

Lecture 14: RNNs, Attention, Transformers

CS486/686 Intro to Artificial Intelligence

2026-2-26

Pascal Poupart
David R. Cheriton School of Computer Science

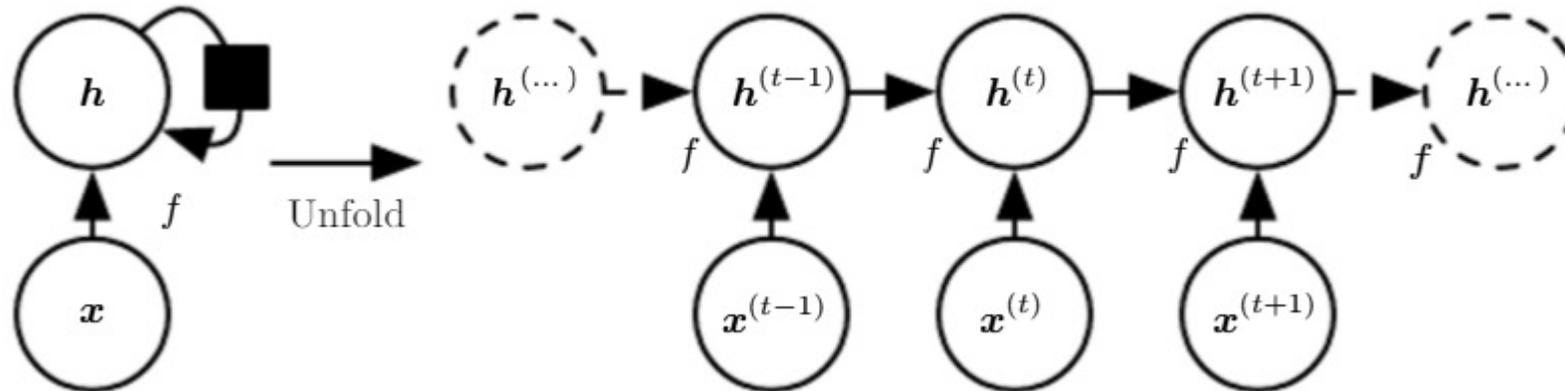


Variable length data

- Traditional feed forward neural networks can only handle fixed length data
- Variable length data (e.g., sequences, time-series, spatial data) leads to a variable # of parameters
- Solutions:
 - Convolutional neural networks
 - **Recurrent neural networks**
 - Graph neural networks (including recursive neural networks)

Recurrent Neural Network (RNN)

- In RNNs, outputs can be fed back to the network as inputs, creating a recurrent structure that can be unrolled to handle varying length data.



Training

- Recurrent neural networks are trained by backpropagation on the unrolled network
 - E.g. backpropagation through time
- Weight sharing:
 - Combine gradients of shared weights into a single gradient
- Challenges:
 - Gradient vanishing (and explosion)
 - Long range memory
 - Prediction drift

RNN for belief monitoring

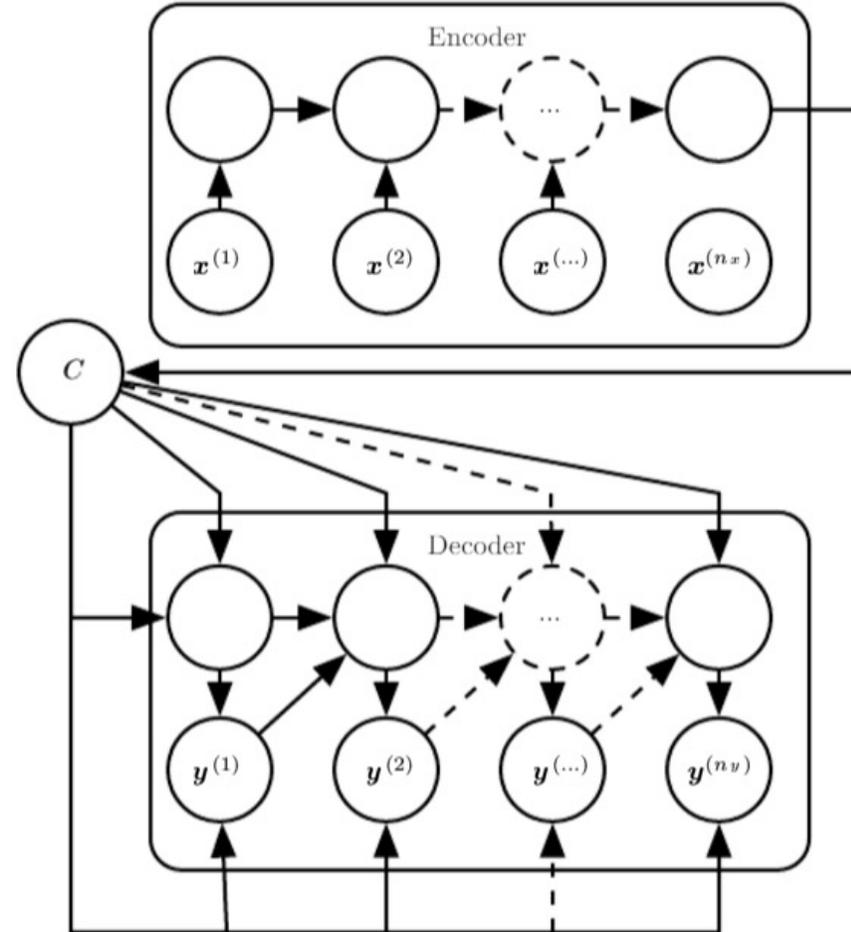
- HMM can be simulated and generalized by a RNN

Bi-Directional RNN

- We can combine past and future evidence in separate chains

Encoder-Decoder Model

- Also known as sequence2sequence
 - $x^{(i)}$: i^{th} input
 - $y^{(i)}$: i^{th} output
 - c : context (embedding)
- Usage:
 - Machine translation
 - Question answering
 - Dialog



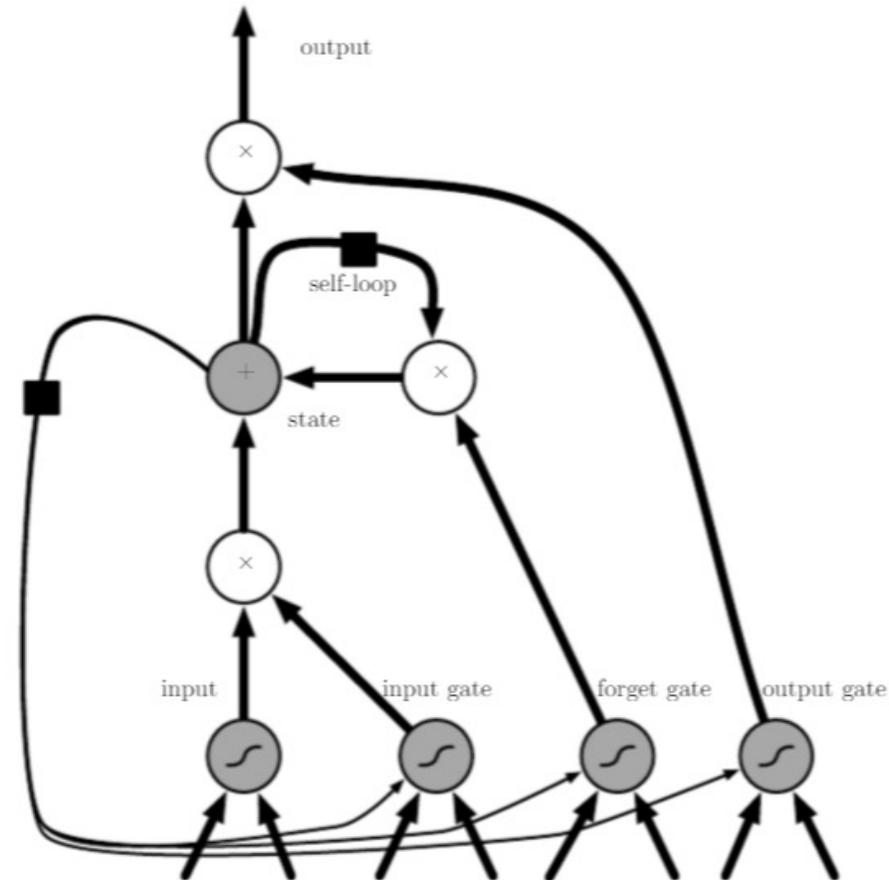
Machine Translation

- Cho, van Merriënboer, Gulcehre, Bahdanau, Bougares, Schwenk, Bengio (2014) Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation

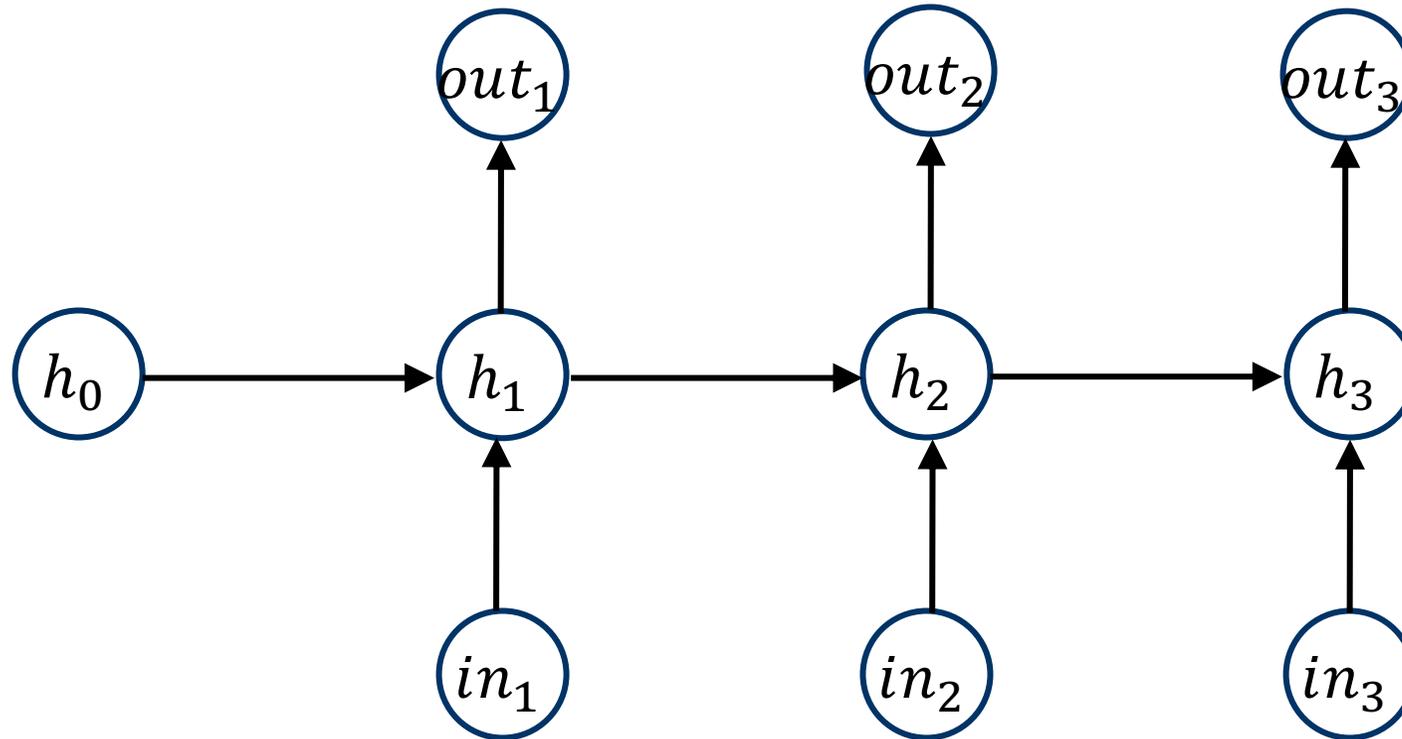
| Source | Translation Model | RNN Encoder-Decoder |
|--------------------------|---|---|
| at the end of the | [a la fin de la] [f la fin des années] [être supprimés à la fin de la] | [à la fin du] [à la fin des] [à la fin de la] |
| for the first time | [r © pour la première fois] [été donnés pour la première fois] [été commémorée pour la première fois] | [pour la première fois] [pour la première fois ,] [pour la première fois que] |
| in the United States and | [? aux ?tats-Unis et] [été ouvertes aux États-Unis et] [été constatées aux États-Unis et] | [aux Etats-Unis et] [des Etats-Unis et] [des États-Unis et] |
| , as well as | [?s , qu'] [?s , ainsi que] [?re aussi bien que] | [, ainsi qu'] [, ainsi que] [, ainsi que les] |
| one of the most | [?t ?l' un des plus] [?l' un des plus] [être retenue comme un de ses plus] | [l' un des] [le] [un des] |

Long Short-Term Memory (LSTM)

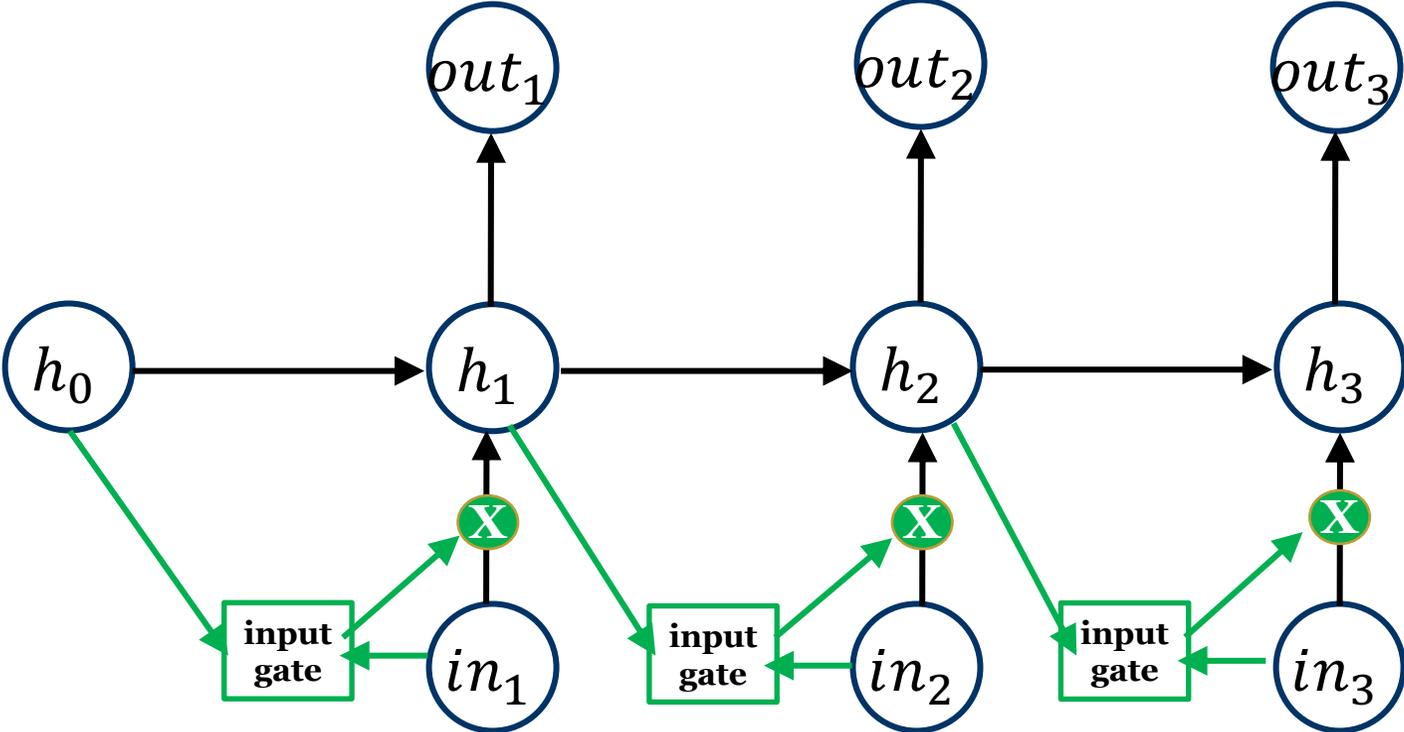
- Special gated structure to control memorization and forgetting in RNNs
- Mitigate gradient vanishing
- Facilitate long term memory



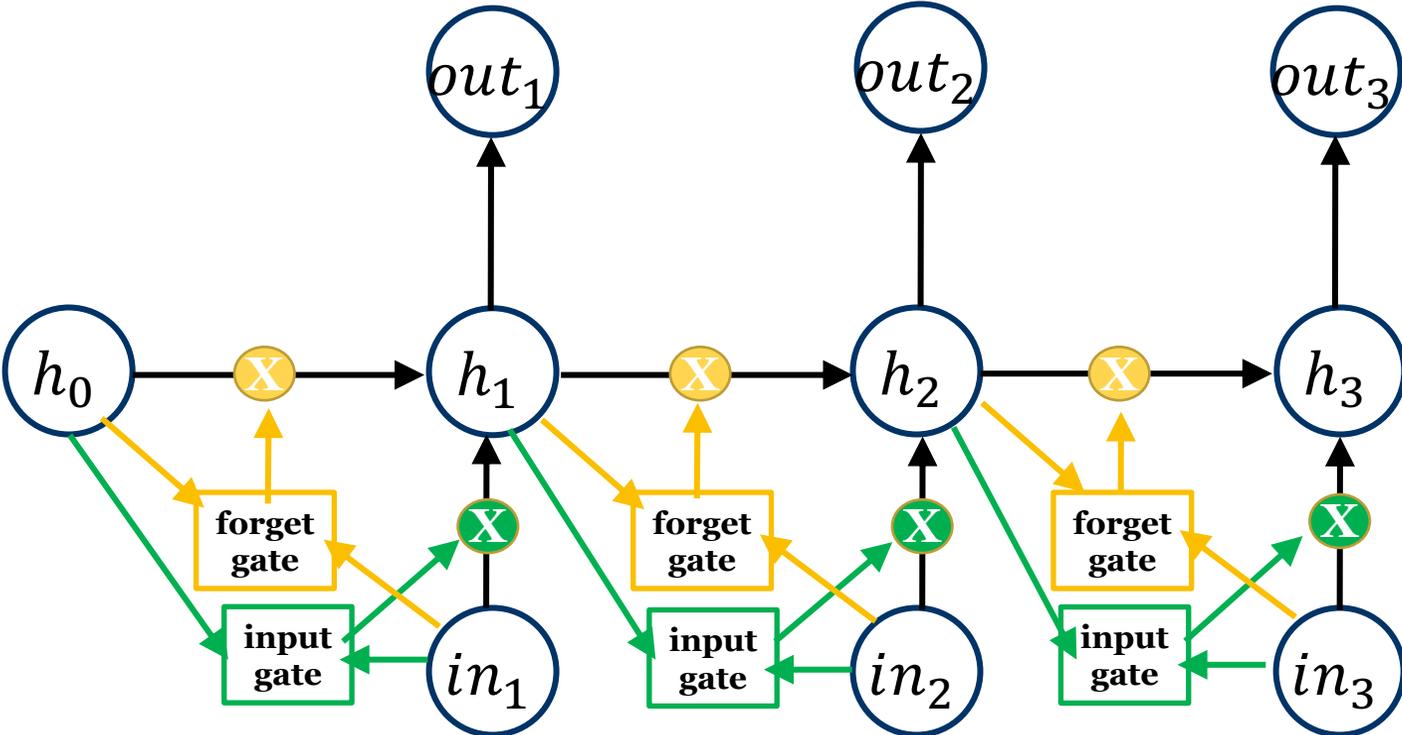
Unrolled Long Short-Term Memory



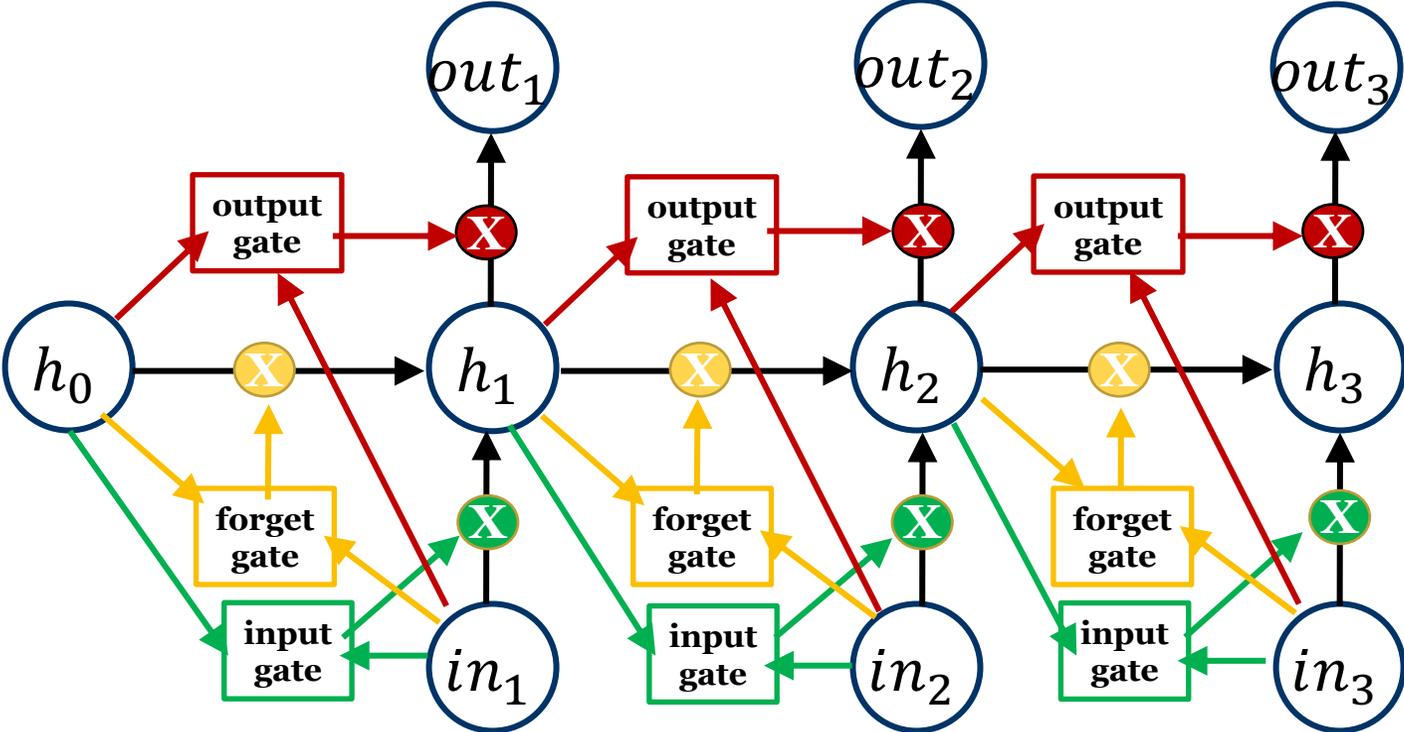
Unrolled Long Short-Term Memory



Unrolled Long Short-Term Memory

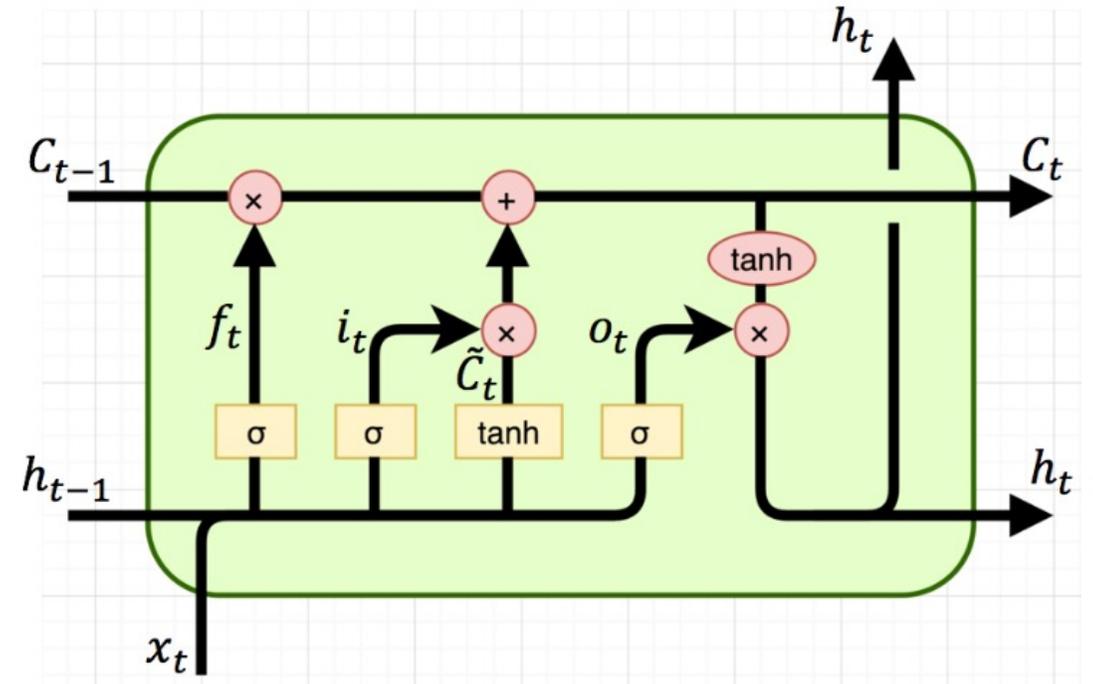


Unrolled Long Short-Term Memory



LSTM cell in practice

- Adjustments:
 - Hidden state h_t called cell state c_t
 - Output y_t called hidden state h_t
- Update equations
 - Input gate: $i_t = \sigma(W^{(ii)}\bar{x}_t + W^{(hi)}h_{t-1})$
 - Forget gate: $f_t = \sigma(W^{(if)}\bar{x}_t + W^{(hf)}h_{t-1})$
 - Output gate: $o_t = \sigma(W^{(io)}\bar{x}_t + W^{(ho)}h_{t-1})$
 - Process input: $\tilde{c}_t = \tanh(W^{(i\tilde{c})}\bar{x}_t + W^{(h\tilde{c})}h_{t-1})$
 - Cell update: $c_t = f_t * c_{t-1} + i_t * \tilde{c}_t$
 - Output: $y_t = h_t = o_t * \tanh(c_t)$



Gated Recurrent Unit (GRU)

- Simplified LSTM

- No cell state
- Two gates (instead of three)
- Fewer weights

- Update equations

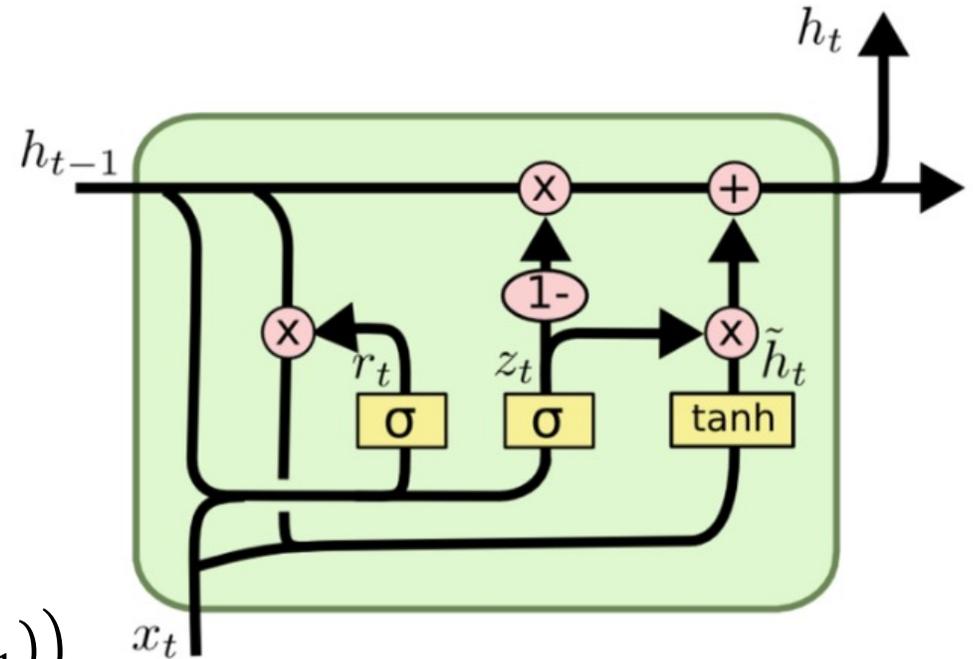
Reset gate: $r_t = \sigma(W^{(ir)}\bar{x}_t + W^{(hr)}h_{t-1})$

Update gate: $z_t = \sigma(W^{(iz)}\bar{x}_t + W^{(hz)}h_{t-1})$

Process input: $\tilde{h}_t = \tanh(W^{(i\tilde{h})}\bar{x}_t + r_t * (W^{(h\tilde{h})}h_{t-1}))$

Hidden state update: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

Output: $y_t = h_t$



Attention

- Mechanism for alignment in machine translation, image captioning, etc.
- Attention in machine translation: align each output word with relevant input words by computing a softmax of the inputs
 - Context vector c_i : weighted sum of input encodings h_j

$$c_i = \sum_j a_{ij} h_j$$

- Where a_{ij} is an alignment weight between input encoding h_j and output encoding s_i

$$a_{ij} = \frac{\exp(\text{alignment}(s_{i-1}, h_j))}{\sum_{j'} \exp(\text{alignment}(s_{i-1}, h_{j'}))} \quad (\text{softmax})$$

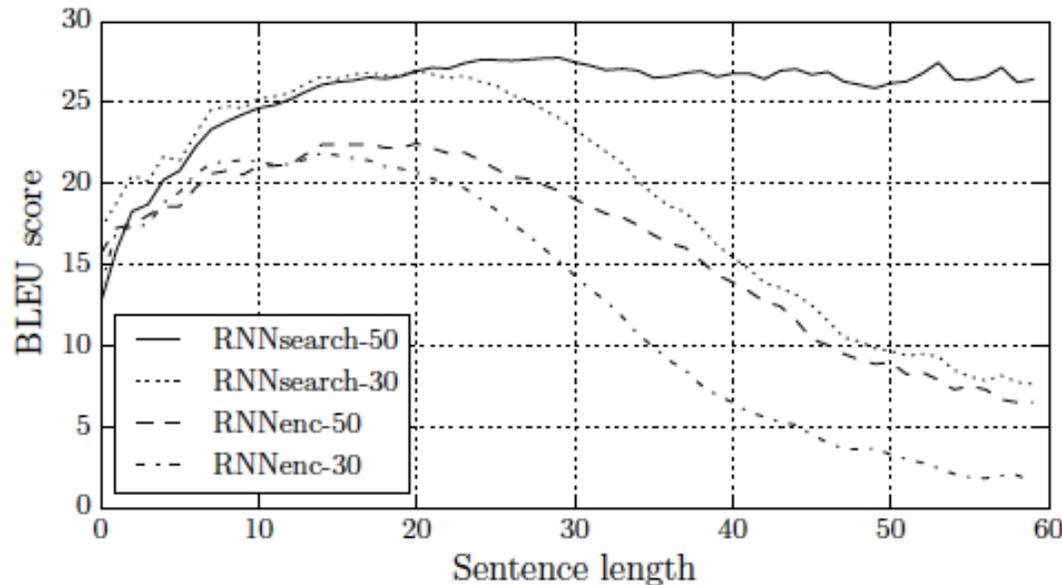
- Alignment example: $\text{alignment}(s_{i-1}, h_j) = s_{i-1}^T h_j$

Attention

- Picture

Machine Translation with Bidirectional RNNs, LSTM units and attention

- Bahdanau, Cho, Bengio (ICLR-2015)

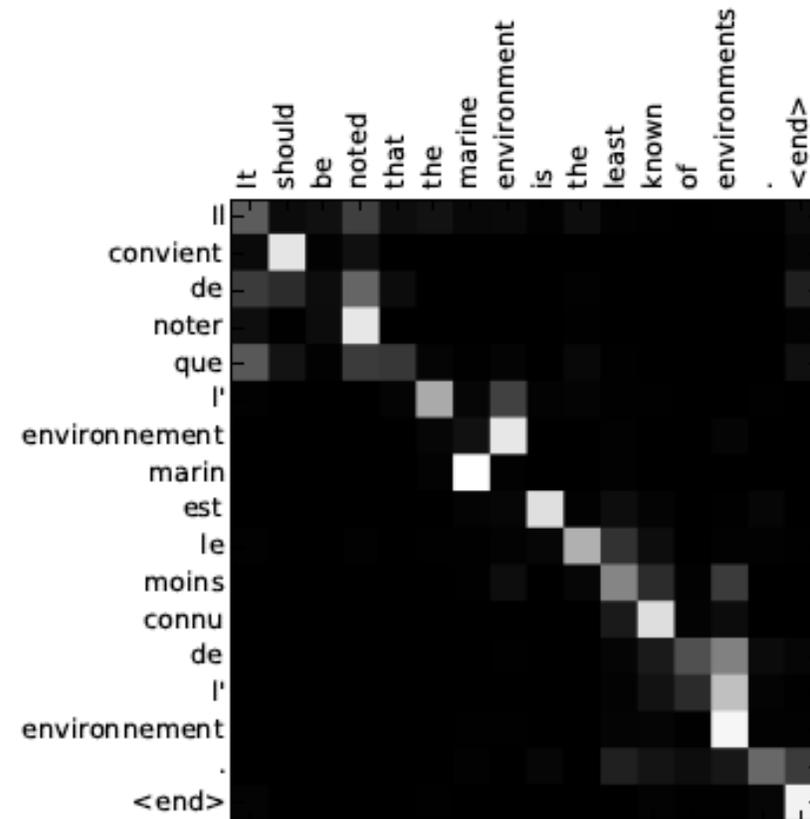
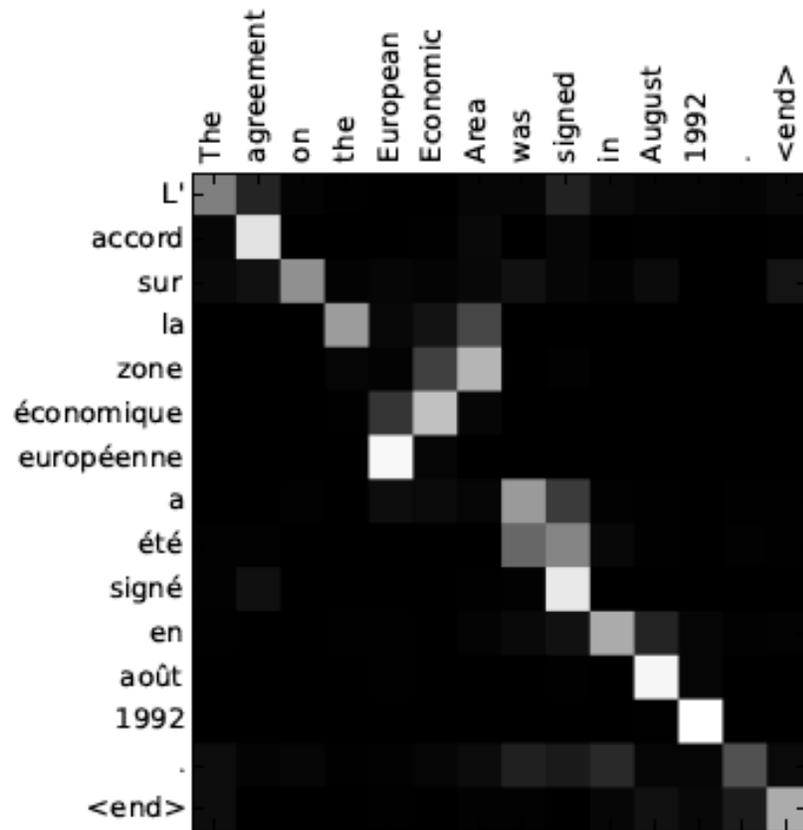


RNNsearch: with attention
RNNenc: no attention

- Bleu: BiLingual Evaluation Understudy
 - Percentage of translated words that appear in ground truth

Alignment example

- Bahdanau, Cho, Bengio (ICLR-2015)



Sequence Modeling

Challenges with RNNs

- Long range dependencies
- Gradient vanishing and explosion
- Large # of training steps
- Recurrence prevents parallel computation

Transformer Networks

- Facilitate long range dependencies
- No gradient vanishing and explosion
- Fewer training steps
- No recurrence that facilitate parallel computation

Attention

- Attention in Computer Vision
 - 2014: Attention used to highlight important parts of an image that contribute to a desired output
- Attention in NLP
 - 2015: Aligned machine translation
 - 2017: Language modeling with **Transformer networks**



Attention Mechanism

- Mimics the retrieval of a **value** v_i for a **query** q based on a **key** k_i in database
- Picture

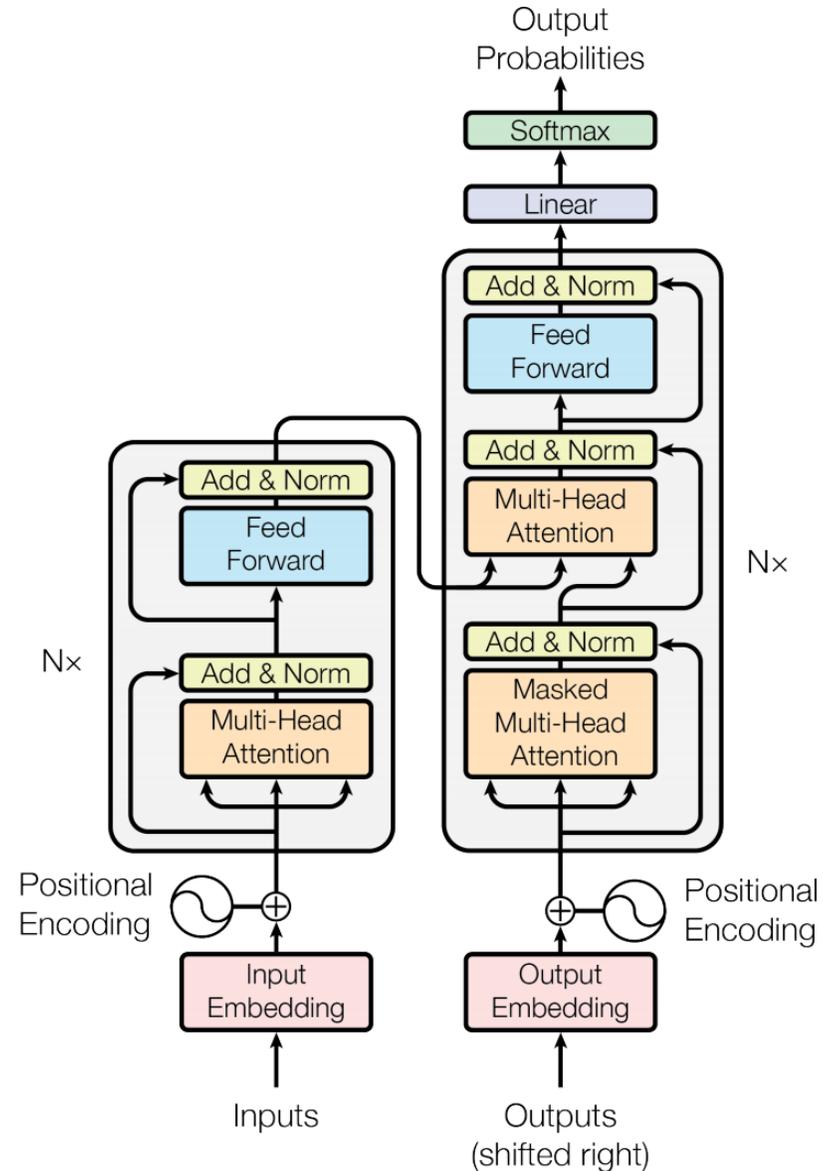
$$attention(q, \mathbf{k}, \mathbf{v}) = \sum_i similarity(q, k_i) \times v_i$$

Attention Mechanism (Neural Architecture)

- Example: machine translation
 - Query: s_{i-1} (hidden vector for $i - 1^{th}$ output word)
 - Key: h_j (hidden vector for j^{th} input word)
 - Value: h_j (hidden vector for j^{th} input word)

Transformer Network

- Vaswani et al., (2017)
Attention is all you need.
- Encoder-decoder based on attention (no recurrence)



Multihead attention

- Multihead attention: compute multiple attentions per query with different weights

$$\text{multihead}(Q, K, V) = W^O \text{concat}(\text{head}_1, \text{head}_2, \dots, \text{head}_n)$$

$$\text{head}_i = \text{attention}(W_i^Q Q, W_i^K K, W_i^V V)$$

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

Masked Multi-head attention

- Masked multi-head attention: multi-head where some values are masked (i.e., probabilities of masked values are nullified to prevent them from being selected).
- When decoding, an output value should only depend on previous outputs (not future outputs). Hence we mask future outputs.

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

$$\textit{maskedAttention}(Q, K, V) = \textit{softmax} \left(\frac{Q^T K + M}{\sqrt{d_k}} \right) V$$

where M is a mask matrix of 0's and $-\infty$'s

Other layers

- Layer normalization:

- Normalize values in each layer to have 0 mean and 1 variance
- For each hidden unit h_i compute $h_i \leftarrow \frac{g}{\sigma}(h_i - \mu)$

where g is a variable, $\mu = \frac{1}{H} \sum_{i=1}^H h_i$ and $\sigma = \sqrt{\frac{1}{H} \sum_{i=1}^H (h_i - \mu)^2}$

- This reduces “covariate shift” (i.e., gradient dependencies between each layer) and therefore fewer training iterations are needed

- Positional embedding (embedding to distinguish each position):

$$PE_{position,2i} = \sin(position/10000^{2i/d})$$

$$PE_{position,2i+1} = \cos(position/10000^{2i/d})$$

Comparison

- Attention reduces sequential operations and maximum path length, which facilitates long range dependencies

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) | $O(r \cdot n \cdot d)$ | $O(1)$ | $O(n/r)$ |

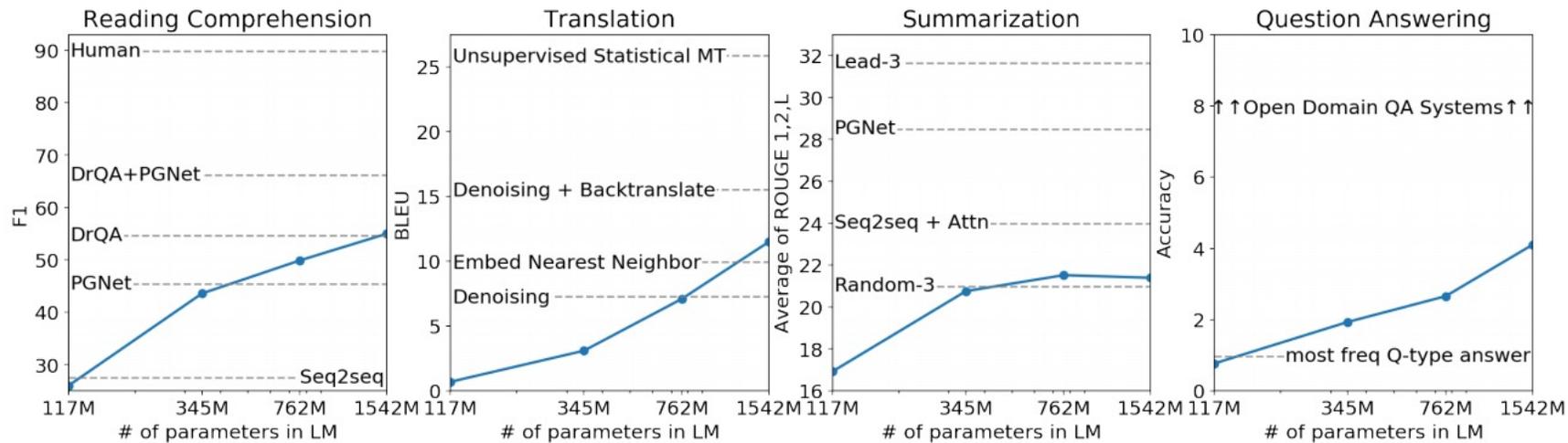
Results

Table 2: 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 | $2.3 \cdot 10^{19}$ | |

GPT and GPT-2

- Radford et al., (2018) Language models are unsupervised multitask learners
 - Decoder transformer that predicts next word based on previous words by computing $P(x_t|x_{1..t-1})$
 - SOTA in “zero-shot” setting for 7/8 language tasks (where zero-shot means no task training, only unsupervised language modeling)



BERT (Bidirectional Encoder Representations from Transformers)

- Devlin et al., (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
 - Decoder transformer that predicts a missing word based on surrounding words by computing $P(x_t|x_{1..t-1,t+1..T})$
 - Mask missing word with masked multi-head attention
 - Improved state of the art on 11 tasks

| 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 |
| BERT _{BASE} | 84.6/83.4 | 71.2 | 90.5 | 93.5 | 52.1 | 85.8 | 88.9 | 66.4 | 79.6 |
| BERT _{LARGE} | 86.7/85.9 | 72.1 | 92.7 | 94.9 | 60.5 | 86.5 | 89.3 | 70.1 | 82.1 |

Limitation

- Transformers **scale quadratically** with sequence length
 - In practice, sequence length often limited to a few thousand tokens
- How can we process long sequences?
 - Hierarchy of transformers (e.g., words → sentences → documents → corpus)
 - Approximate transformers (e.g., longformer, reformer, performer, etc.)
 - Finite memory models (e.g., infini-attention)
 - State space models (e.g., S4, Mamba, Jamba)