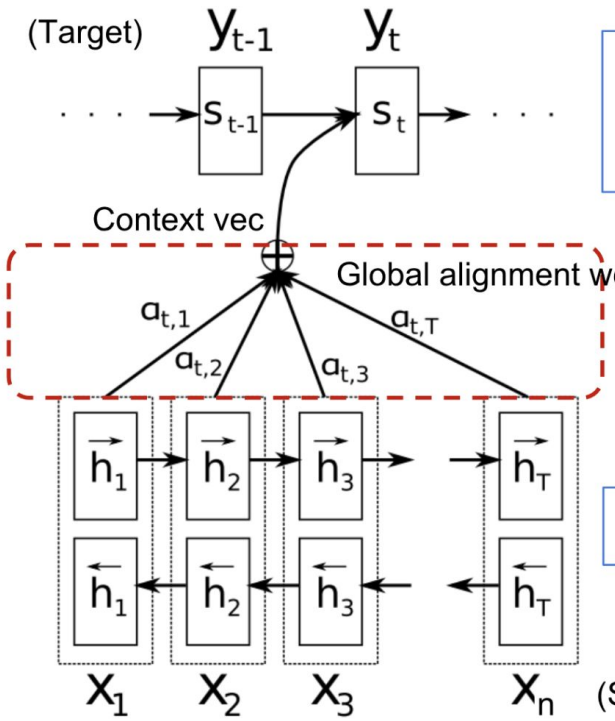
Seq2Seq Model



Let’s focus on a single output



The Attention Mechanism



Decoder: RNN with input from previous state + dynamic context vector.

Attention layer: parameterized by a simple feed-forward network

Additive Attention

Encoder: bidirectional RNN

The Attention Mechanism

- Say, we have a source sequence x of length n and try to output a target sequence y of length m:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]$$
$$\mathbf{y} = [y_1, y_2, \ldots, y_m]$$

- The encoder is a bidirectional RNN with a forward hidden state $\overrightarrow{\mathbf{h}}_i$ and a backward one $\overleftarrow{\mathbf{h}}_i$.
  - A simple concatenation of two represents the encoder state.
  - The motivation is to include both the preceding and following words in the annotation of one word.

$$\boldsymbol{h}_i = [\overrightarrow{\boldsymbol{h}}_i^{\top}; \overleftarrow{\boldsymbol{h}}_i^{\top}]^{\top}, i = 1, \ldots, n$$

The Attention Mechanism

- The decoder network has hidden state $\mathbf{s}_t = f(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t)$ for the output word at position t, t=1,…,m, where the context vector $\mathbf{c}_t$ is a sum of hidden states of the input sequence, weighted by alignment scores:

$$\begin{aligned}
\mathbf{c}_t &= \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i && \text{; Context vector for output } y_t \\
\alpha_{t,i} &= \text{align}(y_t, x_i) && \text{; How well two words } y_t \text{ and } x_i \text{ are aligned.} \\
&= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))} && \text{; Softmax of some predefined alignment score..}
\end{aligned}$$

The alignment model assigns a score $\alpha_{t,i}$ to the pair of input at position i and output at position t, (y_t, x_i), based on how well they match. The set of $\{\alpha_{t,i}\}$ are weights defining how much of each source hidden state should be considered for each output.

The Attention Mechanism

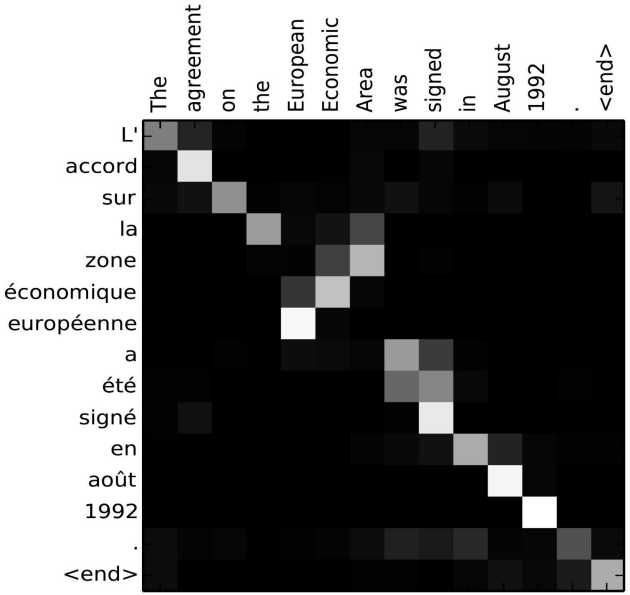
- The alignment score α is parametrized by a feed-forward network with a single hidden layer and this network is jointly trained with other parts of the model.
- The score function is therefore in the following form, given that tanh is used as the non-linear activation function:

$$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$$

where both $\mathbf{v}_a$ and $\mathbf{W}_a$ are weight matrices to be learned in the alignment model.

The Attention Mechanism

- The matrix of alignment scores is a nice byproduct to explicitly show the correlation between source and target words.



A Family of Attention Scores

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
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.	Luong2015
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.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
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.	Vaswani2017

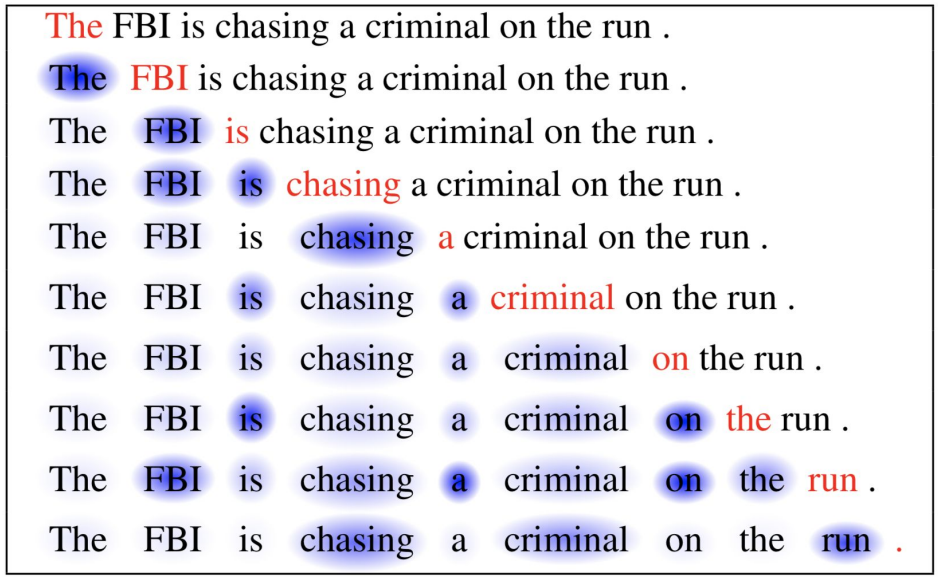
SAPIENZA
UNIVERSITÀ DI ROMA

Self-Attention

Self-Attention

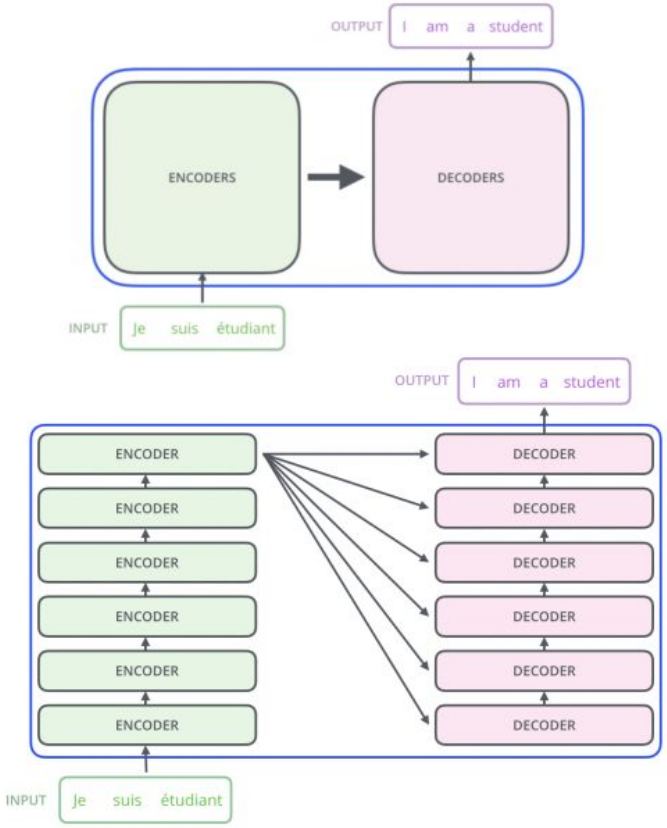
Self-attention, also known as intra-attention, is an attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence.

It has been shown to be very useful in machine reading, abstractive summarization, or image description generation.



The current word is in red and the size of the blue shade indicates the activation level. (Image source: Cheng et al., 2016)

Transformer: a seq2seq model



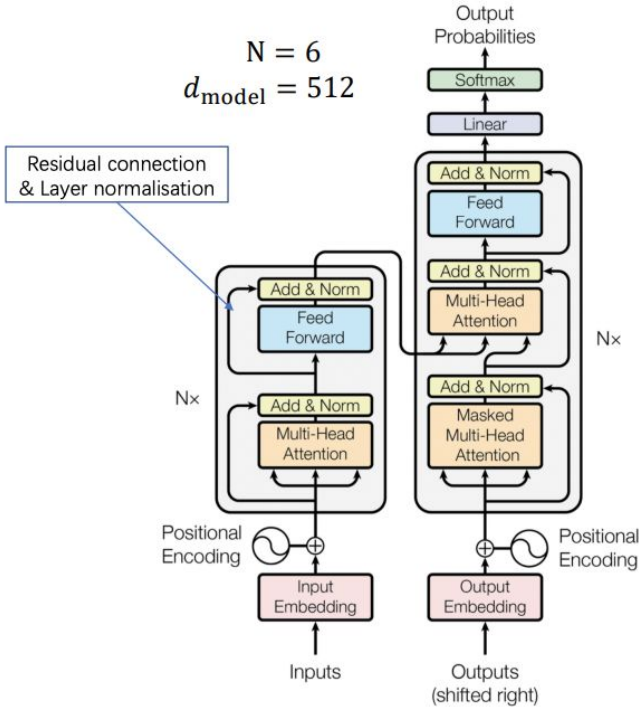


Figure 1: The Transformer - model architecture.

Figure in (Vaswani *et al.*, 2017)

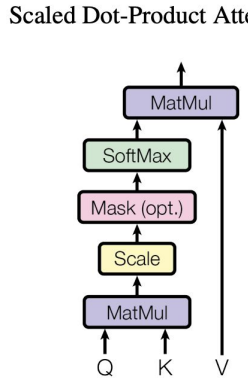
"Attention is All you Need" (Vaswani, et al., 2017), without a doubt, is one of the most impactful and interesting paper published in 2017.

It makes it possible to do seq2seq modeling without recurrent network units.

Key, Value and Query

- The transformer views the encoded representation of the input as a set of key-value pairs, (K,V), both of dimension n (the input sequence length)
  - For example, in the context of NMT, both the keys and values are the encoder hidden states.
- In the decoder, the previous output is compressed into a query Q (of dimension m) and the next output is produced by mapping this query and the set of keys and values.
- The transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}})\mathbf{V}$$



Key, Value and Query: An Example

- Consider this phrase – “Action gets results”.
- To calculate the self-attention for the first word “Action”, we will calculate scores for all the words in the phrase with respect to “Action”.
- This score determines the importance of other words when we are encoding a certain word in an input sequence.

Key, Value and Query: An Example

- The score for the first word is calculated by taking the dot product of the Query vector (q_1) with the keys vectors (k_1, k_2, k_3) of all the words.

Word	q vector	k vector	v vector	score
Action	q_1	k_1	v_1	$q_1 . k_1$
gets		k_2	v_2	$q_1 . k_2$
results		k_3	v_3	$q_1 . k_3$

Key, Value and Query: An Example

- These scores are divided by 8 which is the square root of the dimension of the key vector

Word	q vector	k vector	v vector	score	score / 8
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$

Key, Value and Query: An Example

- Next, these scores are normalized using the softmax activation function.

Word	q vector	k vector	v vector	score	score / 8	Softmax
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$	x_{11}
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$	x_{12}
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$	x_{13}

Key, Value and Query: An Example

- These normalized scores are then multiplied by the value vectors (v1, v2, v3) and sum up the resultant vectors to arrive at the final vector (z1).
- This is the output of the self-attention layer. It is then passed on to the feed-forward network as input.

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum
Action	q_1	k_1	v_1	$q_1 . k_1$	$q_1 . k_1 / 8$	x_{11}	$x_{11} * v_1$	z_1
gets		k_2	v_2	$q_1 . k_2$	$q_1 . k_2 / 8$	x_{12}	$x_{12} * v_2$	
results		k_3	v_3	$q_1 . k_3$	$q_1 . k_3 / 8$	x_{13}	$x_{13} * v_3$	

SAPIENZA
UNIVERSITÀ DI ROMA

Key, Value and Query: An Example

- So, z1 is the self-attention vector for the first word of the input sequence “Action gets results”.
- We can get the vectors for the rest of the words in the input sequence in the same fashion.

Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum#
Action		k_1	v_1	$q_2 . k_1$	$q_2 . k_1 / 8$	x_{21}	$x_{21} * v_1$	
gets	q_2	k_2	v_2	$q_2 . k_2$	$q_2 . k_2 / 8$	x_{22}	$x_{22} * v_2$	z_2
results		k_3	v_3	$q_2 . k_3$	$q_2 . k_3 / 8$	x_{23}	$x_{23} * v_3$	

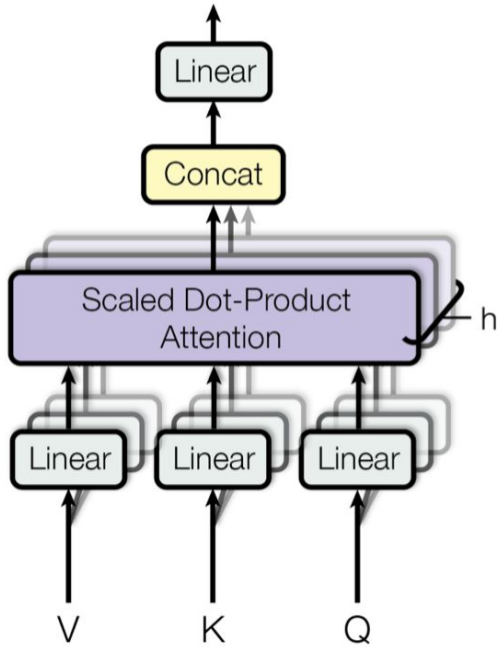
Word	q vector	k vector	v vector	score	score / 8	Softmax	Softmax * v	Sum#
Action		k_1	v_1	$q_3 . k_1$	$q_3 . k_1 / 8$	x_{31}	$x_{31} * v_1$	
gets		k_2	v_2	$q_3 . k_2$	$q_3 . k_2 / 8$	x_{32}	$x_{32} * v_2$	
results	q_3	k_3	v_3	$q_3 . k_3$	$q_3 . k_3 / 8$	x_{33}	$x_{33} * v_3$	z_3

Multi-head Self-Attention

- Rather than only computing the attention once, the multi-head mechanism runs through the scaled dot-product attention multiple times in parallel.
- “multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.”

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \ldots; \text{head}_h]\mathbf{W}^O$$
$$\text{where head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$$

where $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$, and $\mathbf{W}^O$ are parameter matrices to be learned.



Understanding Self-Attention

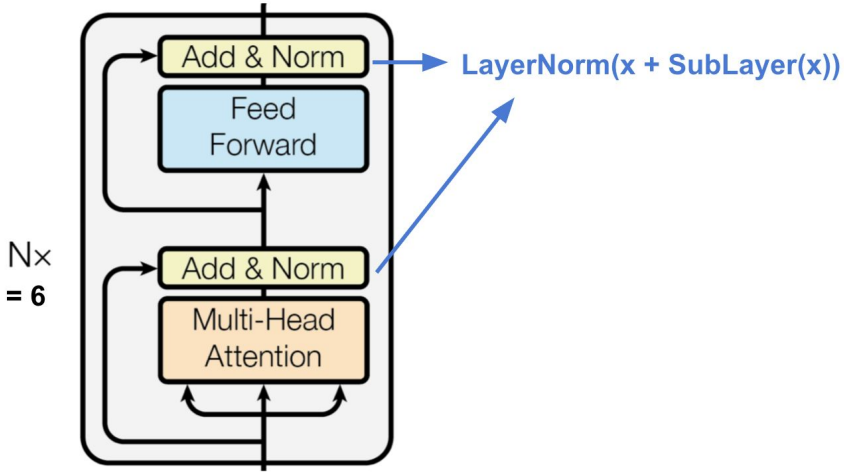
SAPIENZA
UNIVERSITÀ DI ROMA

The Transformer Architecture

Encoder

The encoder generates an attention-based representation with capability to locate a specific piece of information from a potentially infinitely-large context.

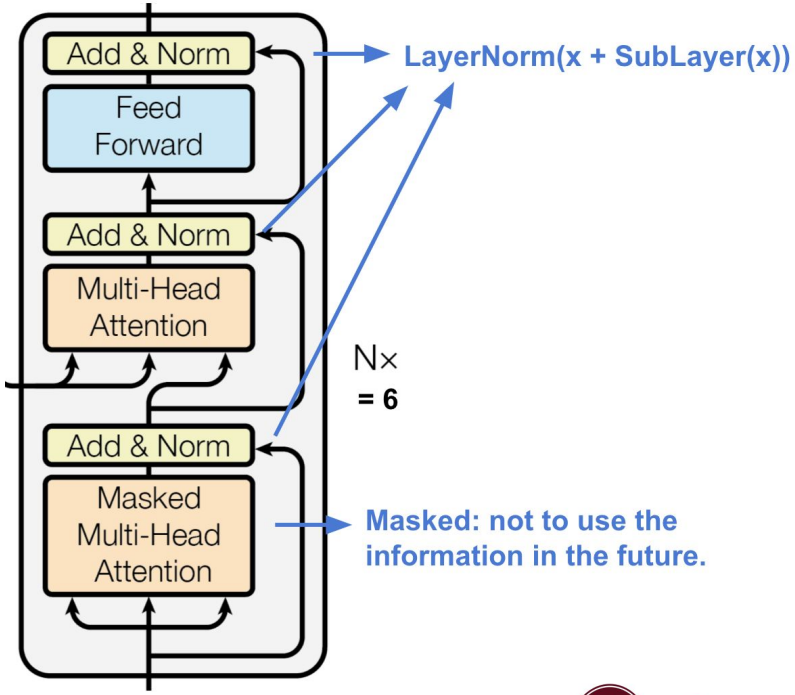
- A stack of N=6 identical layers.
- Each layer has a multi-head self-attention layer and a simple position-wise fully connected feed-forward network.
- Each sub-layer adopts a residual connection and a layer normalization.
- All the sub-layers output data of the same dimension d_{model}=512.



Decoder

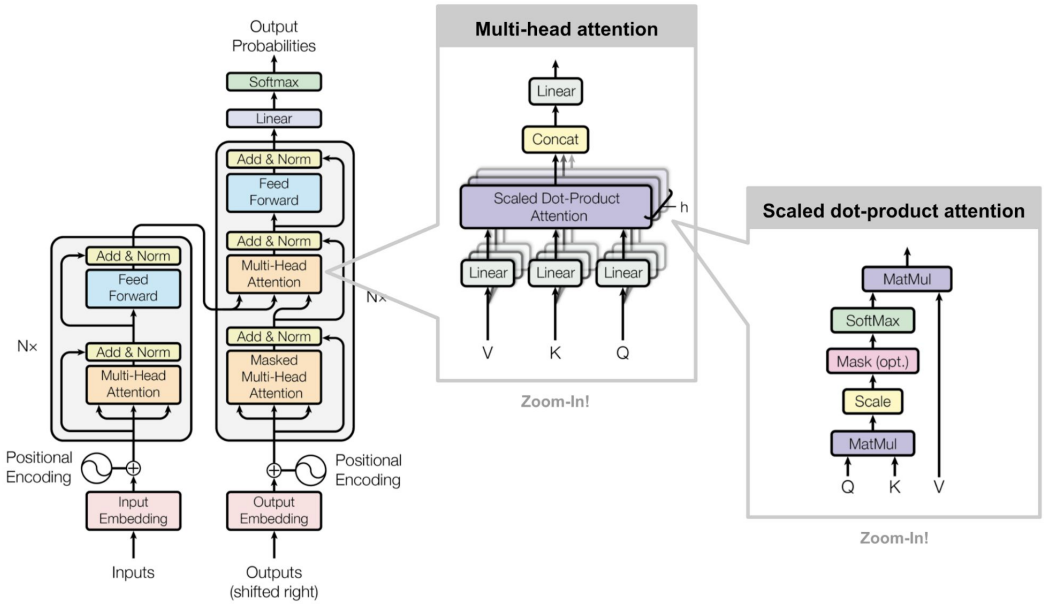
The decoder is able to retrieval from the encoded representation.

- A stack of N = 6 identical layers
- Each layer has two sub-layers of multi-head attention mechanisms and one sub-layer of fully-connected feed-forward network.
- Similar to the encoder, each sub-layer adopts a residual connection and a layer normalization.
- The first multi-head attention sub-layer is modified to prevent positions from attending to subsequent positions, as we don't want to look into the future of the target sequence when predicting the current position.



The Full Architecture

- Both the source and target sequences first go through embedding layers to produce data of the same dimension d_{model}=512.
- To preserve the position information, a sinusoid-wave-based positional encoding is applied and summed with the embedding output.
- A softmax and linear layer are added to the final decoder output.
- The output from the last encoder in the encoder-stack is passed to all the decoders in the decoder-stack



Why Transformers are Preferable to RNNs?

- The total computational complexity per layer
- The amount of computation that can be parallelized
  - minimum number of sequential operations required
- The path length between long-range dependencies in the network.
  - Learning long-range dependencies is a key challenge in many sequence transduction tasks.

Table 1: Maximum path lengths, per-layer complexity and minimum number of sequential operations for different layer types. n is the sequence length, d is the representation dimension, k is the kernel size of convolutions and r the size of the neighborhood in restricted self-attention.

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

Experiments on NMT

The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

SAPIENZA
UNIVERSITÀ DI ROMA

The Annotated Transformer

BERT

Bidirectional **E**ncoder **R**epresentations from **T**ransformers

SAPIENZA
UNIVERSITÀ DI ROMA

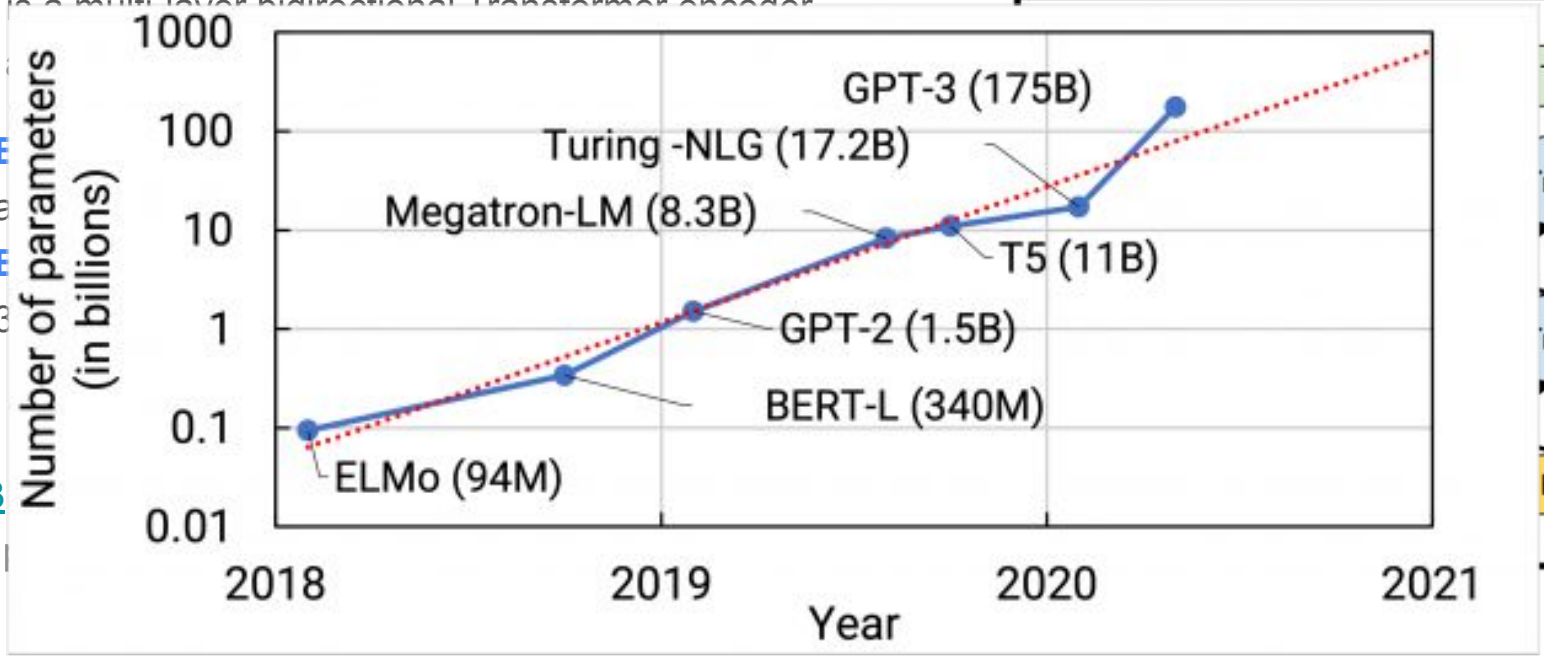
The Architecture

BERT is a multi-layer bidirectional Transformer encoder

There

- E
  a
- E
  3

GPT-3
billion



SAPIENZA
UNIVERSITÀ DI ROMA

Pretraining BERT: Masked LMs and Next Sentence Prediction

- In masked language modeling instead of predicting every next token, a percentage of input tokens is masked at random and only those masked tokens are predicted.
  - Bi-directional models are more powerful than uni-directional language models.
  - But in a multi-layered model bi-directional models do not work because the lower layers leak information and allow a token to see itself in later layers.
- Similarly, next sentence prediction takes a pair of sentences and try to predict if one follows the other in the training dataset.

Masked Language Model

- 80% of the time: Replace the word with the `[MASK]` token, e.g., `my dog is hairy` → `my dog is [MASK]`
- 10% of the time: Replace the word with a random word, e.g., `my dog is hairy` → `my dog is apple`
- 10% of the time: Keep the word unchanged, e.g., `my dog is hairy` → `my dog is hairy`. The purpose of this is to bias the representation towards the actual observed word.

Next Sequence Prediction

Input = [CLS] the man went to [MASK] store [SEP]

he bought a gallon [MASK] milk [SEP]

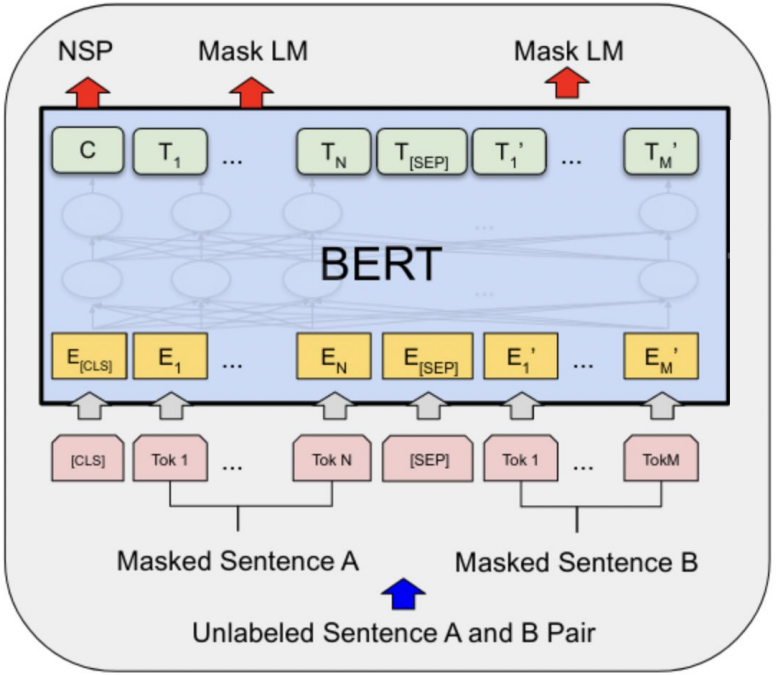
Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]

penguin [MASK] are flight ##less birds [SEP]

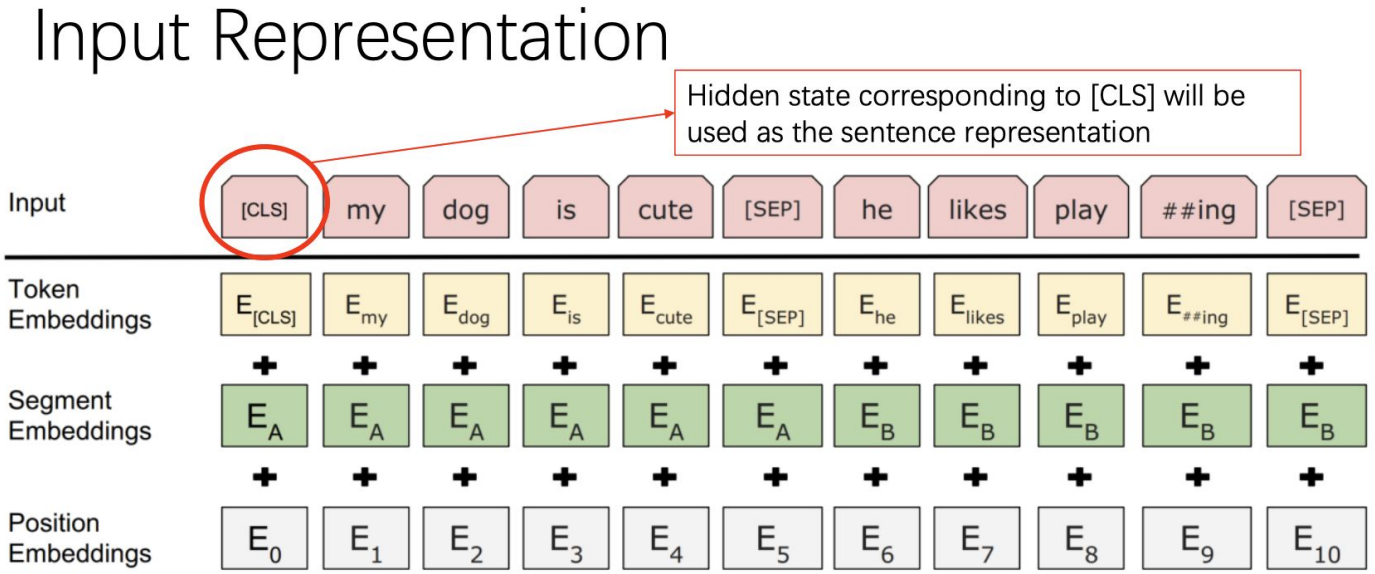
Label = NotNext

Pretraining BERT: Masked LMs and Next Sentence Prediction



Pre-training

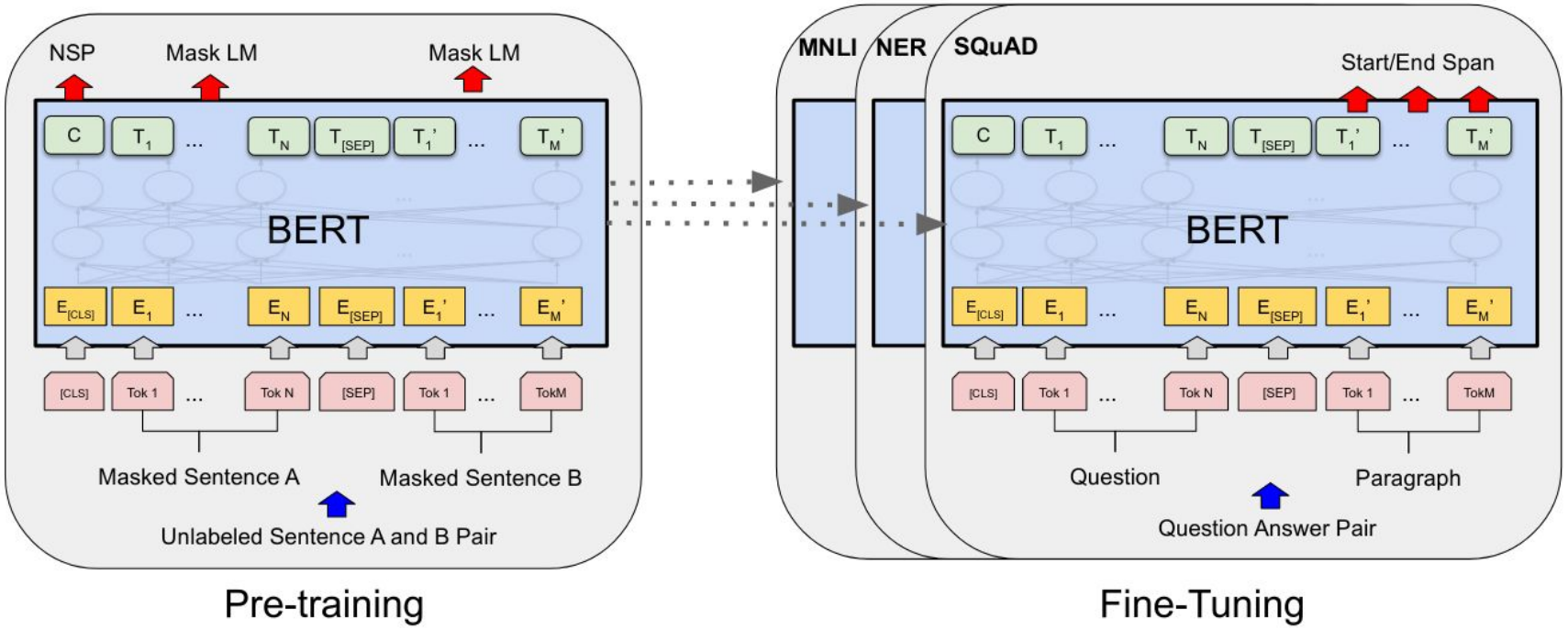
Input Representation



- Token Embeddings: WordPiece embedding (Wu *et al.*, 2016)
- Segment Embeddings: randomly initialized and learned; single sentence input only adds E_A
- Position embeddings: randomly initialized and learned

Figure in (Devlin et al., 2018)

Fine Tuning BERT



Fine Tuning BERT: Sequence Classification

The pre-trained model is trained on a supervised dataset to predict the class of a given sequence. Since the output of the BERT (Transformer Encoder) model is the hidden state for all the tokens in the sequence, the output needs to be pooled to obtain only one label. The Classification token ([CLS] token) is used here. The output of this token is considered as the classifier pooled output and it is further put into a fully-connected classification layer for obtaining the labeled output.

Fine Tuning BERT: Named Entity Recognition (NER)

The hidden state outputs are directly put into a classifier layer with the number of tags as the output units for each of the token. Then these logits are used to obtain the predicted class of each token using argmax.

Fine Tuning BERT: Natural Language Inference (NLI) or Textual Entailment

We train BERT the same as in the NSP task, with both the sentences i.e. the text and the hypothesis separated using [SEP] token, and are identified using the Segment Embeddings. The [CLS] token is used to obtain the classification result as explained in the Sequence Classification part.

Fine Tuning BERT: Grounded Common Sense Inference

Given a sentence, the task is to choose the most plausible continuation among four choices. We take 4 input sequences, each containing the original sentence and the corresponding choice concatenated to it. Here too, we use the [CLS] token state and feed it to a fully-connected layer to obtain the scores for each of the choices which are then normalized using a softmax layer.

Fine Tuning BERT: Question Answering

A paragraph is used as one sequence and the question is used as another. There are 2 cases in this task:

1. The answer is expected to be found within the paragraph. Here, we intend to find the Start and the End token of the answer from the paragraph. For this, we take the dot product of each token T_i and the start token S to obtain the probability of the token i to be the start of the answer. Similarly, we obtain the probability of the end token j. The score of a candidate span from position i to position j is defined as $S.T_i + E.T_j$, and the maximum scoring span where $j \geq i$ is used as a prediction.
2. we consider the possibility that there may be no short answer to the question present in the paragraph, which is a more realistic case. For this, we consider the probability scores for the start and end of the answer at the [CLS] token itself. We call this as s_{null}. We compare this s_{null} with the max score obtained at the best candidate span (i.e. the best score for the first case). If this obtained score is greater than s_{null} by a sufficient threshold τ, then we use the candidate best score as the answer. The threshold τ can be tuned to maximize the dev set F1-score.

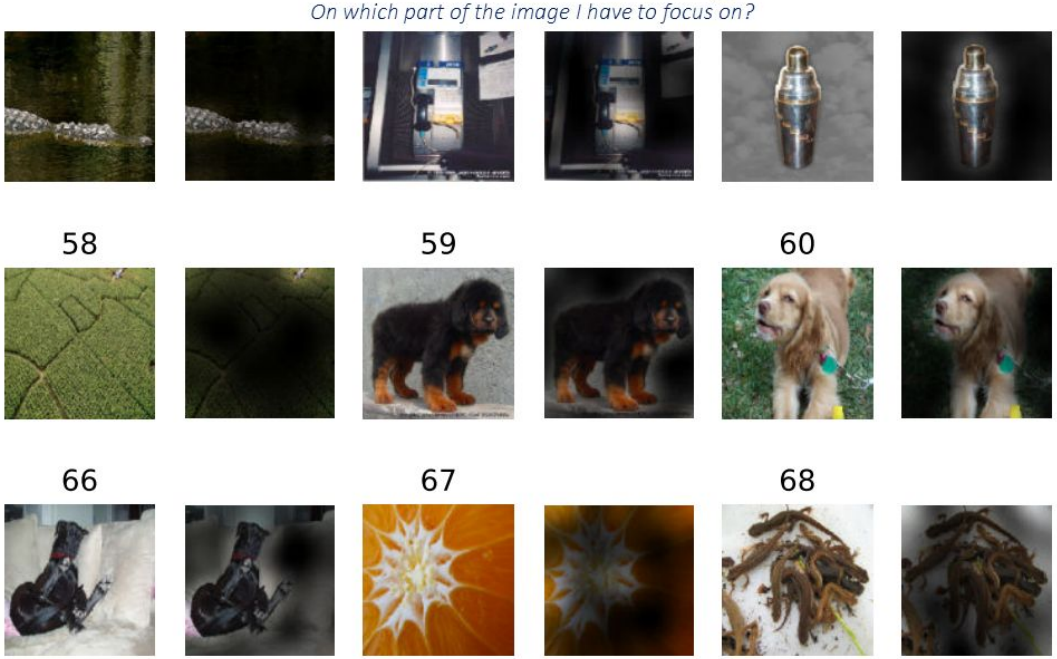
Experiments: GLUE Tests

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Table 1: GLUE Test results, scored by the evaluation server (https://gluebenchmark.com/leaderboard). The number below each task denotes the number of training examples. The "Average" column is slightly different than the official GLUE score, since we exclude the problematic WNLI set.[8] BERT and OpenAI GPT are single-model, single task. F1 scores are reported for QQP and MRPC, Spearman correlations are reported for STS-B, and accuracy scores are reported for the other tasks. We exclude entries that use BERT as one of their components.

Visual Transformers

Attention In Images



Can we use Pixels instead of Wordpieces in Transformers?

Treat pixels as words and calculate attention!

Chop Input and apply Transformers

An image is worth 16x16 words!

Chop Input and apply Transformers

Experiments

	Ours-JFT (ViT-H/14)	Ours-JFT (ViT-L/16)	Ours-I21k (ViT-L/16)	BiT-L (ResNet152x4)	Noisy Student (EfficientNet-L2)
ImageNet	**88.55** ± 0.04	87.76 ± 0.03	85.30 ± 0.02	87.54 ± 0.02	88.4/88.5*
ImageNet ReaL	**90.72** ± 0.05	90.54 ± 0.03	88.62 ± 0.05	90.54	90.55
CIFAR-10	**99.50** ± 0.06	99.42 ± 0.03	99.15 ± 0.03	99.37 ± 0.06	–
CIFAR-100	**94.55** ± 0.04	93.90 ± 0.05	93.25 ± 0.05	93.51 ± 0.08	–
Oxford-IIIT Pets	**97.56** ± 0.03	97.32 ± 0.11	94.67 ± 0.15	96.62 ± 0.23	–
Oxford Flowers-102	99.68 ± 0.02	**99.74** ± 0.00	99.61 ± 0.02	99.63 ± 0.03	–
VTAB (19 tasks)	**77.63** ± 0.23	76.28 ± 0.46	72.72 ± 0.21	76.29 ± 1.70	–
TPUv3-core-days	2.5k	0.68k	0.23k	9.9k	12.3k

Table 2: Comparison with state of the art on popular image classification benchmarks. We report mean and standard deviation of the accuracies, averaged over three fine-tuning runs. Vision Transformer models pre-trained on the JFT-300M dataset outperform ResNet-based baselines on all datasets, while taking substantially less computational resources to pre-train. ViT pre-trained on the smaller public ImageNet-21k dataset performs well too. *Slightly improved 88.5% result reported in Touvron et al. (2020).

Transfer Learning

Transfer Learning with Image Data

- For example, a prediction task that takes photographs or video data as input.
- It is common to use a deep learning model pre-trained for a large and challenging image classification task such as the ImageNet 1000-class photograph classification competition.
- Three examples of models of this type include:
  - Oxford VGG Model
  - Google Inception Model
  - Microsoft ResNet Model

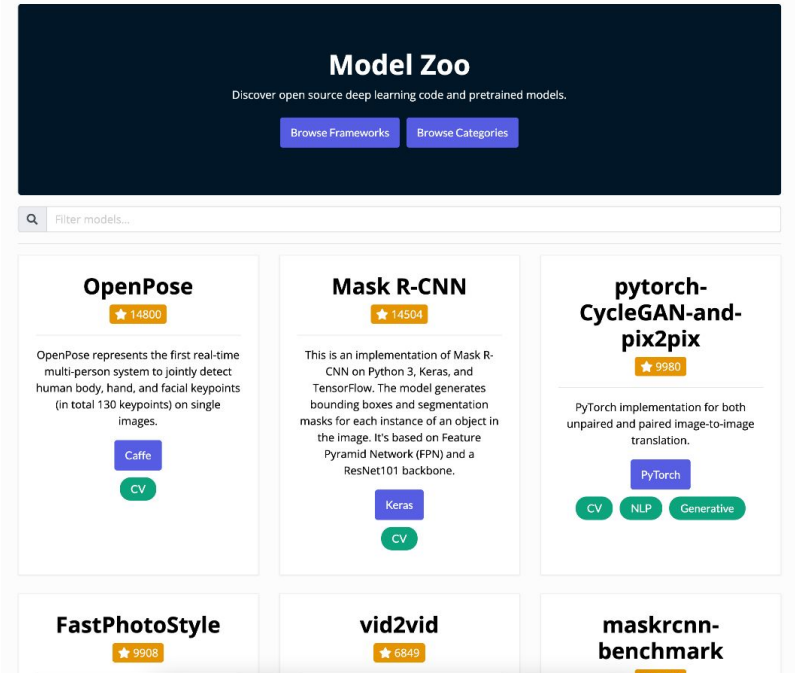
Model Zoo

- It is common to use a Model Zoo.
- For Pytorch: https://pytorch.org/serve/model_zoo.html

Model	Type	Dataset	Size	Download	Sample Input	Model mode
AlexNet	Image Classification	ImageNet	216 MB	.mar	kitten.jpg	Eager
Densenet161	Image Classification	ImageNet	106 MB	.mar	kitten.jpg	Eager
Resnet18	Image Classification	ImageNet	41 MB	.mar	kitten.jpg	Eager
VGG16	Image Classification	ImageNet	489 MB	.mar	kitten.jpg	Eager
Squeezenet 1_1	Image Classification	ImageNet	4.4 MB	.mar	kitten.jpg	Eager
MNIST digit classifier	Image Classification	MNIST	4.3 MB	.mar	0.png	Eager
Resnet 152	Image Classification	ImageNet	214 MB	.mar	kitten.jpg	Eager
Faster RCNN	Object Detection	COCO	148 MB	.mar	persons.jpg	Eager
MASK RCNN	Object Detection	COCO	158 MB	.mar	persons.jpg	Eager

ModelZoo.co

- For other frameworks (including Pytorch): https://modelzoo.co/



How to use Pretrained Models

- Colab Example

  https://colab.research.google.com/drive/1v82guGHX--OHwNdmLEQCGaaQVSjlgLgl

Transfer Learning with Text Data

- It is common to perform transfer learning with natural language processing problems that use text as input or output.
- First class of pre-trained models: word embedding
- Other examples:
  - BERT
  - T5
  - ...

HuggingFace!

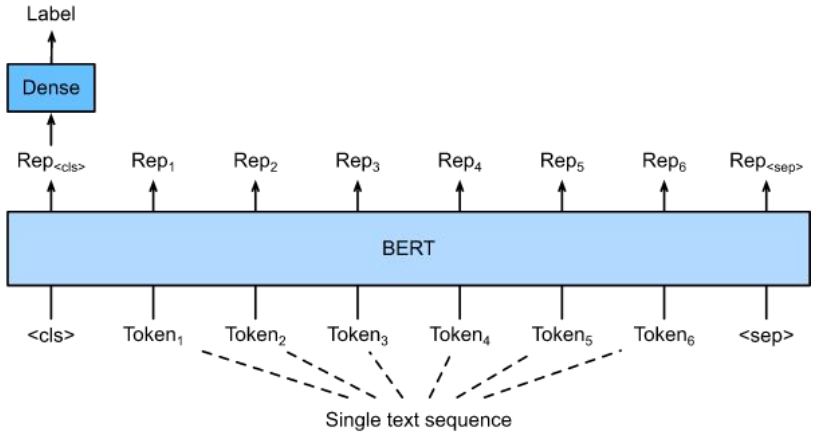
- We have already seen Transformers
- We are now going to see how to actually use them in your ML Pipelines...

HuggingFace!

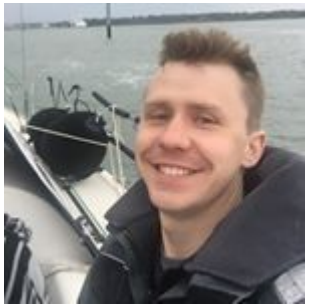
Transformers

How to fine-tune HF Pretrained Models

- Sentiment analysis using HF's BERT and fine-tune it.



An application of Transfer Learning: NeuralDBs



James Thorne
Cambridge

Majid Yazdani
Facebook AI

Marzieh Saeidi
Facebook AI

Sebastian Riedel
Facebook AI

Alon Halevy
Facebook AI

- *J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, and A. Halevy. From natural language processing to neural databases.* ***PVLDB 2021***
- *J. Thorne, M. Yazdani, M. Saeidi, F. Silvestri, S. Riedel, and A. Halevy. Database reasoning over text. To appear in* ***ACL 2022****.*

UNIVERSITY OF CAMBRIDGE

SAPIENZA UNIVERSITÀ DI ROMA









The talk is based on the following papers…

1. Giovanni Trappolini, Andrea Santilli, Emanuele Rodolà, Alon Y. Halevy, Fabrizio Silvestri: **Multimodal Neural Databases**. *SIGIR 2023*
2. Artsiom Sauchuk, James Thorne, Alon Y. Halevy, Nicola Tonellotto, Fabrizio Silvestri: **On the Role of Relevance in Natural Language Processing Tasks**. *SIGIR 2022*
3. James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, Alon Y. Halevy: **From Natural Language Processing to Neural Databases**. *VLDB 2021*
4. James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, Alon Y. Halevy: **Database reasoning over text**. *ACL/IJCNLP 2021*

Databases have a Well-Defined Formal Language

```
INSERT INTO People (PersonID, PersonName, Country) VALUES (123, 'Fabrizio Silvestri', 'Italy');
INSERT INTO People (PersonID, PersonName, Country) VALUES (789, 'Marzieh Saeidi', 'UK');

INSERT INTO Jobs (JobID, JobDescription) VALUES (111, 'Software Engineer')
INSERT INTO Jobs (JobID, JobDescription) VALUES (123, 'Research Scientist')

INSERT INTO PeopleJobs (PersonID, JobID) VALUES (123, 111)
INSERT INTO PeopleJobs (PersonID, JobID) VALUES (789, 123)
```

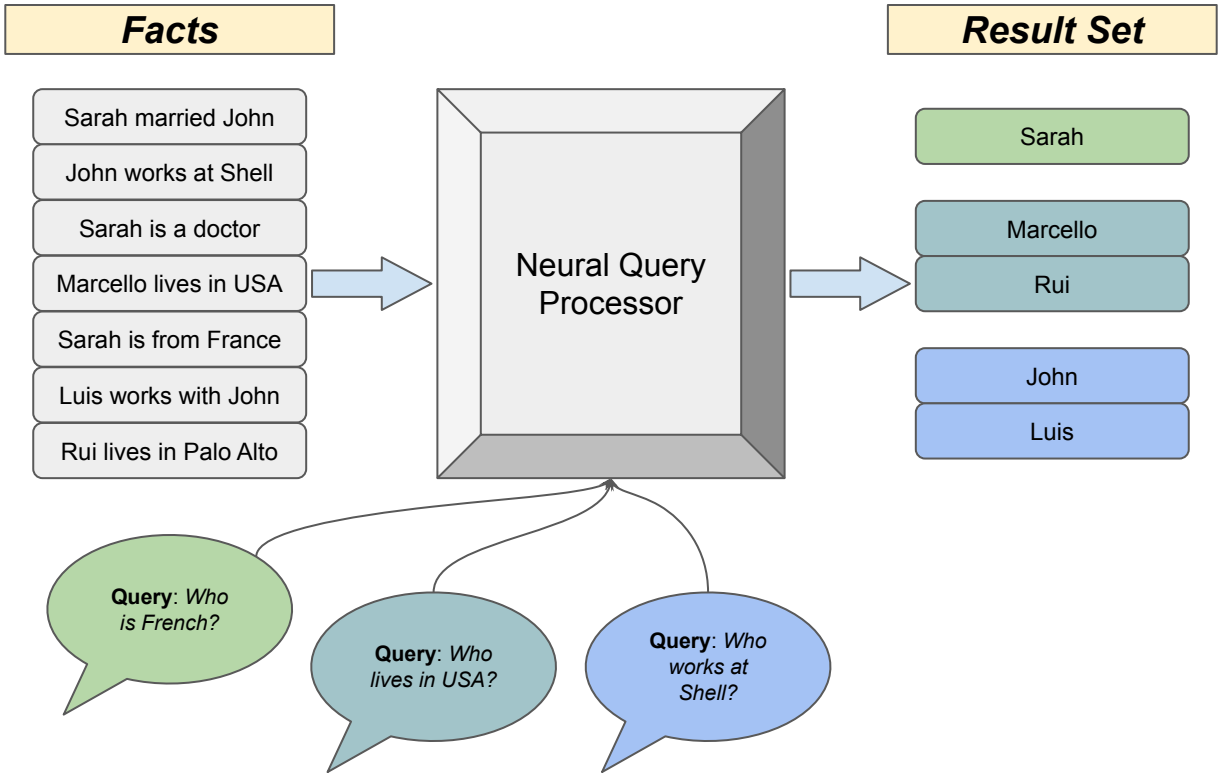
```
SELECT      p.PersonName
FROM        People p
JOIN        PeopleJobs pj
            ON (p.PersonID = pj.PersonID)
JOIN        Jobs j
            ON (pj.JobID = j.JobID)
WHERE       j.JobDescription = “Research Scientist”
```

Database’s Core Component: The Schema

> “ The database schema of a database is its structure described in a formal language supported by the database management system (DBMS). The term ‘schema’ refers to the organization of data as a blueprint of how the database is constructed (divided into database tables in the case of relational databases). ”

from Wikipedia

What if… We Removed Schema from Databases?



Is it QA?

- QA tasks deal with questions posed in natural language:
  - When were the Normans in Normandy?
  - Which kicker had most field goals?
  - Several datasets, i.e., SQuAD, DROP, MSMARCO-QA, etc.
- In QA typically the answer to a query is located within a passage or multiple passages that are (usually) locally available
  - In NeuralDBs facts that form a single result set might be scattered around in the dataset.
- In QA typically the answer is, well… “an answer” 😁 Typically a single sentence, e.g., “*Adam Vinatieri*”
  - In NeuralDBs we should target both sets of answers, and aggregations (count, avg, etc.).

An LLM Based Solution

- Transformers, e.g., BERT, have became ubiquitous in NLP.
- Introduced in the famous ‘Attention Is All You Need’ paper.
- It consists in applying (self-)attention to each token of a sequence of text, i.e., subwords.

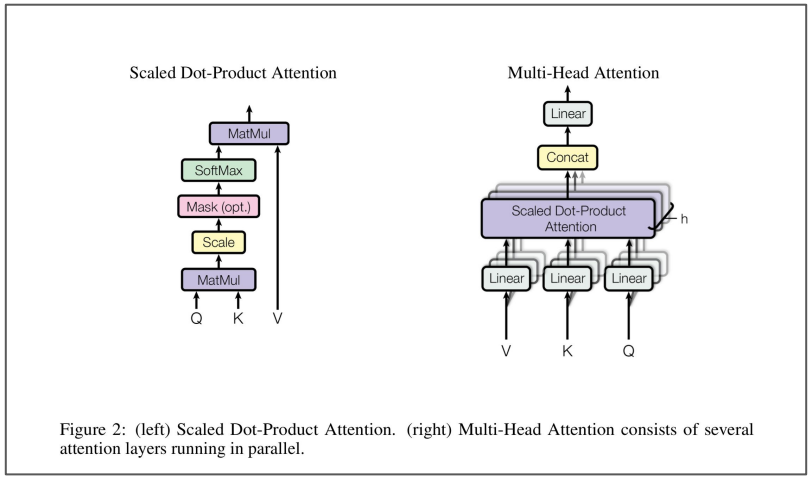
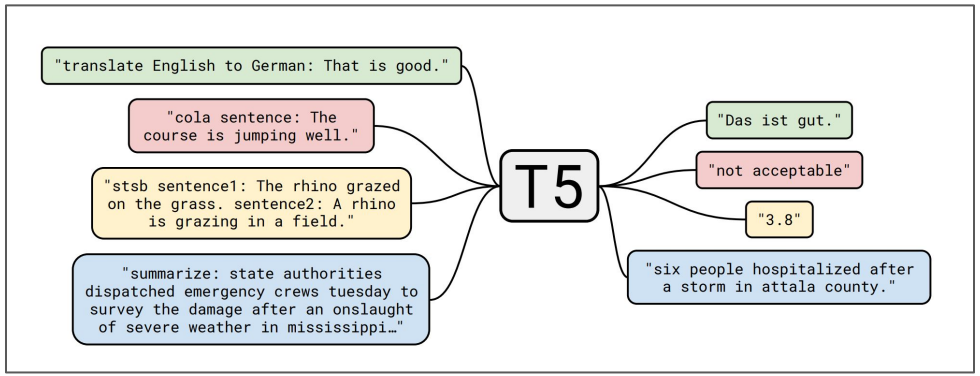


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

Credits: Attention Is All You Need' by Vaswani et al.

Google's T5 (**T**ext-**t**o-**T**ext **T**ransfer **T**ransformer) Model

Trained in a multi-tasking fashion on the following tasks: (i) *GLUE* and *SuperGLUE* meta-tasks; (ii) CNN/Daily Mail *Abstractive Summarization*; (iii) *SQuAD* Question Answering; and (iv) WMT English to German, French, and Romanian *Translation*



Credits: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. Raffel et al.

Neural Query Processing

- Task:
  - Given a query and a small number of facts from the database, can the T5 accurately answer queries that are posed in natural language, whose answer may require projection (i.e., extracting part of a sentence), join, and aggregation?
- Data: 7 different relationships from Wikidata extracted using a template
  - we generate a training, validation and held-out test set containing *535*, *50*, and *50* databases respectively
  - Each database contains *50* facts and has *100-200* QA pairs
  - In total: *60,000* training, *5,500* validation and *6,000* test instances
- Input: To provide input to the transformer, we jointly encode relevant facts from the database by concatenating them with the query (separated by a special delimiter token)

Some Example Queries

Facts: (8 of 500 shown)

- Nicholas lives in Washington D.C. with his wife.
- Sheryl is Nicholas's wife.
- Teuvo was born in 1912 in Ruskala.
- Sheryl's mother gave birth to her in 1978.
- Nicholas is a doctor.
- Sarah was born in Chicago in 1982.
- Sarah married John in 2010.
- Sarah works in a hospital in NY as a doctor.

Queries:

List everyone born before 1980.
(`Set`) → Sheryl, Teuvo, . . .

Whose spouse is a doctor?
(`Join`) → Sheryl, John, . . .

Who is the oldest person?
(`Max`) → Teuvo

Who is Sheryl's mother?
(`Set`) → NULL

How to build Facts and Queries?

- Training a NL database requires supervision in the form of *(D, Q, A)*:
  - *D* is a set of facts
  - *Q* is a query
  - *A* is the correct answer
- We generate training data in a controlled fashion by transforming structured data from Wikidata into NL facts and queries
- Pros:
  - Scale
  - Breadth

(**S**ubject, **R**elation, **O**bject)
(*Bezos*, *employedBy*, *Amazon*)

Facts

- We “verbalize” knowledge graph triples that are synthesized through a sequence to sequence model
  - Data is from KELM, we generate a rule-based post-hoc mapping back to Wikidata considering: string similarity, and compatibility of the generated triple

Input Triples	Target Sentence
Das Tagebuch der Anne Frank, (distributor, Universal Pictures), (country, Germany), (publication date, 03 March 2016)	The film was theatrically released in the Germany on March 3, 2016, by Universal Pictures International.
Neff Maiava, (date of birth, 01 May 1924), (date of death, 21 April 2018), (occupation, professional wrestler)	Maiava (May 1, 1924 April 21, 2018) was an American Samoan professional wrestler.
Barack Obama 2012 presidential campaign, (country, United States), (end time, 06 November 2012), (start time, 04 April 2011)	The 2012 reelection campaign of Barack Obama, the 44th President of the United States, was formally announced on April 4, 2011.
Blue whale (parent taxon, Balaenoptera)	The blue whale (Balaenoptera musculus) is a marine mammal belonging to the baleen whale suborder Mysticeti.

Agarwal, O., Ge, H., Shakeri, S. and Al-Rfou, R., 2020. Large Scale Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training *arXiv preprint arXiv:2010.12688*.

Queries

- We generate queries using a number of templates for each property and question type
  - *(X, bornIn, Y)*
  - *(X, employedBy, Y)*
- We also “simulate” joins by “chaining” triples:
  - (*Y, locatedIn, Z*) ⛓ (*X, employedBy, Y*) → “Does $X work at a company located in $Z?”
  - (*Y, marriedTo, Z*) ⛓ (*Z, leavesIn, Y*) → “Does $Z’s spouse leaves in $Y?”
  - (*Y, childOf, Z*) ⛓ (*Y, bornIn, X*) ∏ (*Y’, bornIn, X’*) → “Is $Z’s child younger than $Y’?”
  - (*Y, rel1, Z*) ⛓ (*Z, rel2, Y*) → “Does $Z’s **rel1** also **rel2** $Y?”

Example Queries, Supporting Facts, and Correct Answer

Example: `Set`	
Question	Who studied at University of Minnesota?
Supporting Facts	**1.** [John B Totushek was born on 7 September 1944 in Minneapolis. He attended the University of Minnesota and became a US Naval Aviator. Mr. Totushek was also a human being.] **2.** [Melvin Maas graduated from the University of Minnesota and is buried at Arlington National Cemetery. He is a native of Minnesota and his language is English.] **3.** [Clarence Larson graduated from the University of Minnesota and is a member of the National Academy of Engineering.] **4.** [Ted Mann, who is the surname of Ted Mann, attended Duke University and the University of Minnesota. He is a human being.]
Answer	[John B. Totushek, Ted Mann, Clarence Larson, Melvin Maas]

Example Queries, Supporting Facts, and Correct Answer

Example: `count`	
Question	How many people work for Yale Law School?
Supporting Facts	**1.** [Michael Ponsor, born in Oxford, graduated from Pembroke College in Oxford. He was awarded the Rhodes Scholarship and is an employee at Yale Law School. He is an expert in the field of human rights.] **2.** [Stephen Wizner is an American legal scholar who graduated from Dartmouth College and is a graduate of the University of Chicago Law School. He works at Yale Law School.]
Answer	2

Example Queries, Supporting Facts, and Correct Answer

Example: `Min/Max`	
Question	What is the largest yearly attendance?
Supporting Facts	**1.** [The musee en herbe has a visitor per year of] 70000. **2.** [The total number of visitors to the Hirschsprung Collection is 71779 per year.] ... **24.** [The Tate Modern has a visitor count of 5839197 visitors per year.] **25.** [Catoctin Mountain Park attracts 221750 visitors per year.]
Answer	5839197

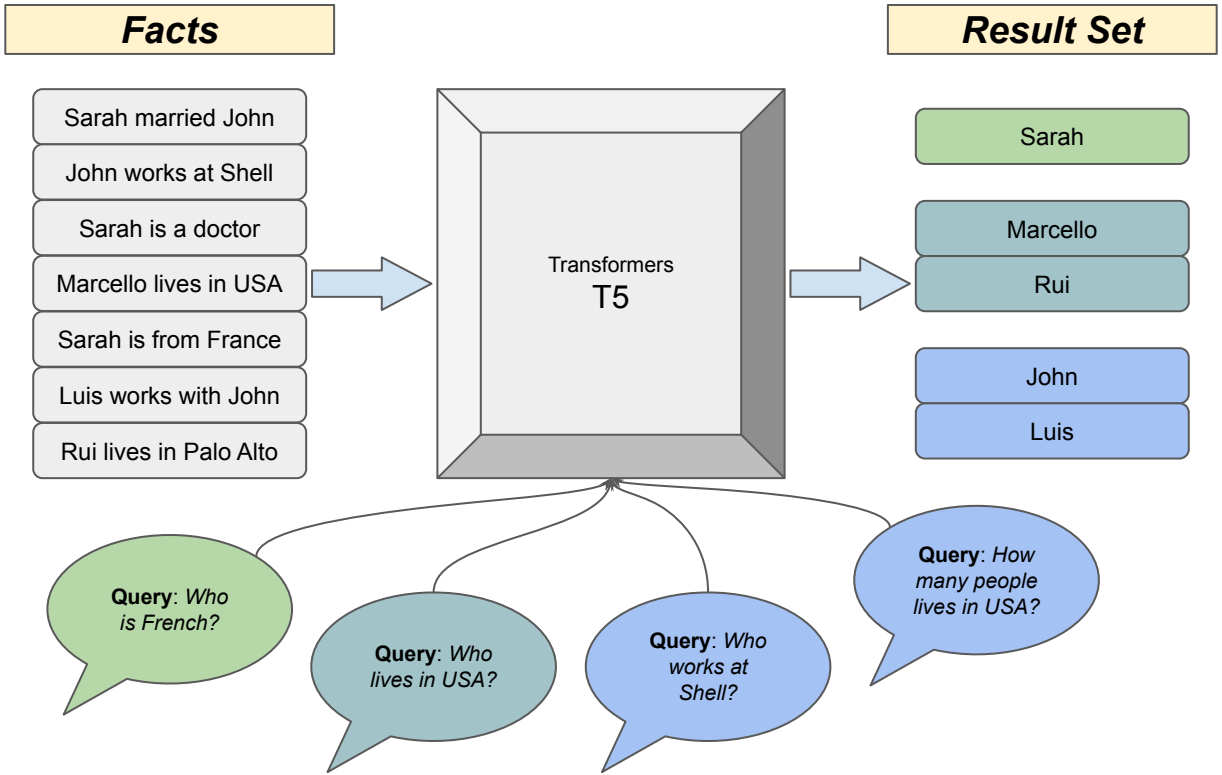
Example Queries, Supporting Facts, and Correct Answer

Example: `Bool`	
Question	Is North Carolina State University the employer of Wes Moore?
Supporting Facts	**1.** [Wes Moore is a human being who is employed at Francis Marion University and is a basketball player for North Carolina State University.]
Answer	TRUE

Example Queries, Supporting Facts, and Correct Answer

Example: `Join`	
Question	Who plays for a team in Ligue 1?
Supporting Facts	**1.** [Thomas Allofs started his career in 1989 with RC Strasbourg Alsace. He finished his career in 1990., RC Strasbourg Alsace is an association football club in the Ligue 1 league. It was founded in 1906 and is located in Strasbourg, France.]
Answer	[Thomas Allofs]

A “Simple” Solution



Possible Issues

✅ When fed with relevant facts from the database, T5 can produce results with reasonable accuracy

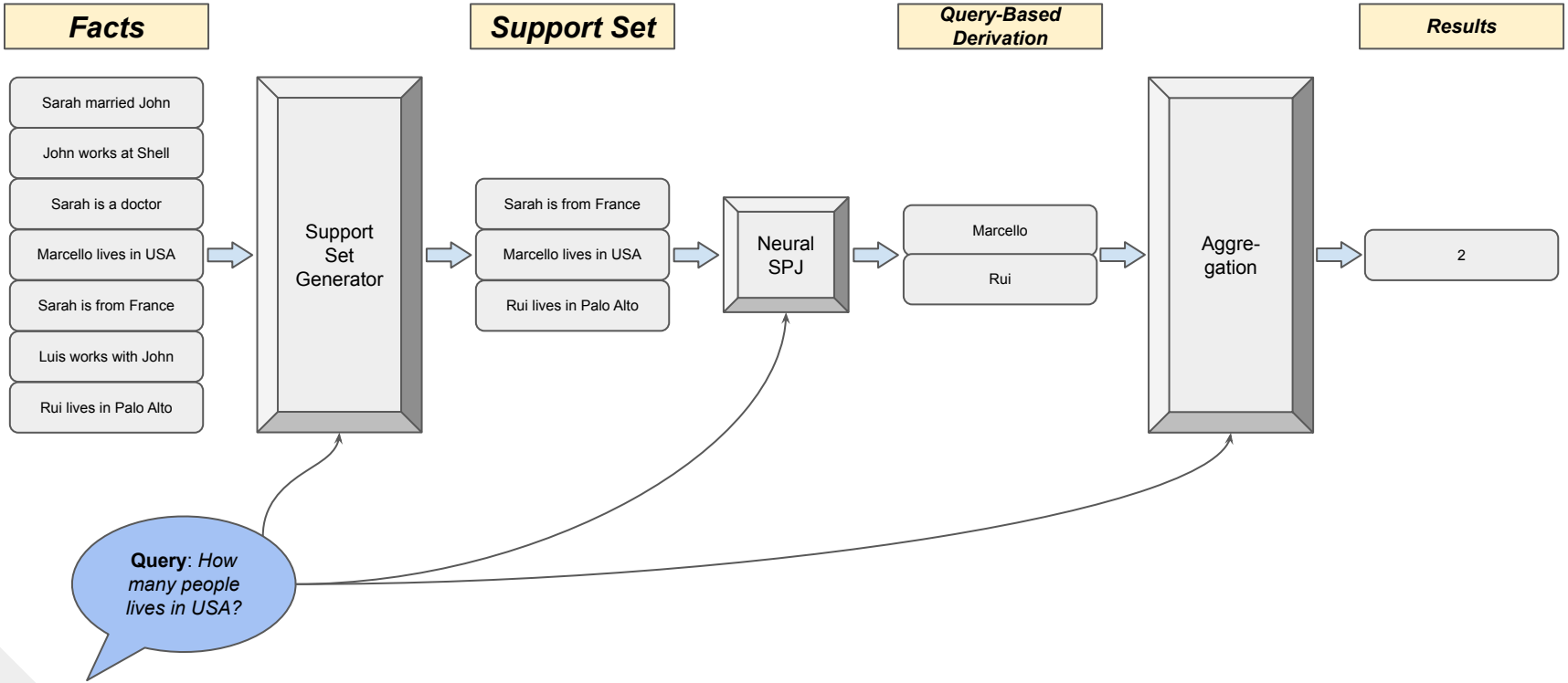
❌ Aggregation queries need to be performed outside of the neural machinery

👉 In order to handle queries that result in sets of answers and in order to prepare sets for subsequent aggregation operators, we need to develop a neural operator that can process individual (or small sets of) facts in isolation and whose results outputted as the answer or fed into a traditional (i.e. non-neural) aggregation operator.

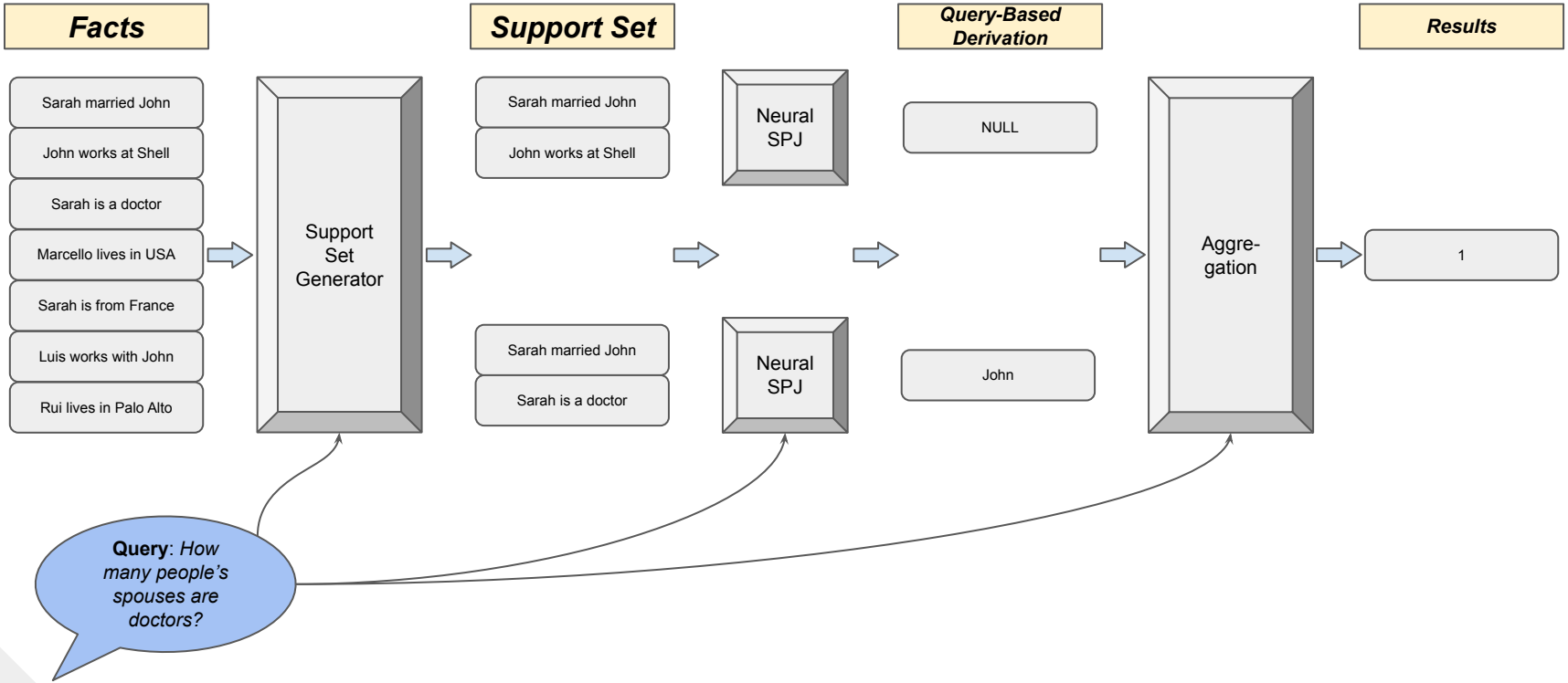
Challenges

- **Scale**
  - neural reasoning to databases of non-trivial size
    - In open-domain QA we usually complement the transformer reasoning with an IR component that extracts a small subset of the facts from the corpus
- **Multiple answer spans**
  - NDB might need to generate 100K facts and aggregate over them
- **Locality and Document Structure**
  - Answers might be dependent on several facts scattered across the DB
- **Multi-hop and Conditioned Retrieval**
  - E.g., *Whose spouse is a doctor?*

The Neural DB Architecture

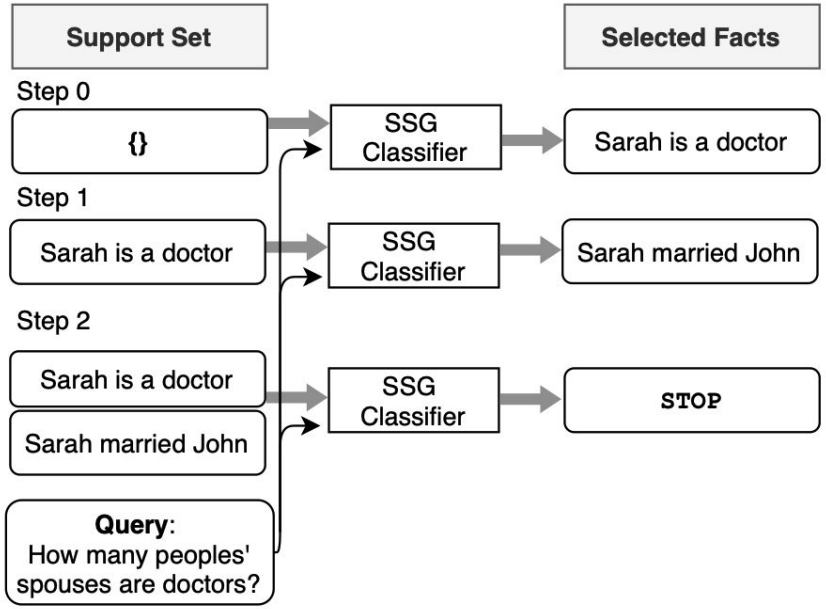


The Neural DB Architecture



Support Set Generator (SSG)

- Simple queries over single facts → TF-IDF based IR
  - not scalable for joins, aggregation queries or for queries outputting a set of answers as generating relevant sets requires incremental decoding, conditioning on already retrieved facts.



Neural Select-Project-Join (SPJ)

- For support sets that are insufficient to answer a question, the operator should return no output.
- For queries that require short chains of reasoning over multiple facts, the SPJ operator joins the facts when generating the output.
- SPJ generates a projection of the fact to a machine readable format dependent on the task, query and fact.
- Depending on the query type:
  - **Boolean Answers** → binary value
  - **Count/Set Queries** → entities
  - **Argmin/max operators** → key-value pairs.
    - For example "*Which place has the highest yearly number of visitors?*" has the projection of the form: (place,number of visitors).

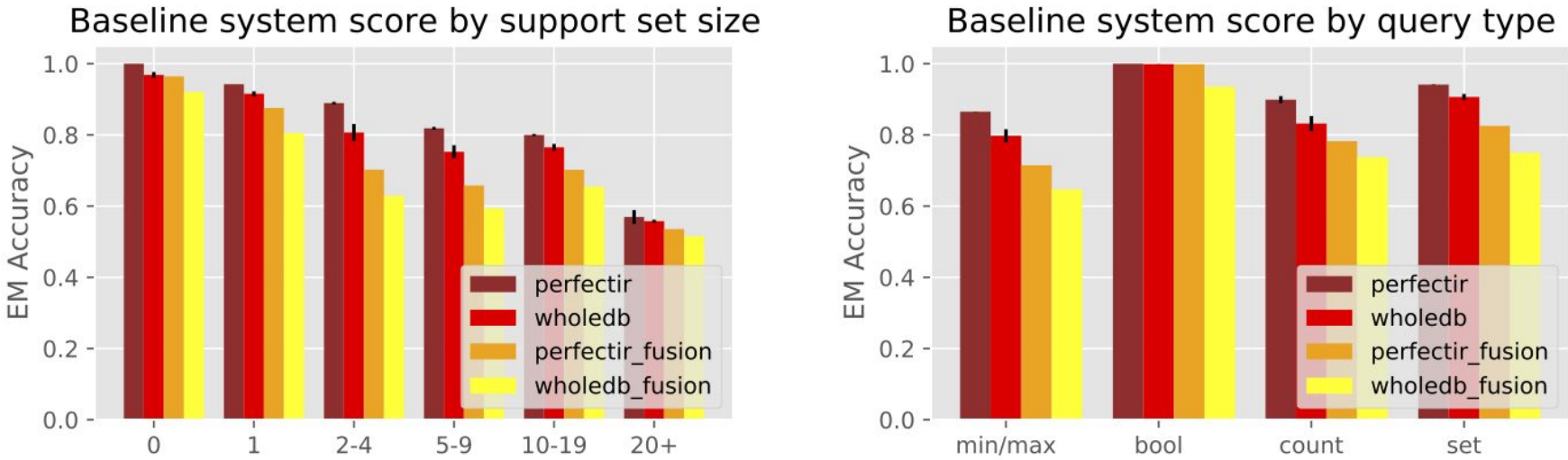
Examples of Neural SPJ Outputs

- Query: Does Nicholas's spouse live in Washington D.C.?
  - {*Nicholas lives in Washington D.C. with Sheryl.*, *Sheryl is Nicholas's spouse.*} → TRUE
- Query: Who is the oldest person in the database?
  - {*Teuvo was born in 1912.*} → (Teuvo, 1912)
- Query: Does Nicholas's spouse live in Washington D.C.?
  - {*Teuvo was born in 1912.*} → NULL

Results: SPJ Performance

Method	Exact Match (%)				
	Count	Min/Max	Sets	Atomic	Joins
NeuralDB	**79.45**	**100.00**	**91.91**	**97.90**	**79.29**
TF·IDF+T5	31.06	0.00	44.25	98.05	68.02
DPR+T5	38.07	21.19	54.55	97.38	58.64

Exact Match



Exact match accuracy for different classes of queries for a transformer model encoding up to 25 facts in one input. The results show that the model obtains high accuracy for queries performing Boolean inference, but falls short for queries with aggregation or yielding set answers (top) over multiple support sets (bottom).

Results: SPJ Performance

Method	Exact Match (%)			
	Min/Max	Bool	Count	Set
SPJ PerfectIR	88.3	99.8	90.1	89.4
SSG + SPJ	87.3	99.8	90.1	89.6

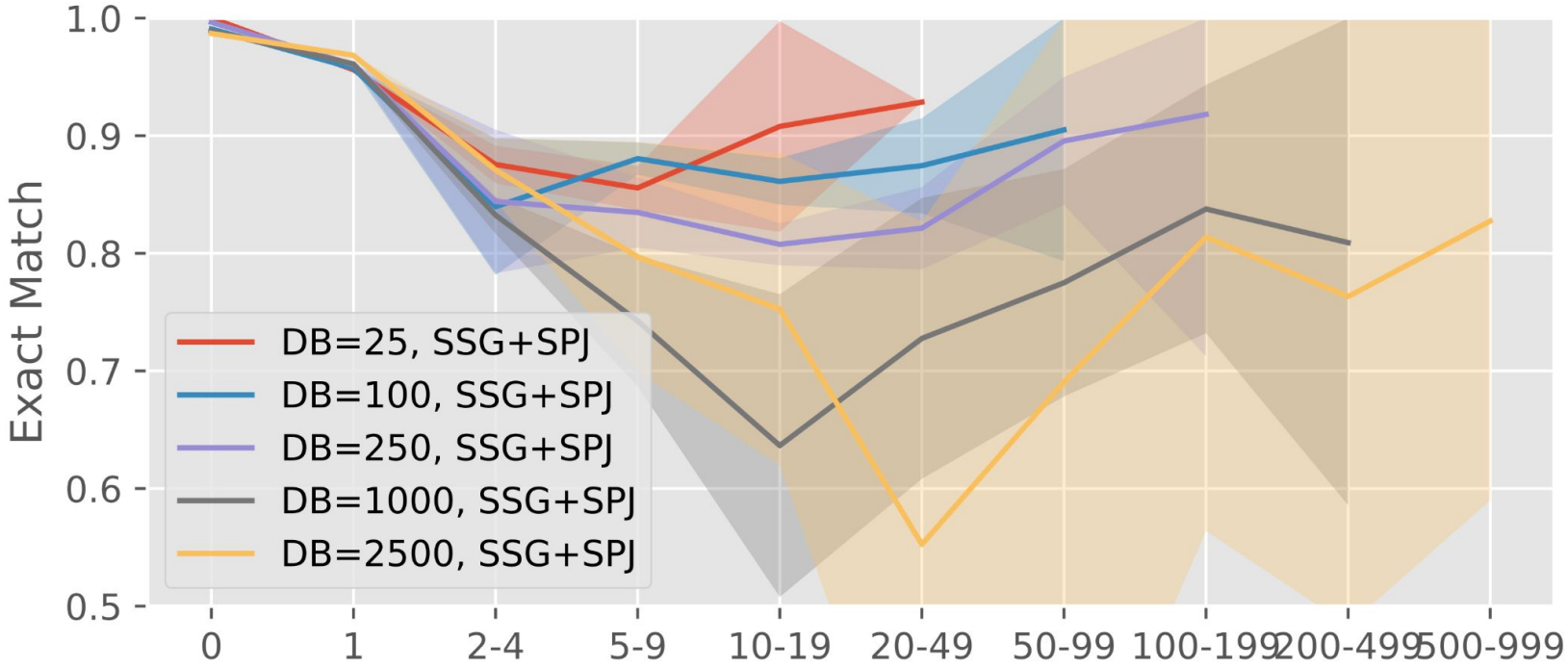
Using retrieved evidence achieves results competitive to the PerfectIR

Results: SSG Precision/Recall

Query Type	Exact Match (%)		Soft Match (%)	
	Precision	Recall	Precision	Recall
Boolean	85.12	94.04	85.39	94.04
Set	61.05	94.58	61.33	94.58
Count	57.88	96.15	58.00	96.15
Min/Max	60.68	95.82	60.68	95.82
Join	38.33	75.39	42.74	75.43
Average	57.08	90.53	58.24	90.54

Precision and recall of supervised SSG w.r.t. the reference set. Note that errors in retrieval do not necessarily translate to wrong query answers because the SPJ operator is trained to be robust to noise.

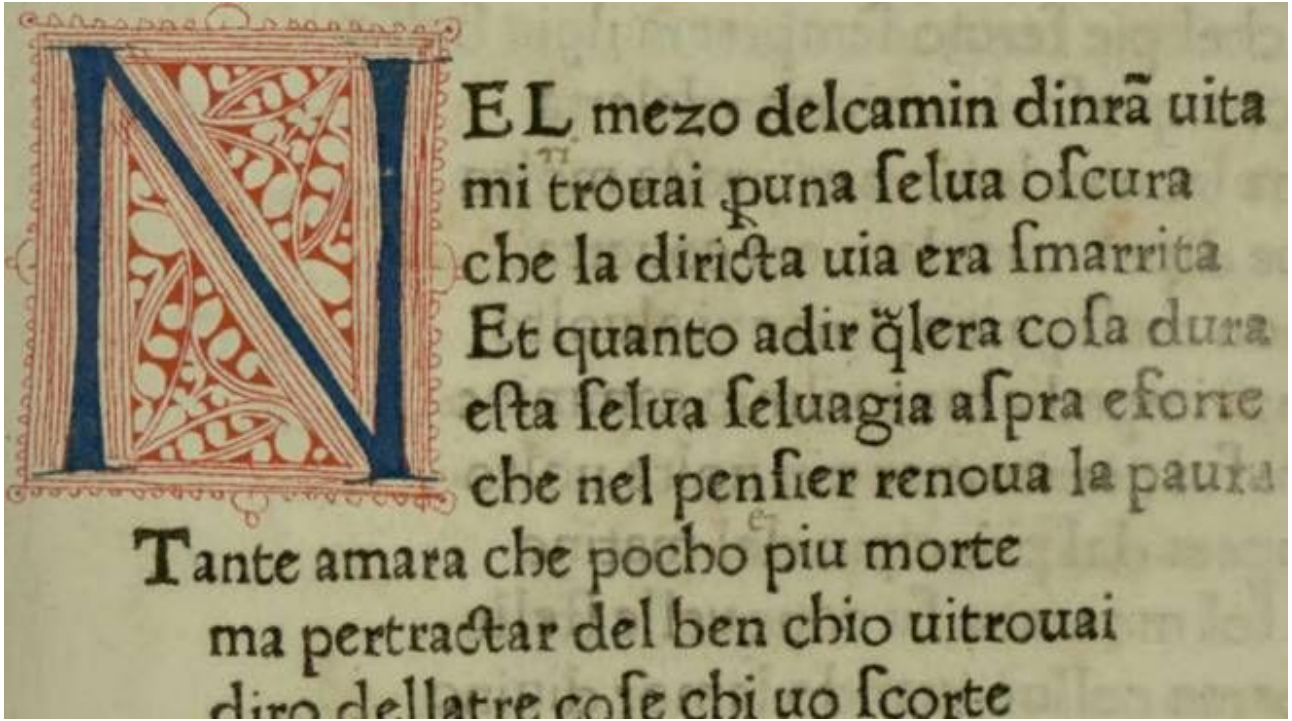
Results: SSG + SPJ Accuracy (different DB Sizes)



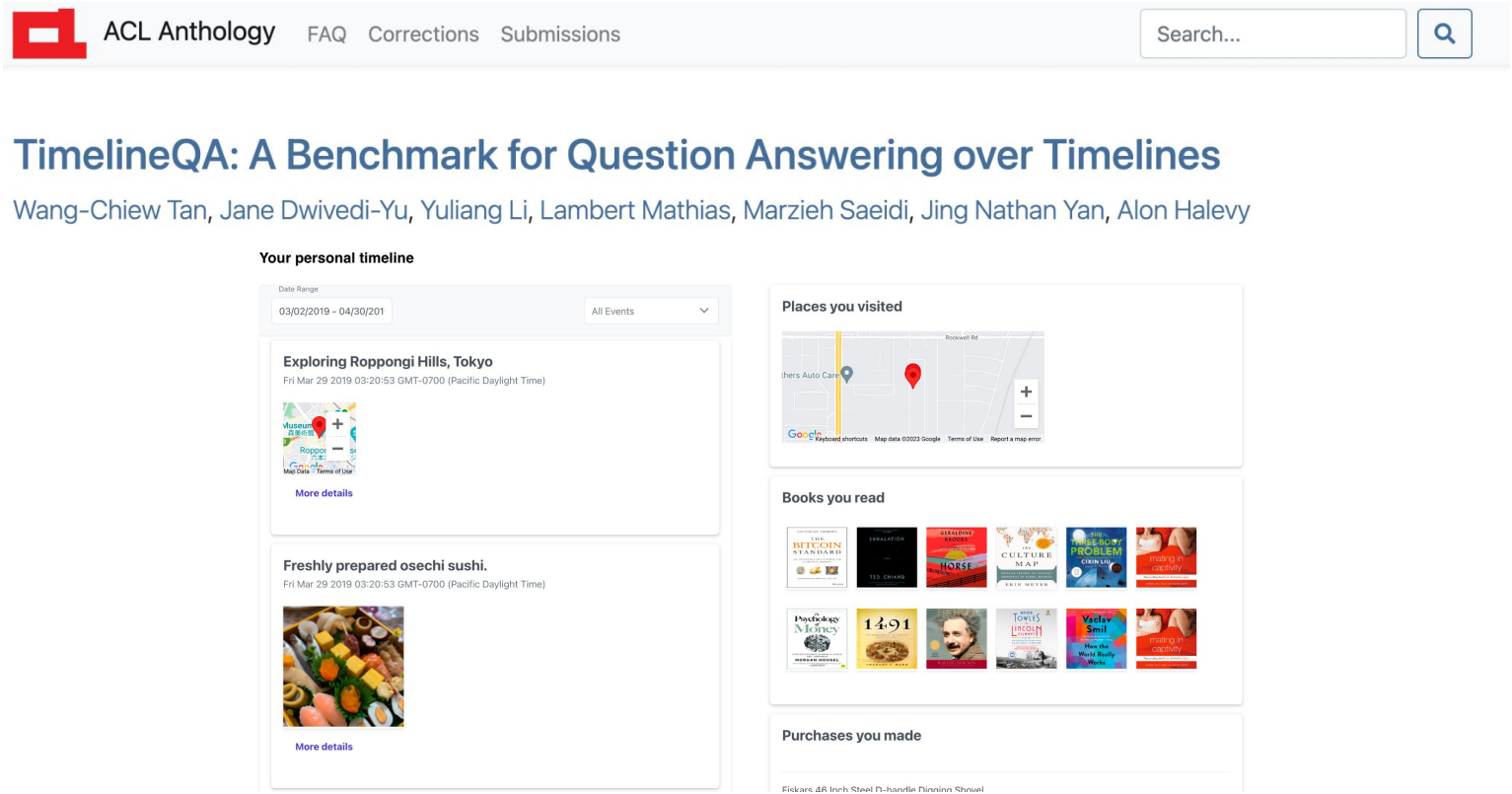
SSG+SPJ by support set size for all 5 databases. The SPJ is trained on databases of 25 facts and tested on larger databases. Low recall from SSG reduced answer EM for DBs of more than 1000 facts.

Neural Databases: Moving Beyond Text

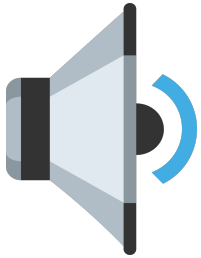
Moving Beyond Text

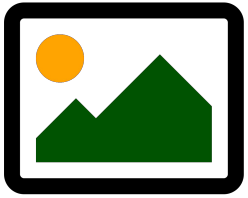


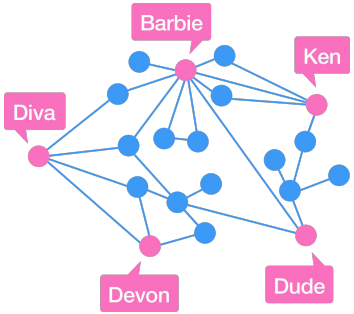
Moving Beyond Text



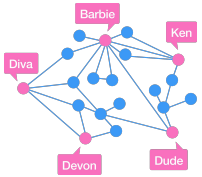
Multimodality







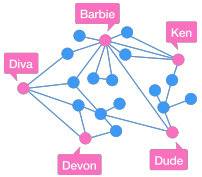
Multimodal NDB → The Vision



Multimodal Database

Multimodal NDB → The Vision

Which of the **restaurants** that I have seen today makes good **pasta** and play **jazz** music?

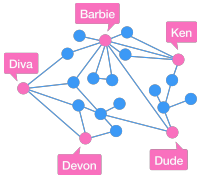


Multimodal Database

Multimodal NDB → The Vision

Which of the **restaurants** that I have seen today makes good **pasta** and play **jazz** music?

Multimodal Neural Databases

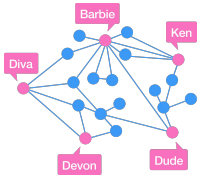


Multimodal Database

Multimodal NDB → The Vision

Which of the **restaurants** that I have seen today makes good **pasta** and play **jazz** music?

Multimodal Neural Databases



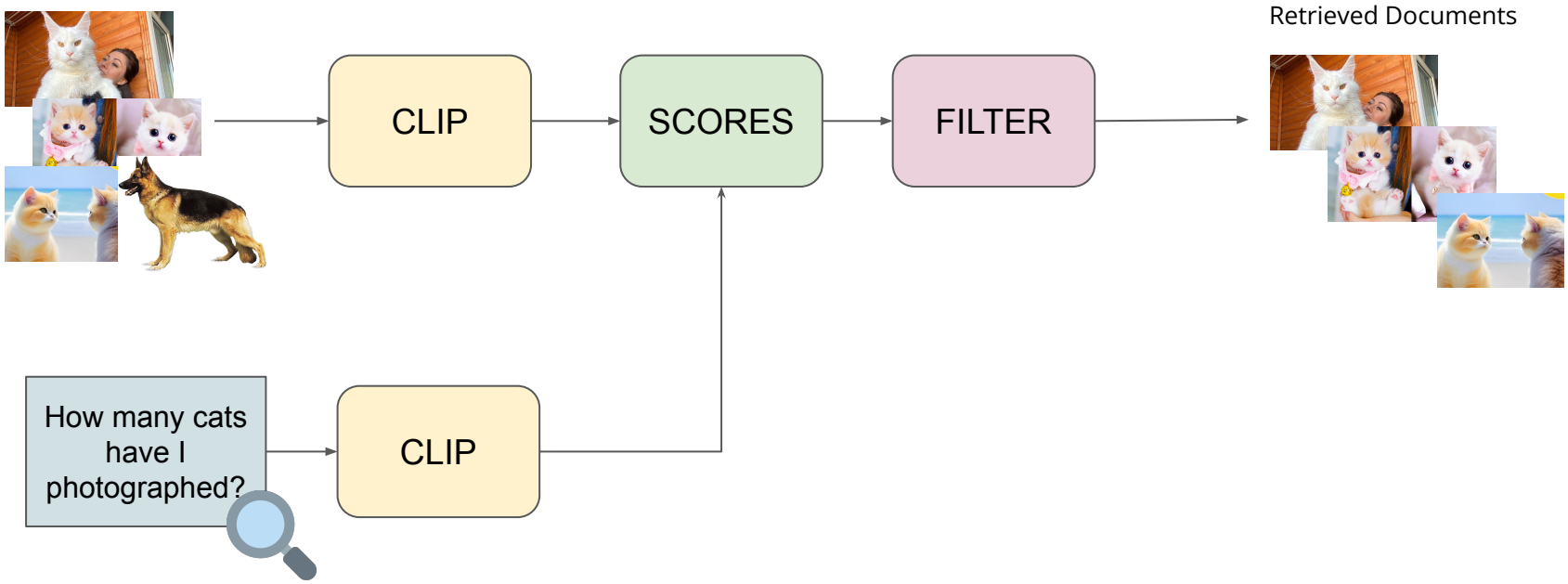
Multimodal Database

Pizzeria Le due Sorelle

John's Pasta

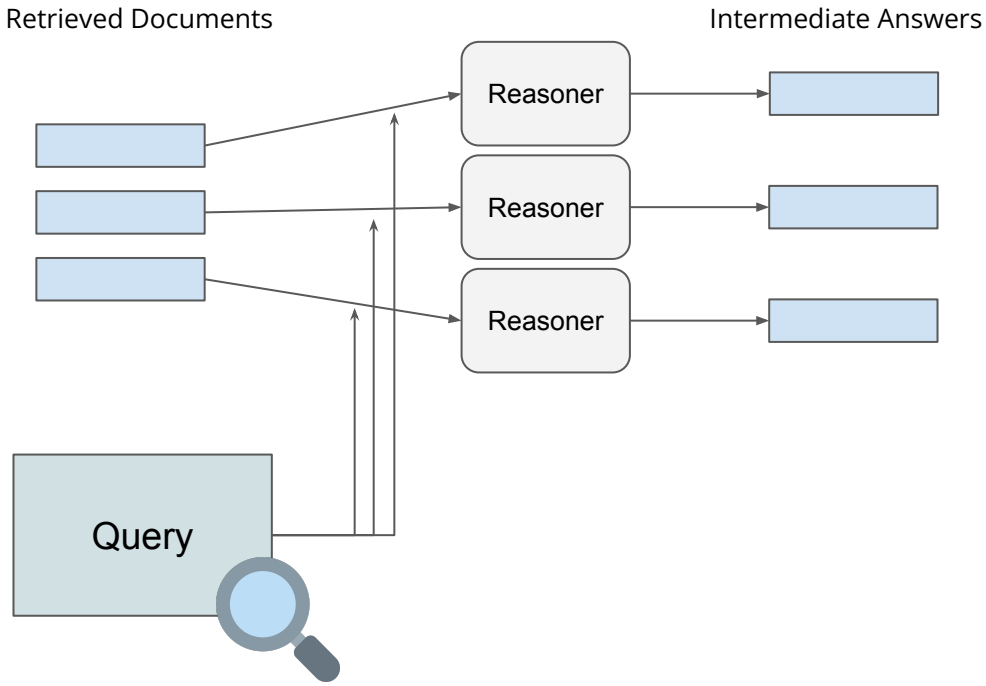
Answer Set(s)

Proposed Architecture

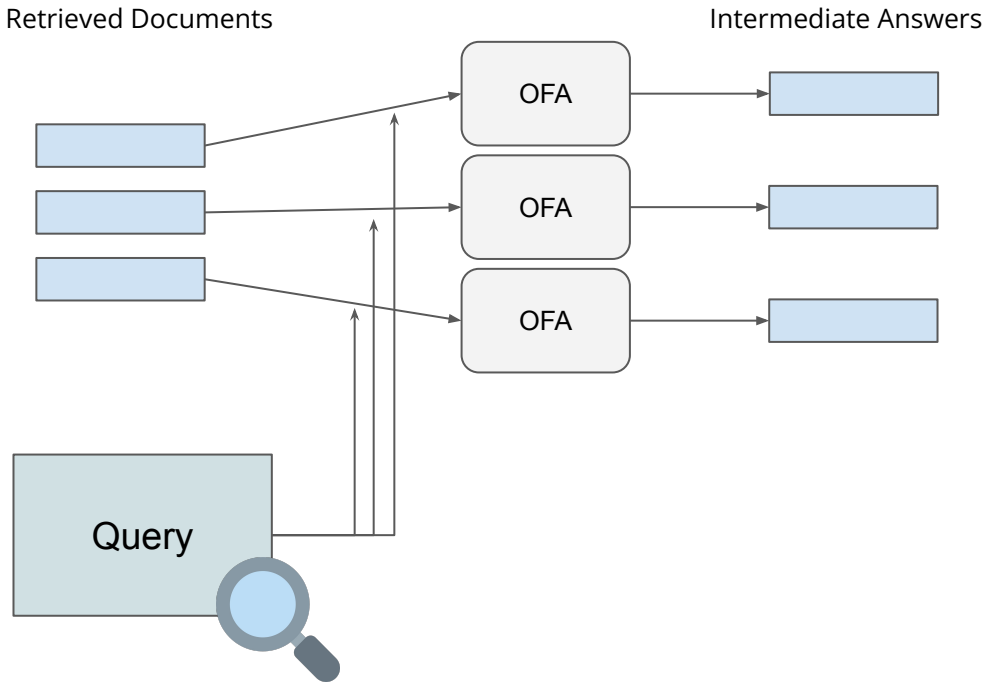


Alec Radford, et al. "Learning Transferable Visual Models From Natural Language Supervision." *International Conference on Machine Learning*. PMLR, 2021.

Proposed Architecture

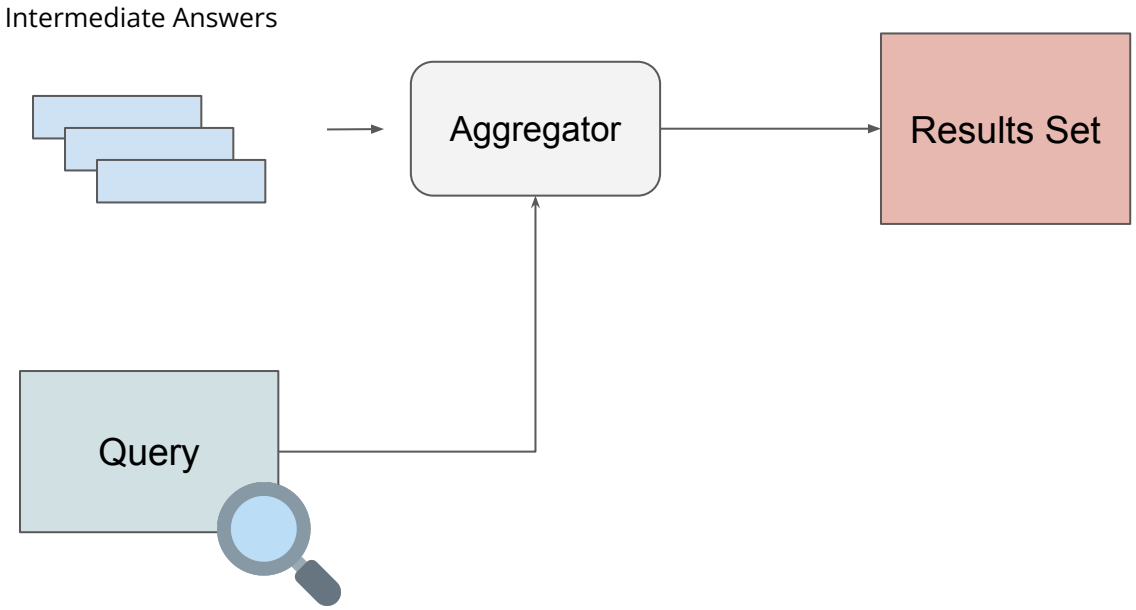


Proposed Architecture

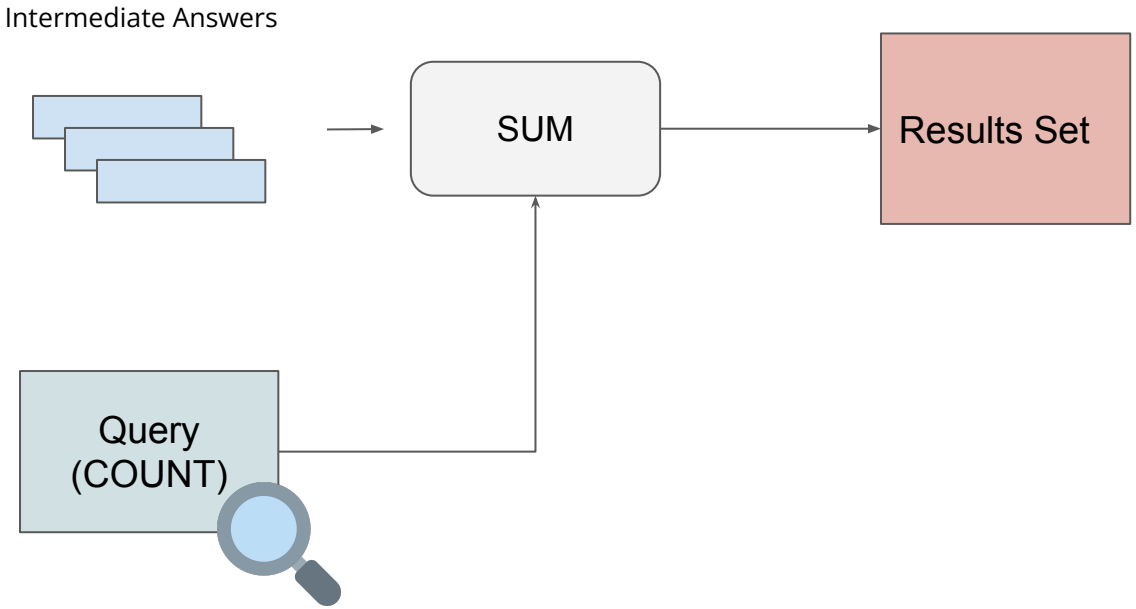


Wang, Peng, et al. "Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework." *International Conference on Machine Learning*. PMLR, 2022.

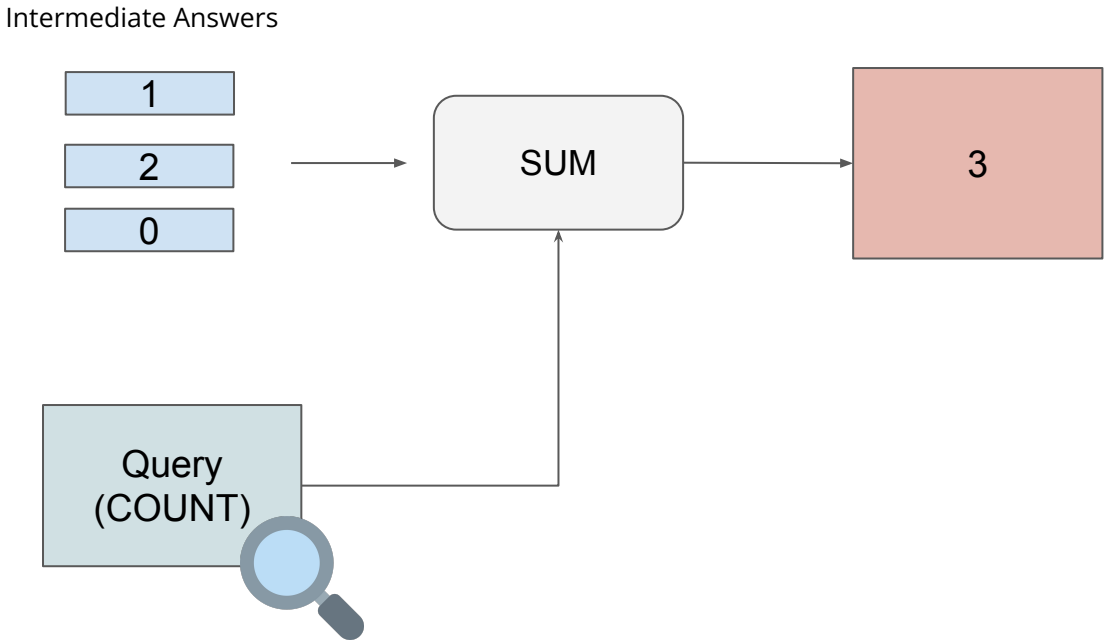
Proposed Architecture



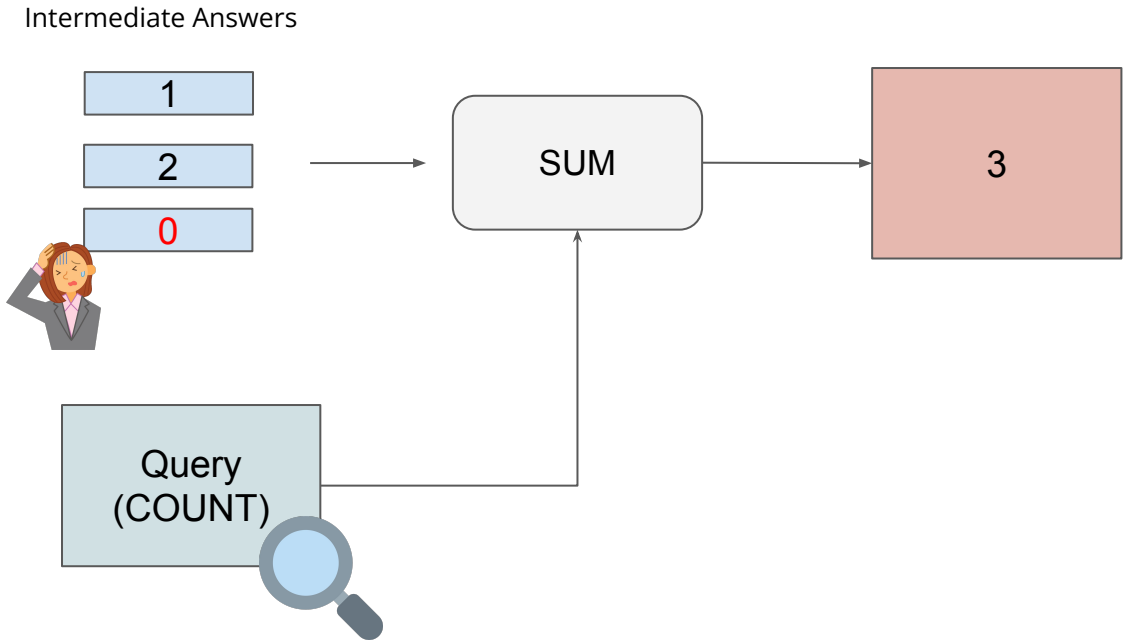
Proposed Architecture



Proposed Architecture



Proposed Architecture



Example Results on Count Queries

	Total Error ↓			
Stock	Error	Error TP	Error FP	Error FN
Perfect IR	**0.46 ± 0.07**	0.46 ± 0.07	N/A	N/A
Noisy IR	0.77 ± 0.16	0.46 ± 0.07	**0.31 ± 0.15**	N/A
Dmg. IR	1.24 ± 0.32	0.46 ± 0.07	0.78 ± 0.32	N/A
Full	1.27 ± 0.17	**0.42 ± 0.07**	0.76 ± 0.13	0.09 ± 0.02
FTmodel				
Perfect IR	**0.14 ± 0.01**	0.14 ± 0.01	N/A	N/A
Noisy IR	0.22 ± 0.01	0.14 ± 0.01	0.08 ± 0.01	N/A
Dmg. IR	0.54 ± 0.05	0.14 ± 0.01	0.40 ± 0.05	N/A
Full	0.99 ± 0.06	**0.11 ± 0.01**	0.79 ± 0.06	0.09 ± 0.02

More details can be found in the following papers…

1. Giovanni Trappolini, Andrea Santilli, Emanuele Rodolà, Alon Y. Halevy, Fabrizio Silvestri: **Multimodal Neural Databases**. *SIGIR 2023*
2. Artsiom Sauchuk, James Thorne, Alon Y. Halevy, Nicola Tonellotto, Fabrizio Silvestri: **On the Role of Relevance in Natural Language Processing Tasks**. *SIGIR 2022*
3. James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, Alon Y. Halevy: **From Natural Language Processing to Neural Databases**. *VLDB 2021*
4. James Thorne, Majid Yazdani, Marzieh Saeidi, Fabrizio Silvestri, Sebastian Riedel, Alon Y. Halevy: **Database reasoning over text**. *ACL/IJCNLP 2021*

Data Mining and Language Technology

End of Lecture

08 - Transformers

SAPIENZA
UNIVERSITÀ DI ROMA

Fabrizio Silvestri