Introduction to Transformers

Andrea Zugarini

Universities of Florence and Siena, SAILab

Table of contents

Introduction

Introduction

In the last decade most of the advances in NLP were achieved by learning textual representations with self-supervised **Language Modeling**-related tasks on large corpora.

Recently, **transformers** pushed this principle further, with dataset and model sizes that have grown by orders of magnitude.

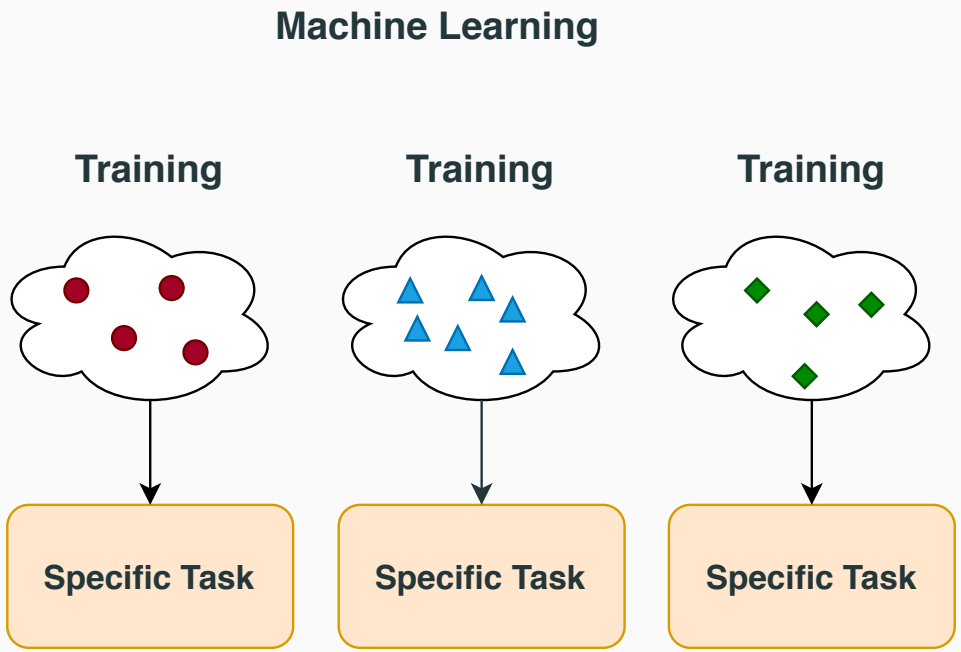
Learning Language Representations

Machine Learning

Given a collection of related problems, in standard ML each one is addressed separately.

However, there may be **shared features** useful for several tasks.

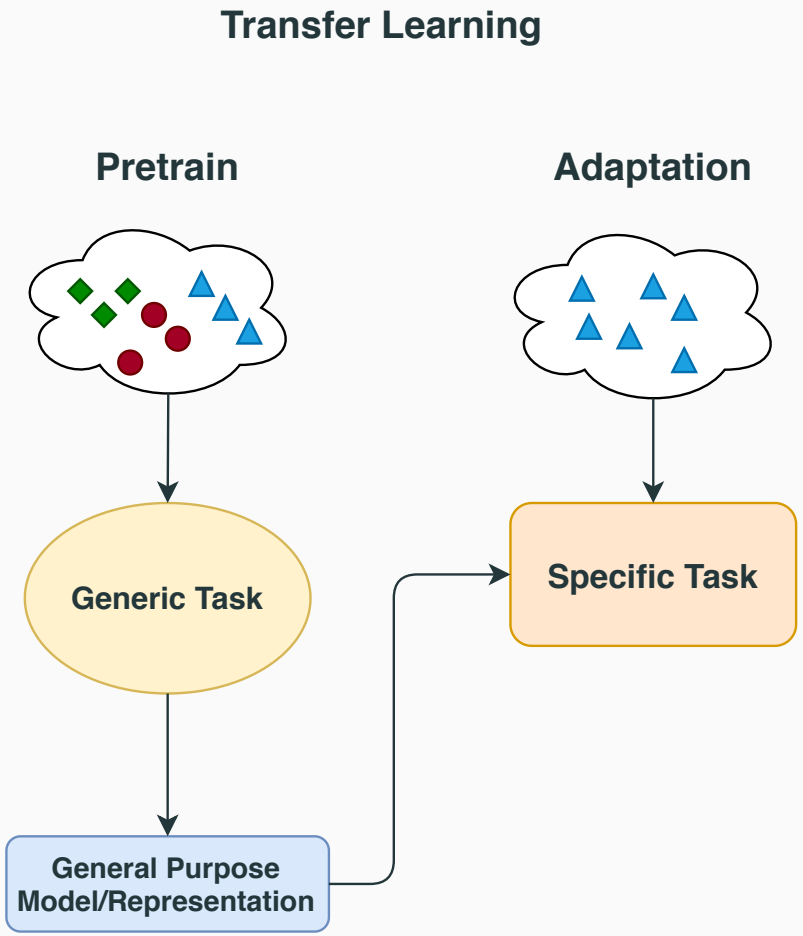
Why not instead, learning **general purpose representations** first and **adapt** them later?



Transfer Learning

General purpose representations are typically learnt from a more generic task.

Pretraining is usually on **more data**.



Language Modeling (1)

Language Modeling Tasks are powerful **general purpose** problems to learn representations of texts.

Language Modelling is the problem of estimating the probability distribution of text:

$$p(w_1, \cdots, w_n) = \prod_{i=1}^{n} p(w_i | w_{<i})$$

where $w_1, \ldots, w_n$ are the words (or other tokens) of the text.

Language Modeling (2)

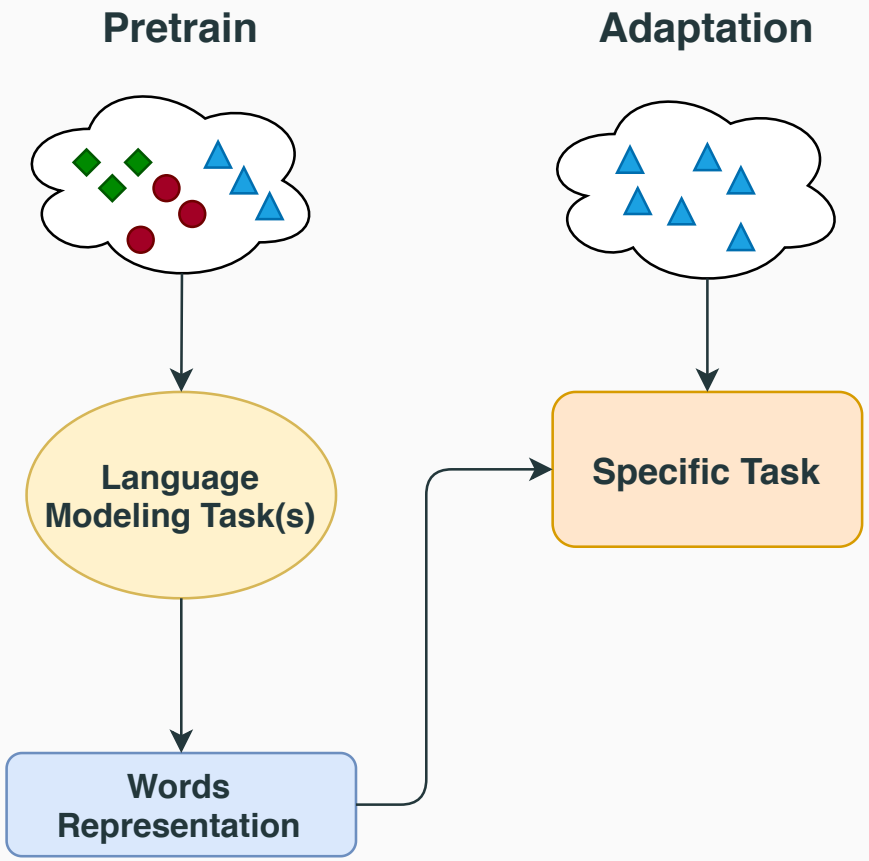
Advantages:

- A strong language model learns **syntactic** and **semantic** information that is paramount for any NLP task.
- It is a **self-supervised** task, not requiring any annotated data.
- We are plenty of data: web is a **huge** collection of textual corpora.

Word Representations

At the beginning the focus was on learning **representation of words**. Examples:

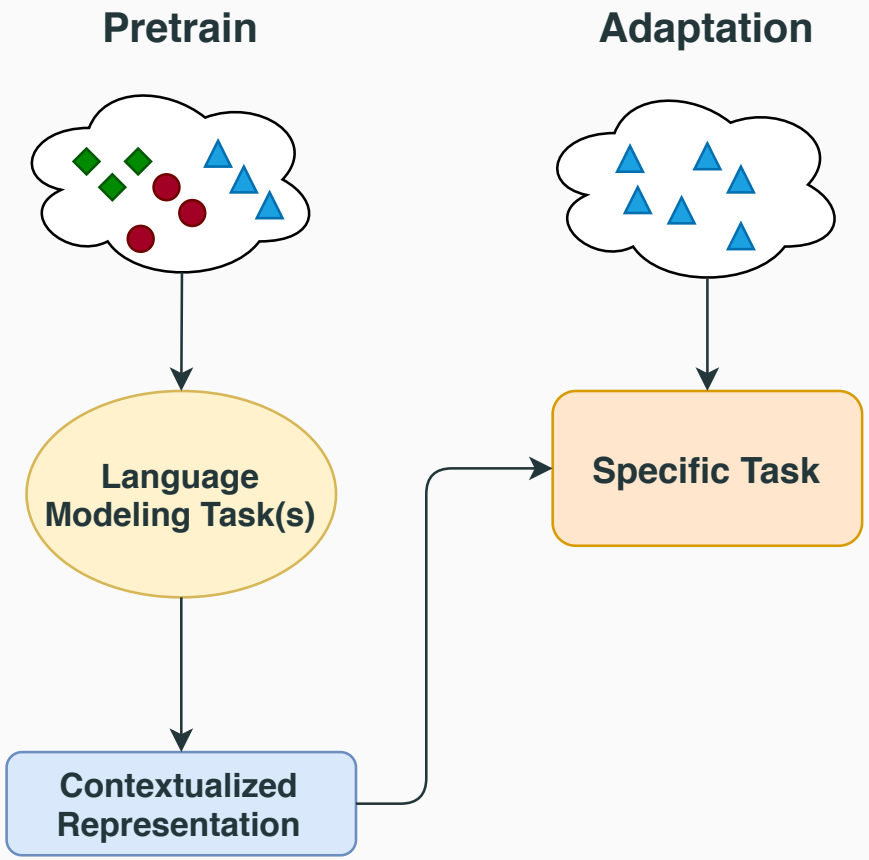
- Word2Vec
- GloVe



Contextualized Representations

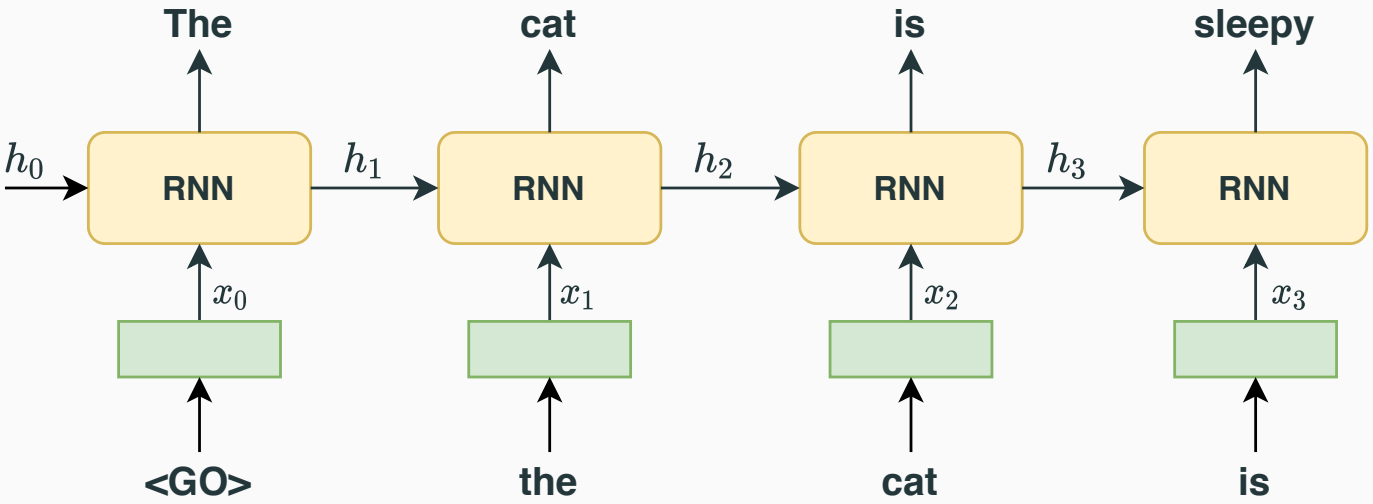
But most of the information is actually in the whole **context**.
Examples:

- Context2Vec
- ELMo



The learning of **contextualized** representations required the use of Recurrent Neural Networks.

Recurrent Neural Networks (1)

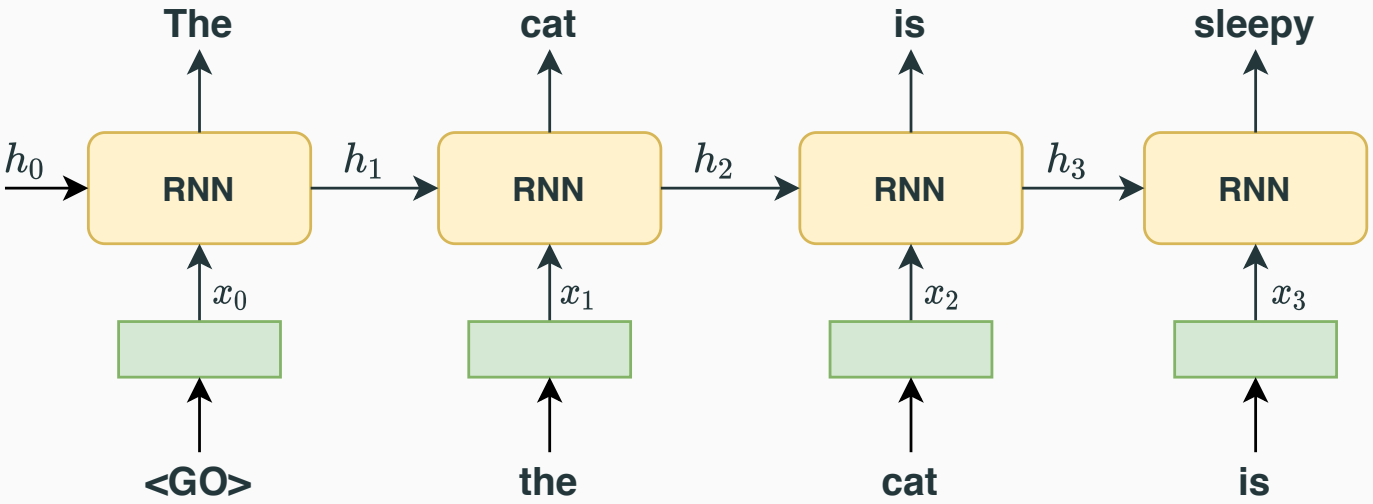


An RNN is a function f defined as follows:

$$h_t = f(x_t, h_{t-1}), \forall t > 0$$

and h_0 is the zero vector.

Recurrent Neural Networks (2)



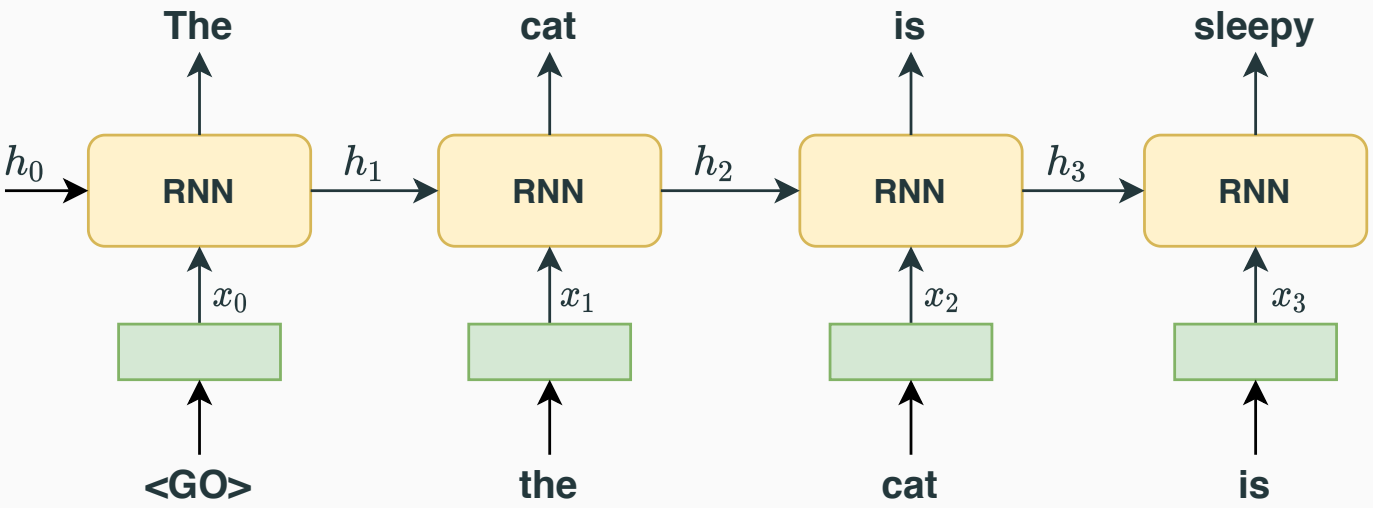
An RNN is a non-linear function f defined as follows:

$$h_t = f(x_t, h_{t-1}), \forall t > 0$$

In the simplest case:

$$h_t = f(x_t, h_{t-1}) = \sigma(Wx_t + Uh_{t-1})$$

Recurrent Neural Networks (3)



Limitations

- RNNs suffers of the **long-term dependencies** problem: intuitively, they struggle in remembering **old past** information.
- Computation is sequential, hence its complexity is **linear** wrt the **sequence length**, i.e. $O(n)$.

Transformers

Attention is all you need

Originally proposed in [7] for Machine Translation.

Designed to overcome the limits of recurrent neural networks.

Recurrent cells are replaced with **attention mechanisms**.

The computation of hidden states is not **sequential**, therefore parallelizable $O(1)$:

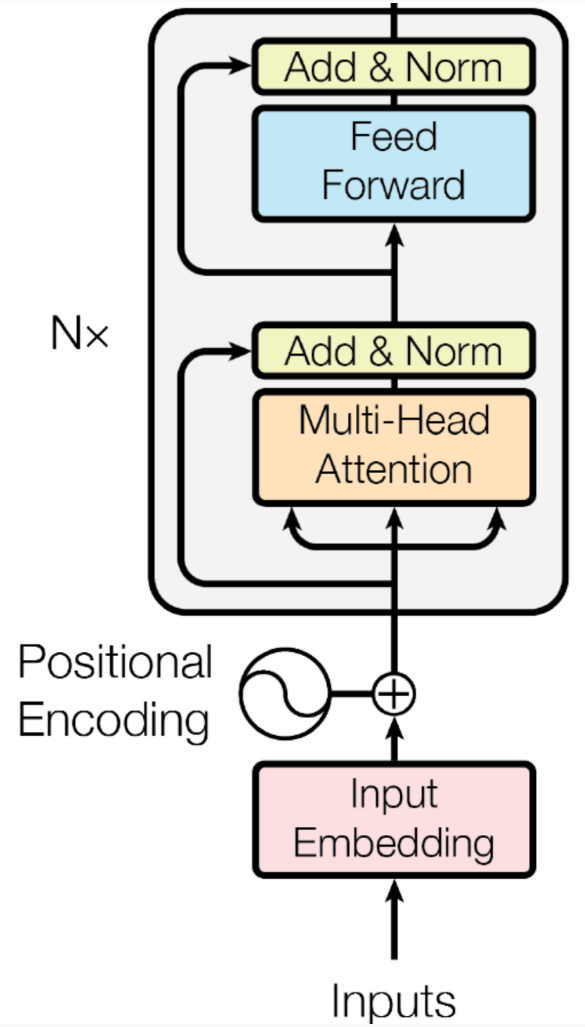
$$h_t = f(x_1, \ldots, x_t)$$

h_t is a **contextualized** representation of the t**-th token**.

Lets describe the architecture.

Architecture

- A stack of *N* (e.g. 12 in **BERT**) identical layers.
- Each Layer has two sub-layer, **multi-head** attention and a feed-forward network.
- Residual Connections and Layer Normalization after each sub-layer.
- **Position Encodings** inform the model of the sequential order of the **input** tokens.



Positional Encodings cope with the **lack of recursion** within the model.

They inject **relative** or absolute position of a token by adding a **position-wise** embeddings to the input embeddings.

In the original paper, position encodings were cosine and sine functions of the token position.

Most models now use learnable embeddings associated to the relative position of a token.

Inputs: Keys K, Query Q, Values V. Keys and Values are the same, the output of the previous layer, Q is the current token.

E.g. in first layer the attention at token t is: $K = V = (x_0, \ldots, x_n)$ and $Q = x_t$.

Linear Transformation:

$$Q = QW^Q, K = KW^K, V = VW^V$$

where $W^Q \in \mathbb{R}^{d_{model} \times d_q}, W^K \in \mathbb{R}^{d_{model} \times d_k}, W^V \in \mathbb{R}^{d_{model} \times d_v}$

Scaled dot-product Attention:

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

where d_k is the dimension of keys.

Attention acts as a **weighting average** of the Values.

Instead of performing one single attention, there are h attention functions on the same Q, K, V:

$$o = MultiHead(Q, K, V) = Concat(head_1, \cdots, head_h)W^O,$$
$$head_i = Attention(Q, K, V)$$

Multi-head attention allows the model to jointly attend to information from different positions.

Architecture

Feed-forward Layer

The final hidden state h_t for token t is obtained from an MLP.

$$h_t = MLP(o)$$

Pre-training (1)

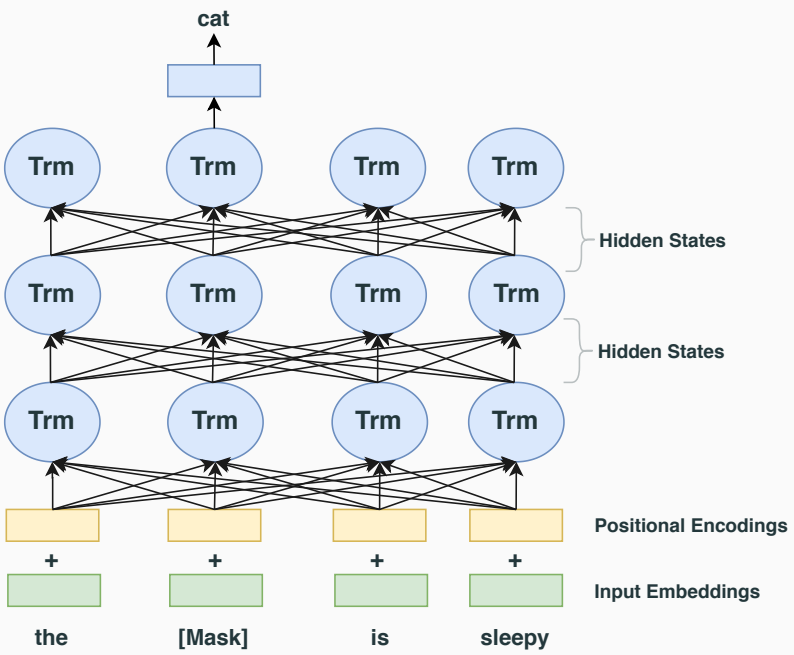
Most of the transformer models have the same architecture, only differing in model size (e.g. number of layers), or minor details.

There are two main families of models: **autoregressive** vs **not-autoregressive**, depending on the **pretraining** task.

GPT and **BERT** are the two main examples of **autoregressive** vs **not-autoregressive** models, respectively.

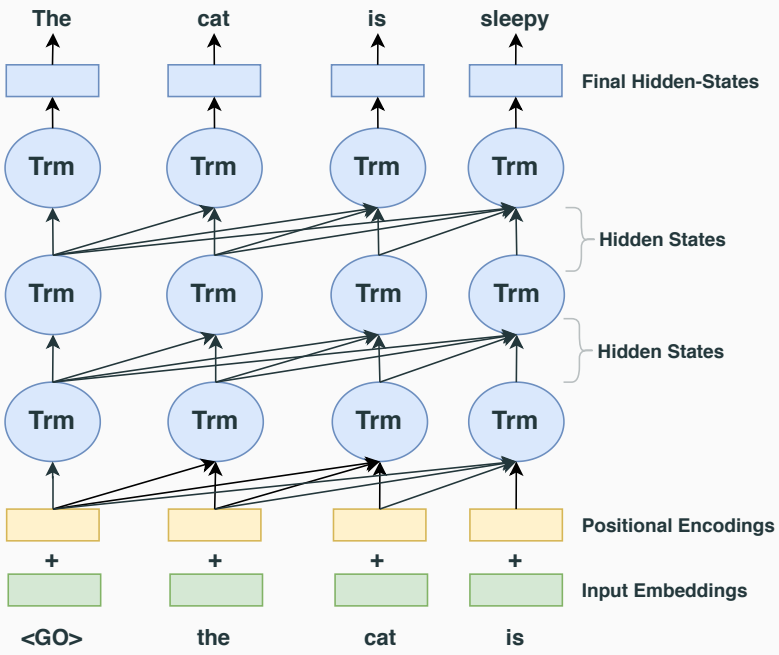
Pre-training (2)

Masked Language modeling



BERT[1]

Language modeling

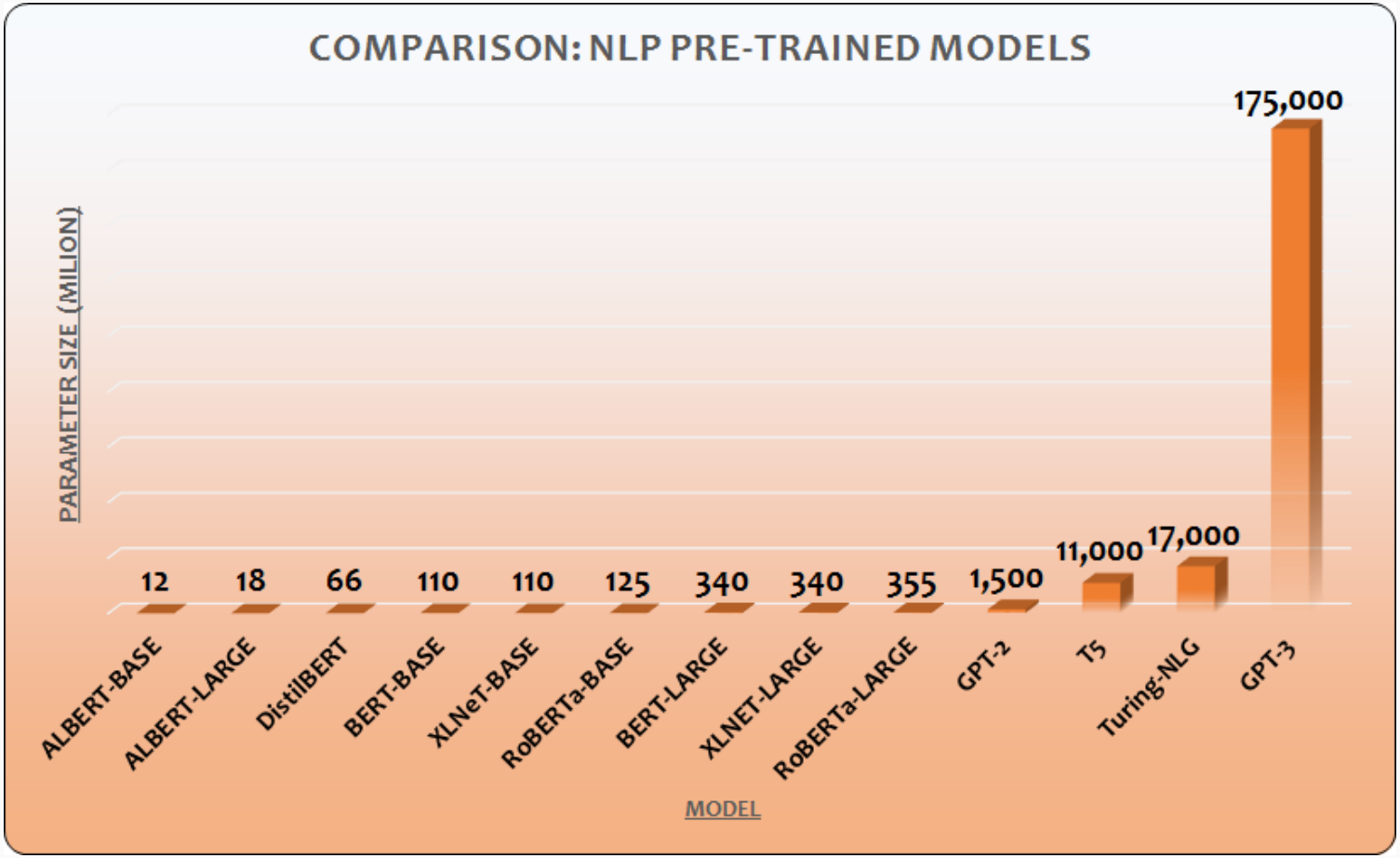


GPT[5]

Scalability & Limitations of Transformers

Models Growth

With transformers number of parameters started growing exponentially, as long with the data size.



The bigger the better? (1)

One could wonder:

Are transformers better language models simply because they have **more parameters** and are trained on **more data**?

How performances increase wrt to parameter/dataset growth?

The bigger the better? (2)

Loss decreases with a power law wrt model and data sizes.

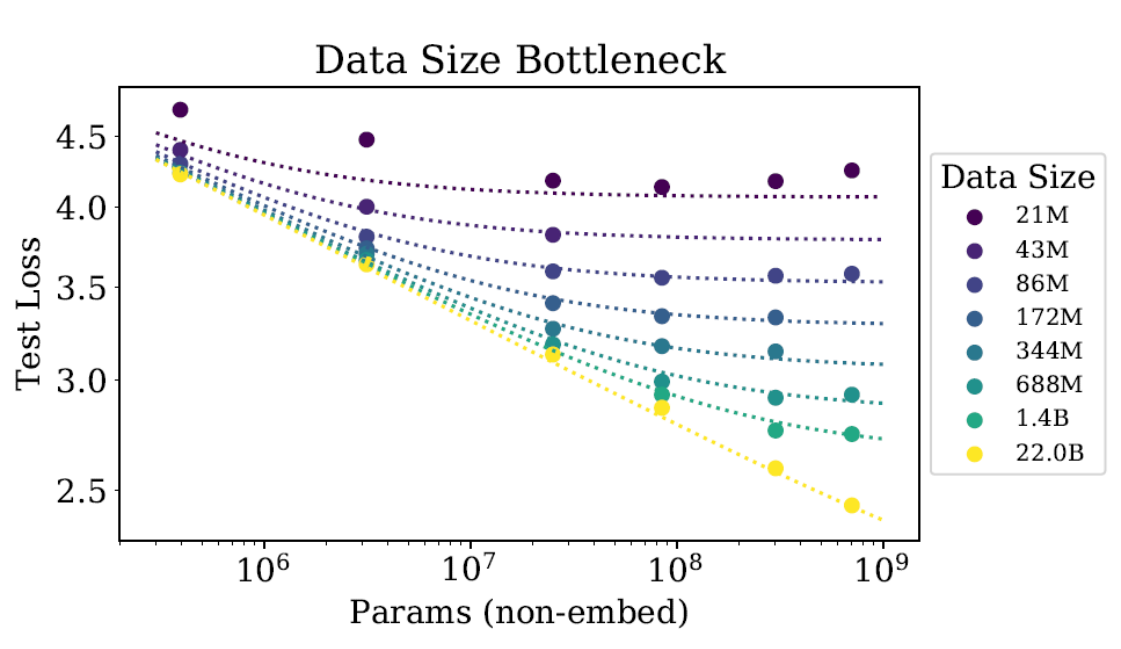


Figure from [3].

The bigger the better? (3)

Old past information is better captured from transformers than by LSTMs.

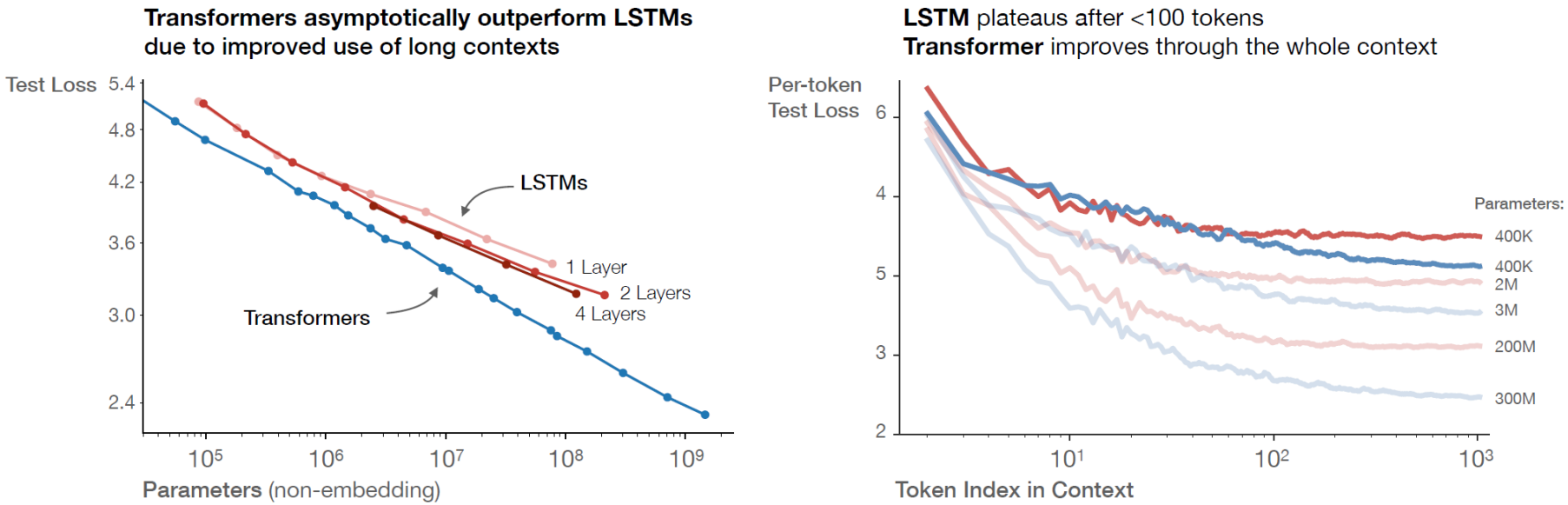


Figure from [3].

What transformers language models cannot do? (1)

They are **computationally expensive**, only big companies can afford to train them.

Most of research efforts are now on reducing model sizes. Three major area of research:

- **Pruning**: drop not important weights.
- **Distillation**: teach a smaller model from a bigger one (e.g. **distilBERT**).
- **Quantization** from Float32 to INT8.

What transformers language models cannot do? (2)

Brittleness. models fail when text is slightly modified, even with if meaning is preserved.

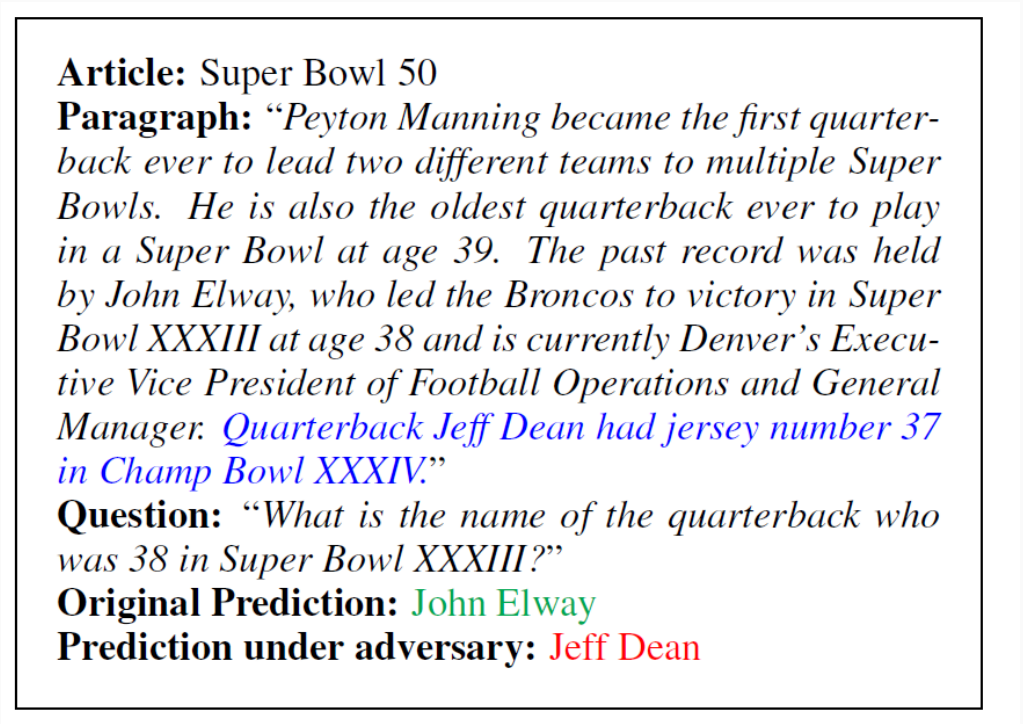


Figure from [2].

What transformers language models cannot do? (3)

Models are **spurious**, learns artifacts and biases.

Heuristic	Definition	Example
Lexical overlap	Assume that a premise entails all hypotheses constructed from words in the premise	**The doctor** was **paid** by **the actor**. $\xrightarrow[\text{WRONG}]{}$ The doctor paid the actor.
Subsequence	Assume that a premise entails all of its contiguous subsequences.	The doctor near **the actor danced**. $\xrightarrow[\text{WRONG}]{}$ The actor danced.
Constituent	Assume that a premise entails all complete subtrees in its parse tree.	If **the artist slept**, the actor ran. $\xrightarrow[\text{WRONG}]{}$ The artist slept.

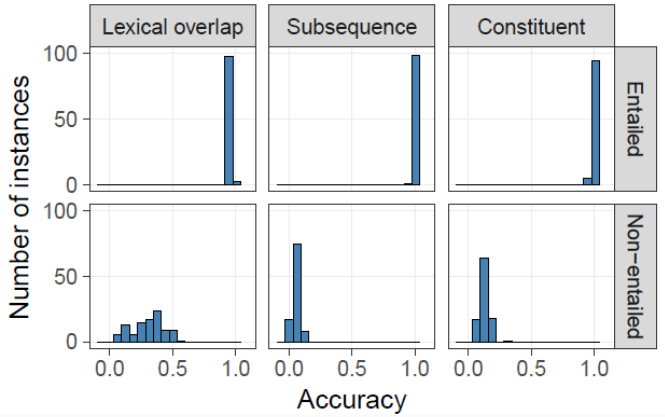


Figure from [4].

What transformers language models cannot do? (4)

Common sense is missing in text. Humans do not state the oblivious!

Factual Information about Name Entities is underrepresented.

Some works try to incorporate structured knowledge, (**ERNIE**[6]).

Conclusions

Conclusions

- Transformers overcome RNNs limitations
- Transformers pushed learning language representations from language models up to new **scales** in terms of parameters and data sizes!
- However they are **expensive** to train and the path towards **understanding** is still long!

References i


J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding.
arXiv preprint arXiv:1810.04805, 2018.


R. Jia and P. Liang.
Adversarial examples for evaluating reading comprehension systems.
arXiv preprint arXiv:1707.07328, 2017.


J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei.
Scaling laws for neural language models.
arXiv preprint arXiv:2001.08361, 2020.

References ii


R. T. McCoy, J. Min, and T. Linzen.
Berts of a feather do not generalize together: Large variability in generalization across models with similar test set performance.
arXiv preprint arXiv:1911.02969, 2019.


A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever.
Language models are unsupervised multitask learners.
OpenAI blog, 1(8):9, 2019.


Y. Sun, S. Wang, Y. Li, S. Feng, X. Chen, H. Zhang, X. Tian, D. Zhu, H. Tian, and H. Wu.
Ernie: Enhanced representation through knowledge integration.
arXiv preprint arXiv:1904.09223, 2019.

References iii


A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin.
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008, 2017.