Deep Learning for Vision & Language

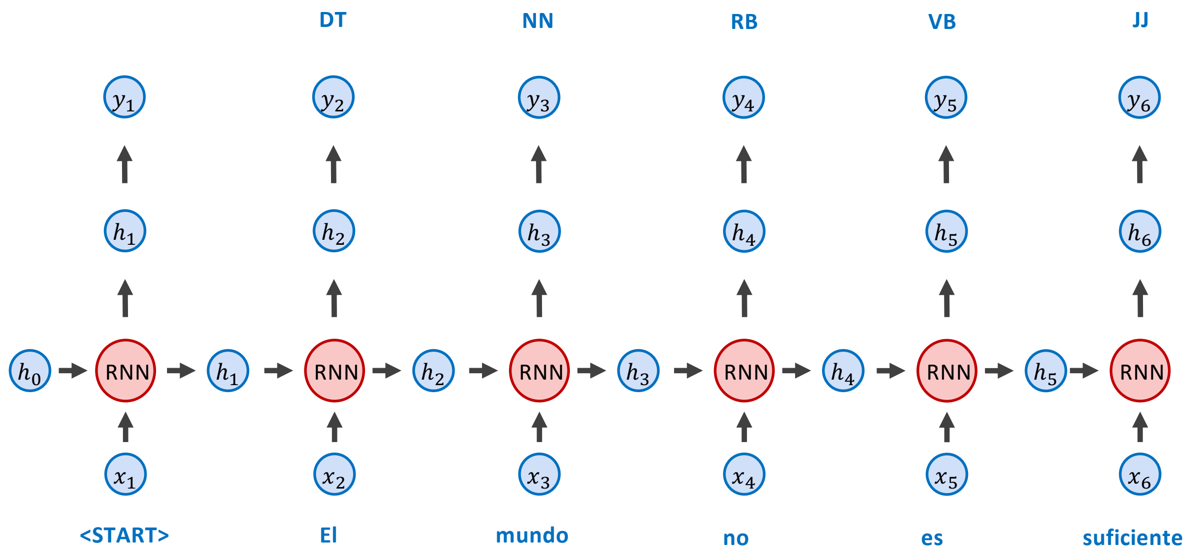
Transformers I: Introduction

RICE UNIVERSITY

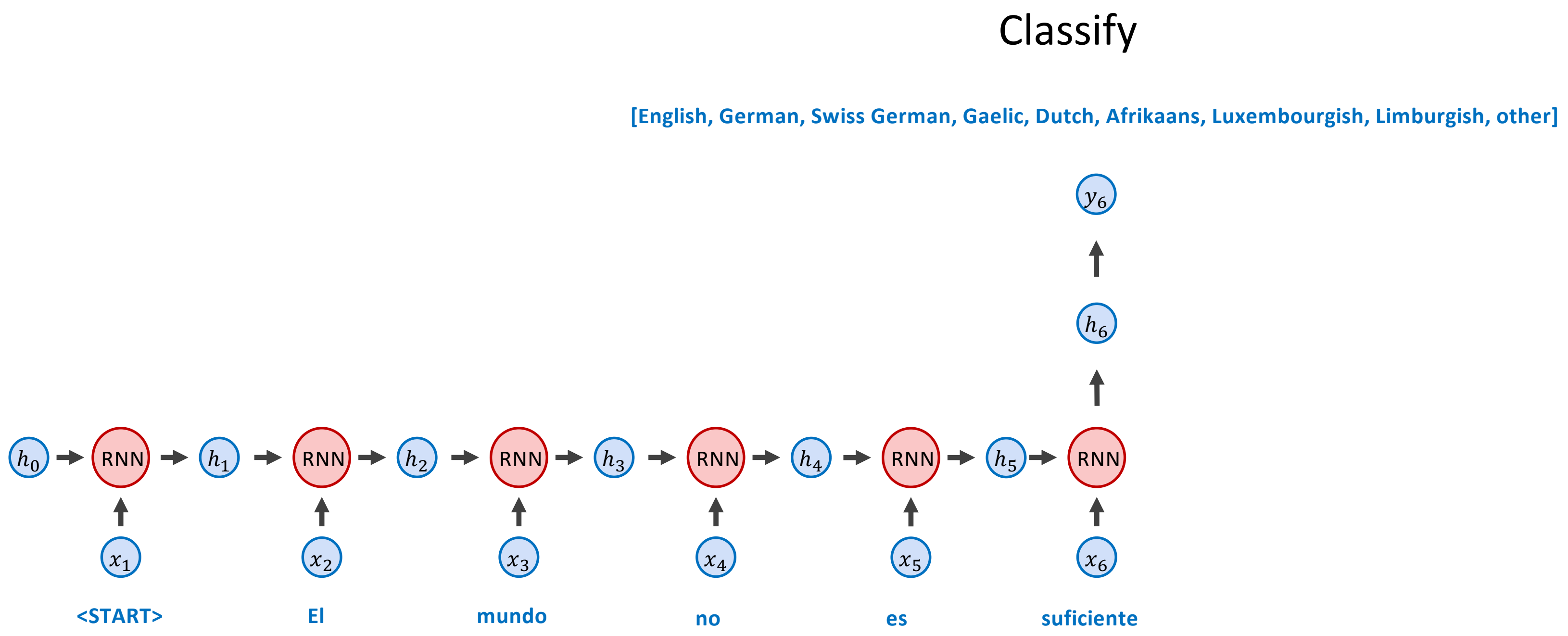
Today

- Sequence-to-sequence (RNNs) for Machine Translation
- Learning to Align and Translate with Soft Attention
- Image Captioning (CNNs + RNNs): Show and Tell
- Image Captioning (CNNs + RNNs + Attention): Show Attend and Tell
- Attention is All you Need!
- Encoder Transformers: BERT
- Decoder Transformers: GPT-2 – maybe next class

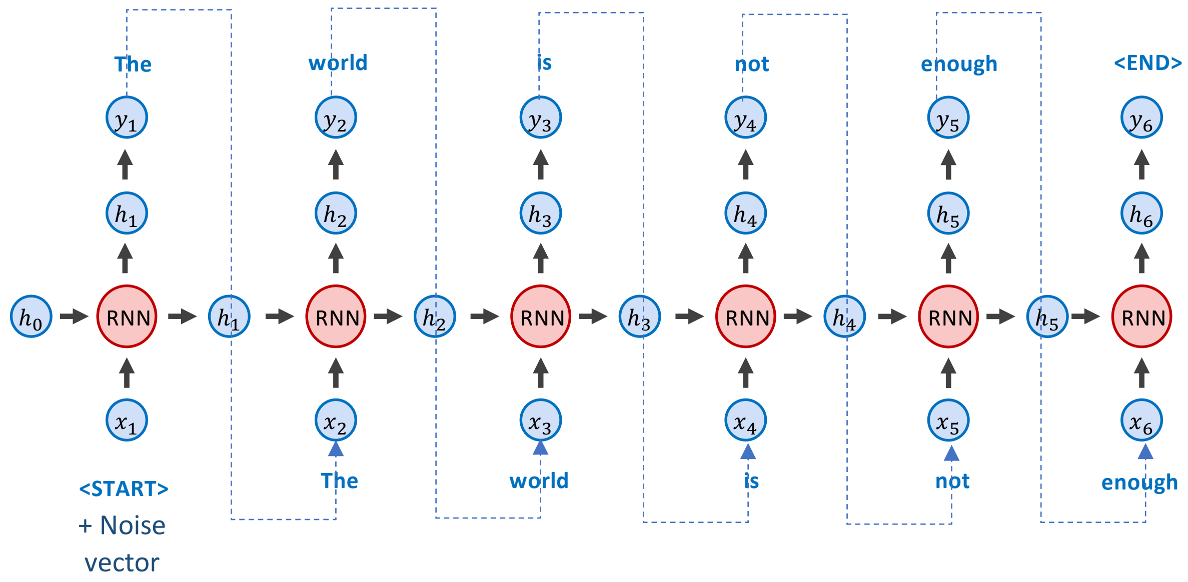
RNNs – One-to-one sequence prediction



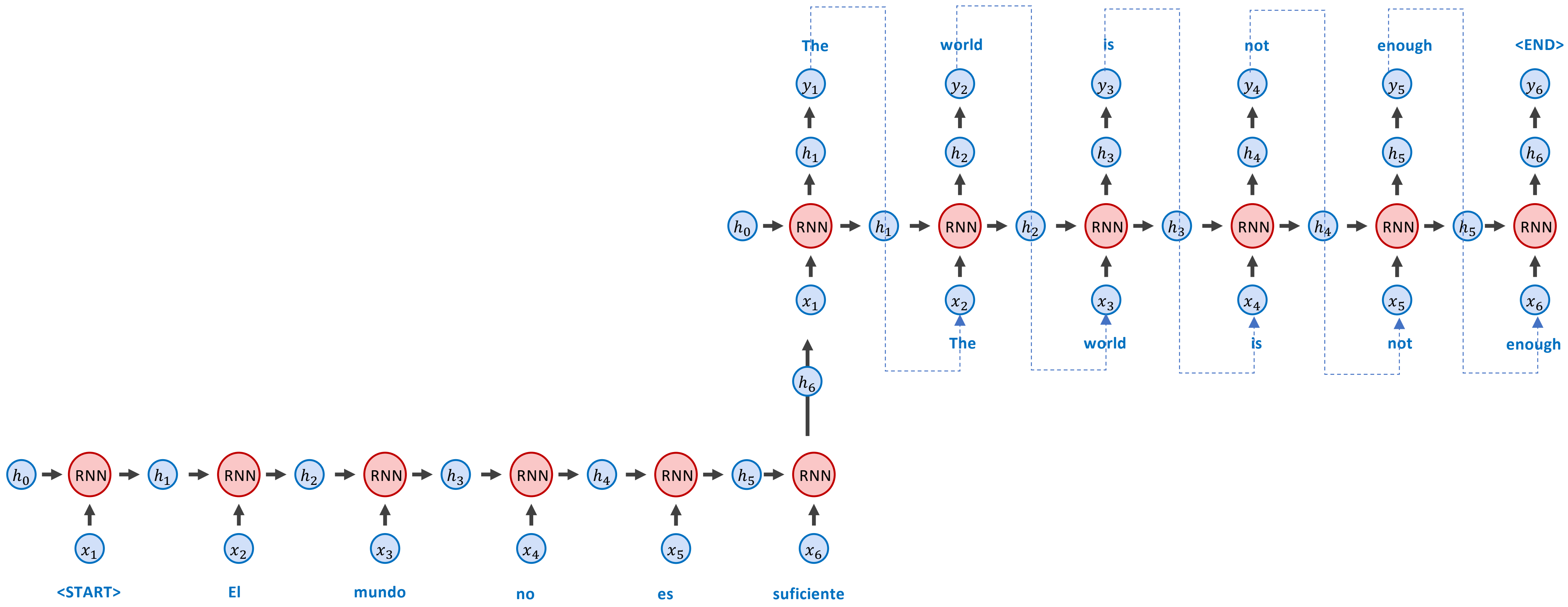
RNNs – Sequence to score prediction



RNNs for Text Generation (Auto-regressive)

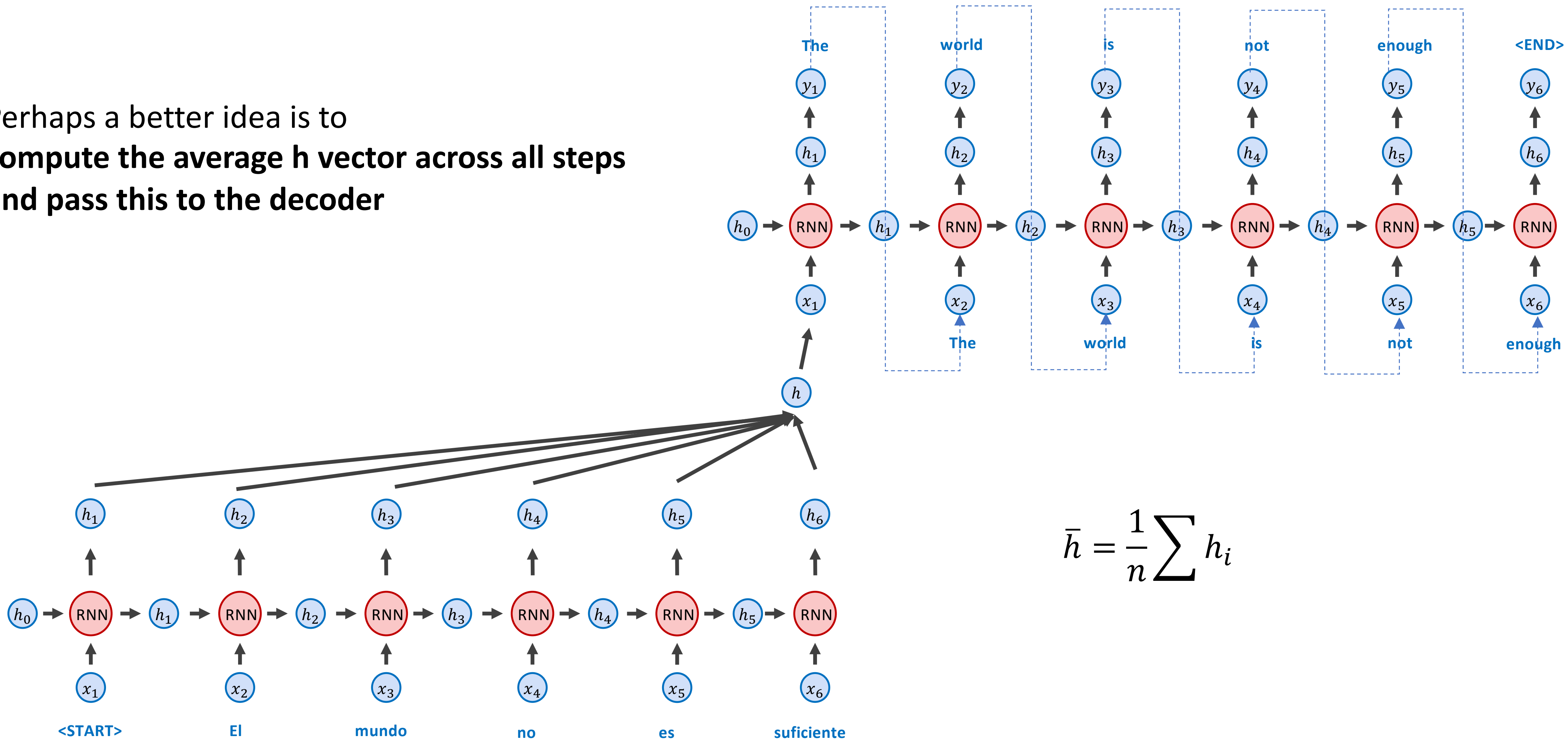


RNNs for Machine Translation Seq-to-Seq



RNNs for Machine Translation Seq-to-Seq

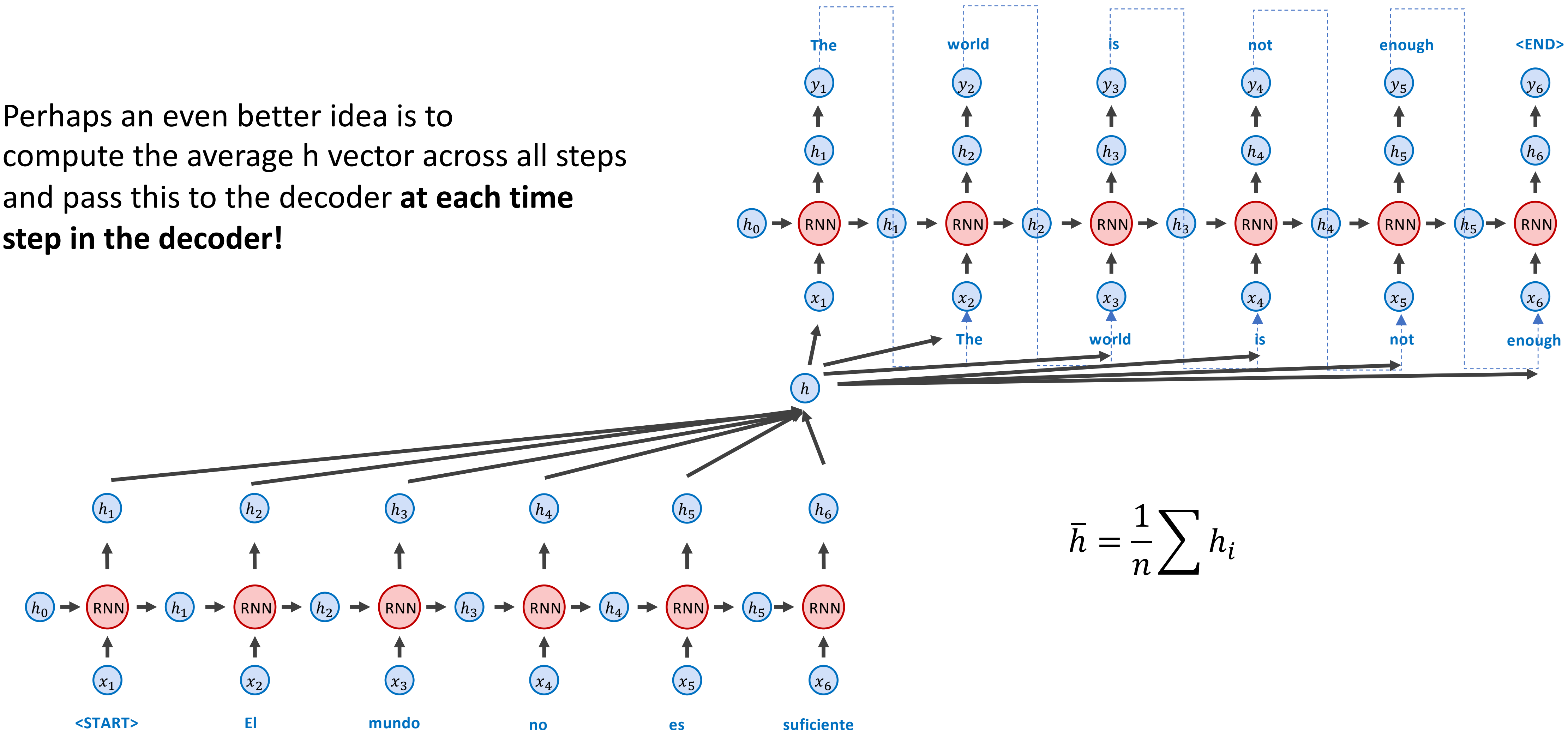
Perhaps a better idea is to
compute the average h vector across all steps and pass this to the decoder



$$\bar{h} = \frac{1}{n}\sum h_i$$

RNNs for Machine Translation Seq-to-Seq

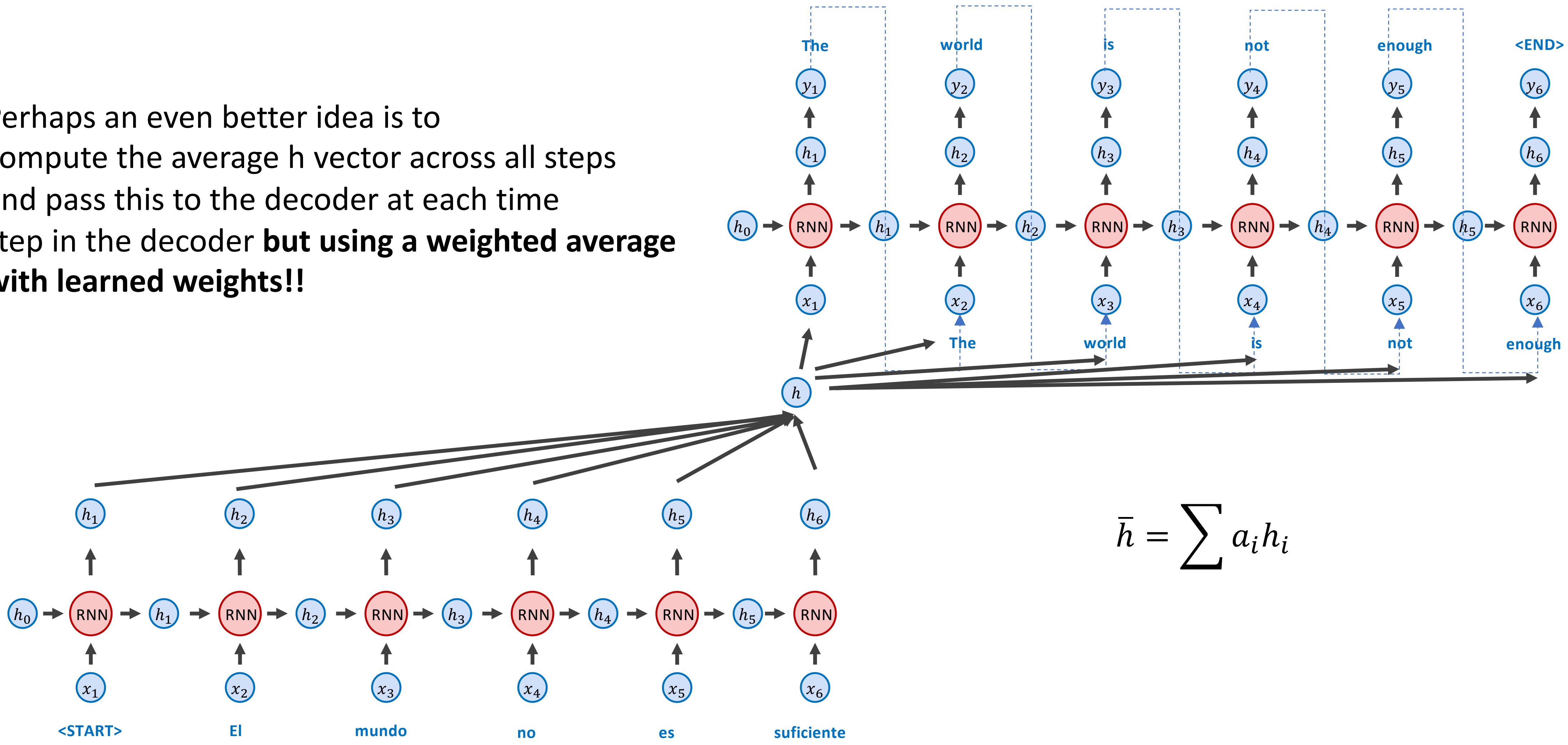
Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder **at each time step in the decoder!**



$$\bar{h} = \frac{1}{n}\sum h_i$$

RNNs for Machine Translation Seq-to-Seq

Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder at each time step in the decoder **but using a weighted average with learned weights!!**

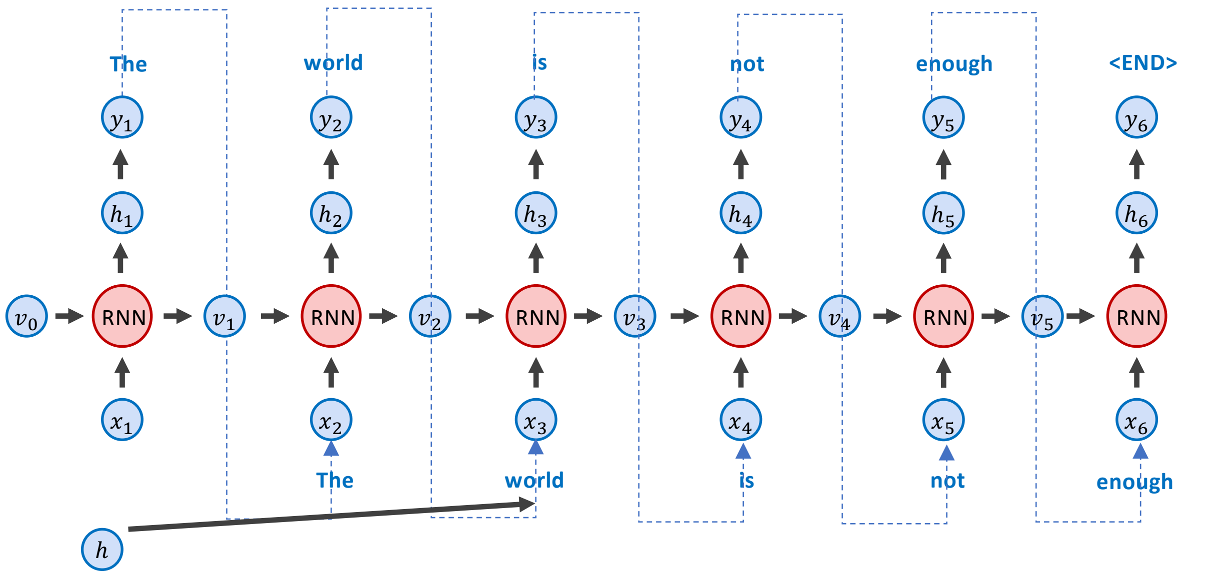


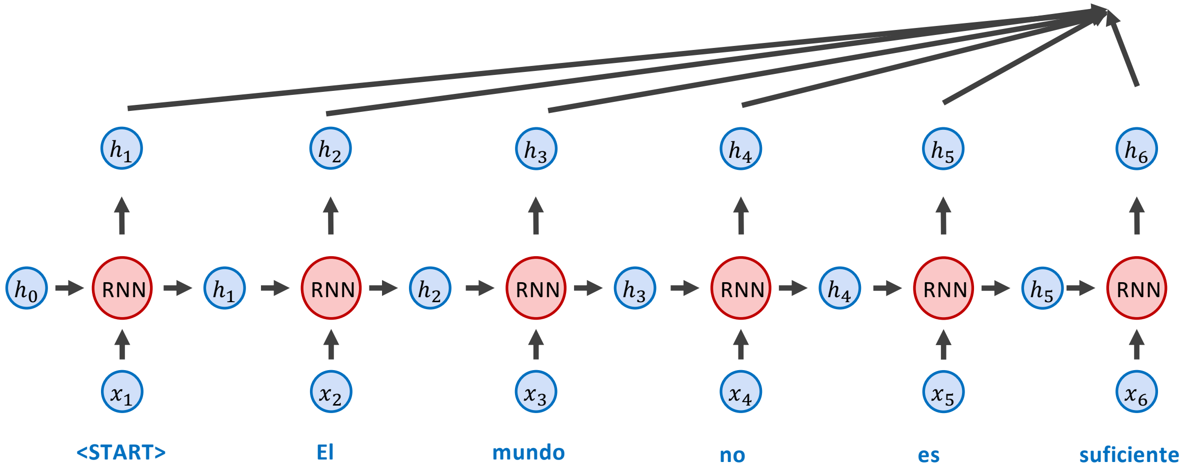
$$\bar{h} = \sum a_i h_i$$

RNNs for Machine Translation Seq-to-Seq

Only showing the third time step encoder-decoder connection

Perhaps an even better idea is to compute the average h vector across all steps and pass this to the decoder at each time step in the decoder but using a weighted average with learned weights, **and the weights are specific for each time step!!!**





$$\bar{h_j} = \sum a_{j,i} h_i$$

such that:

$$a_{j,i} = \frac{\exp(h_j v_{j-1})}{\sum \exp(h_i v_{i-1})}$$

Neural Machine Translation by Jointly Learning to Align and Translate

Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

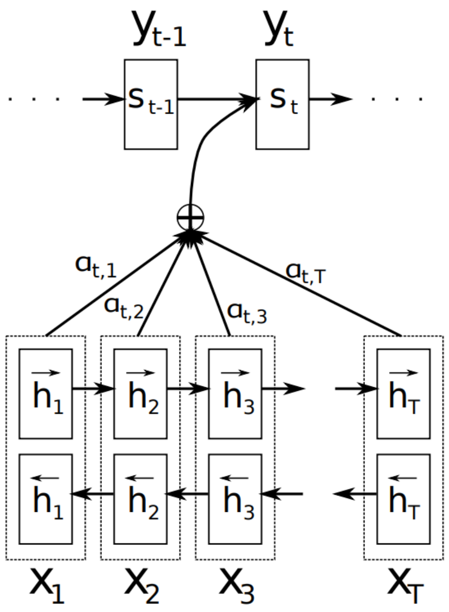


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

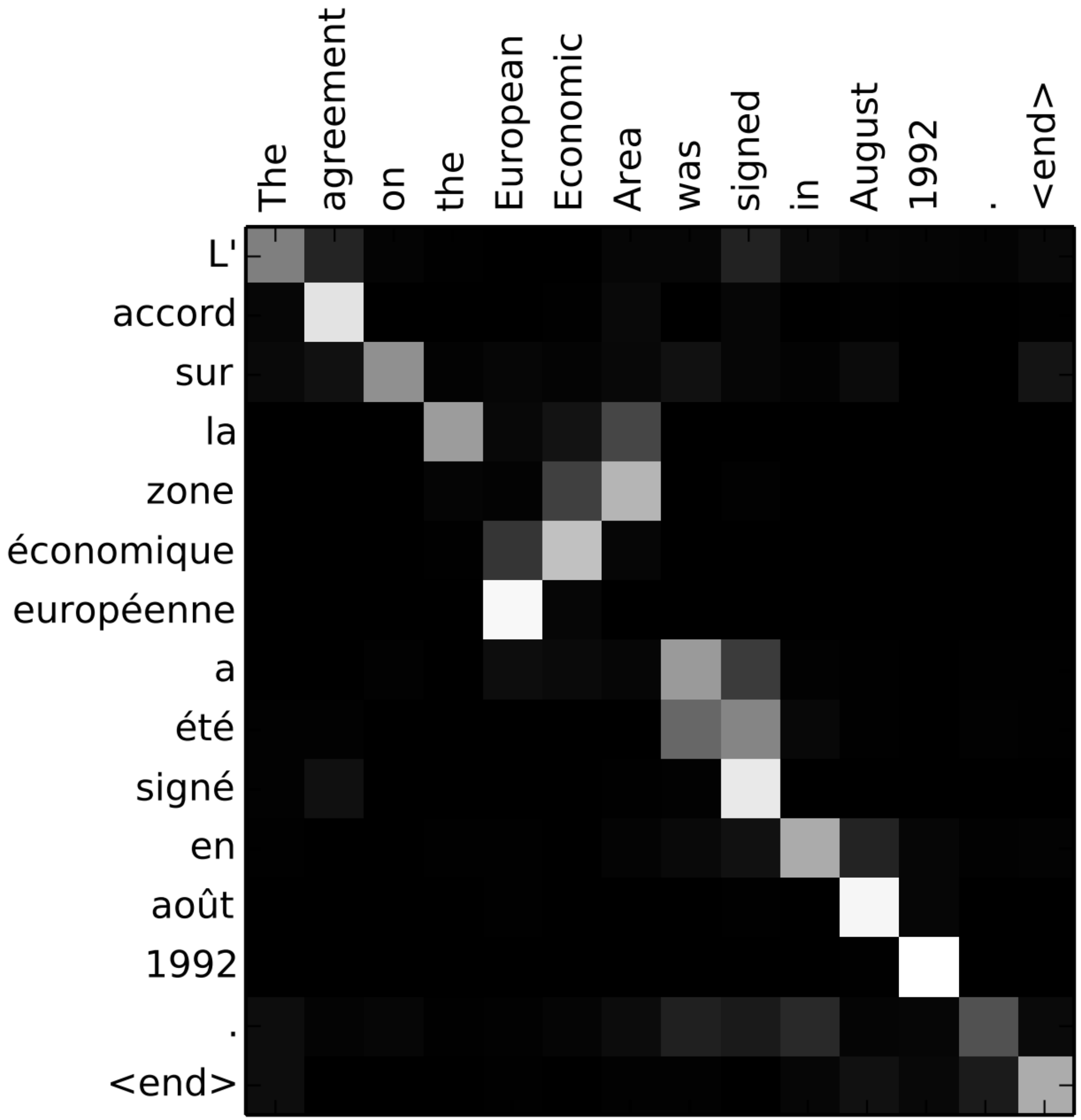
$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

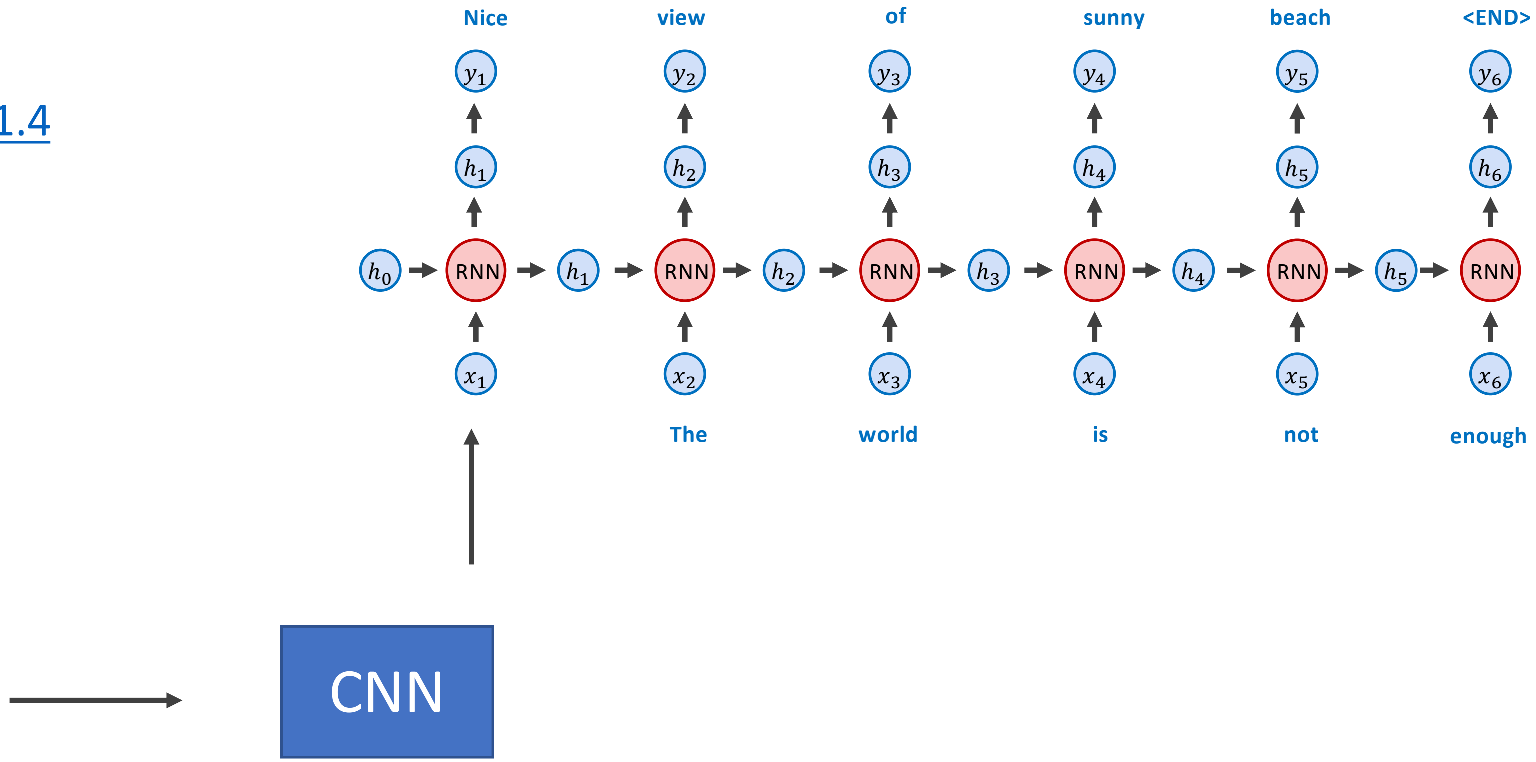
Let's take a look at one of the first papers introducing this idea.

Let’s look at the Attention weights



CNNs + RNNs for Image Captioning

Vinyals et al. Show and Tell: A Neural Image Caption Generator https://arxiv.org/abs/1411.4555



References (a lot of them)

- Vinyals et al. Show and Tell: A Neural Image Caption Generator https://arxiv.org/abs/1411.4555

- Mao et al. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). https://arxiv.org/abs/1412.6632

- Karpathy and Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. https://arxiv.org/abs/1412.2306

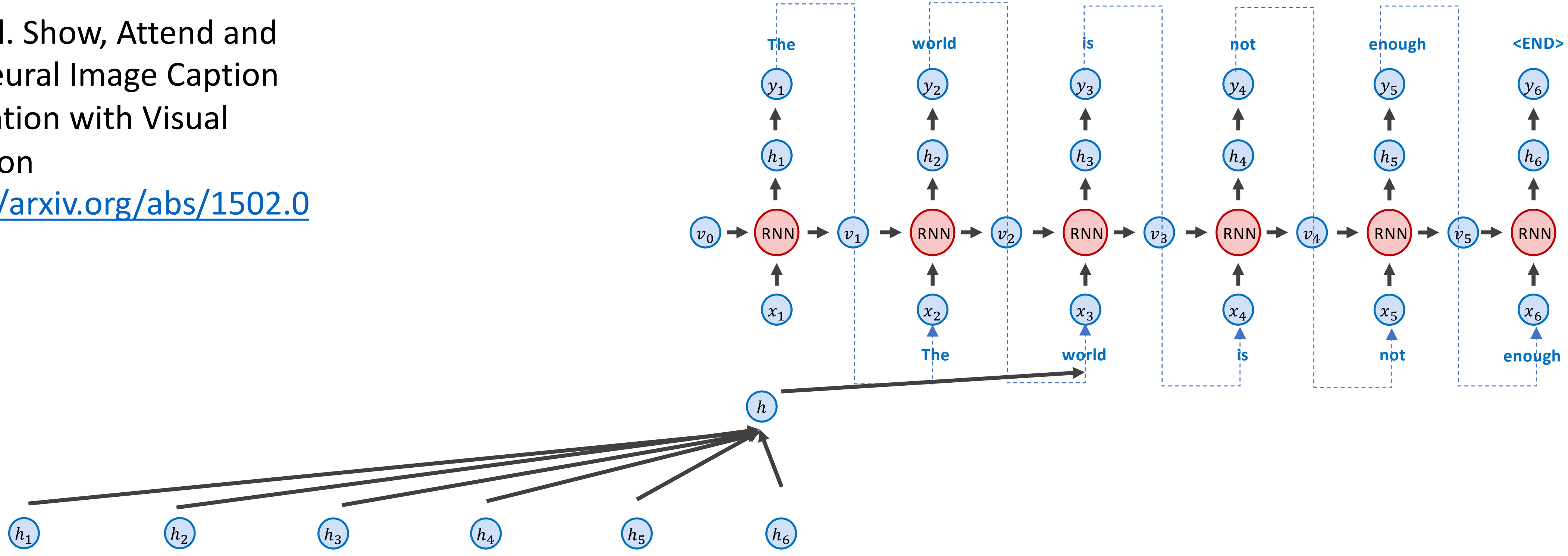
- Fang et al. From Captions to Visual Concepts and Back. https://arxiv.org/abs/1411.4952

- Yin and Ordonez. OBJ2TEXT: Generating Visually Descriptive Language from Object Layouts. https://arxiv.org/abs/1707.07102 (not exactly targeting image captioning specifically but locally grown paper so let me self-promote)

CNNs + RNNs for Image Captioning w/ Attention

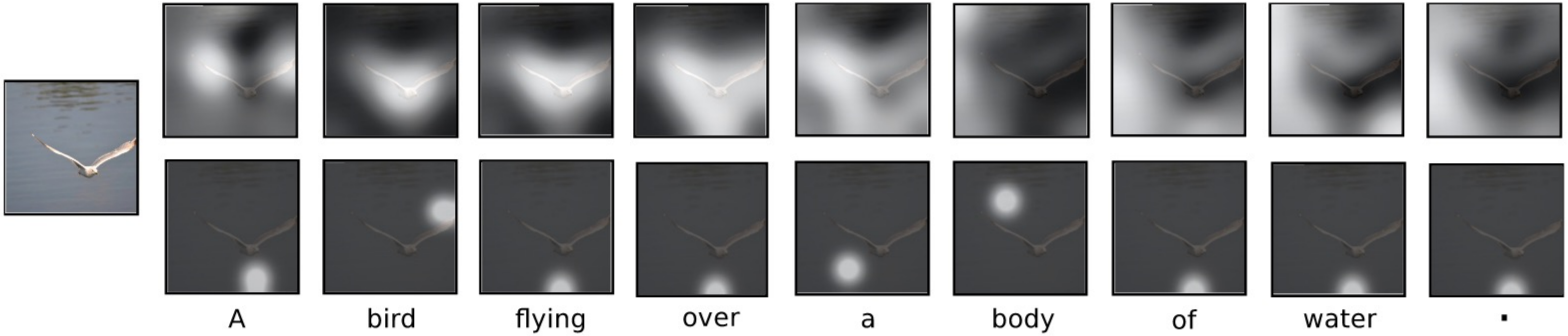
Xu et al. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention
https://arxiv.org/abs/1502.03044

Only showing the third time step encoder-decoder connection



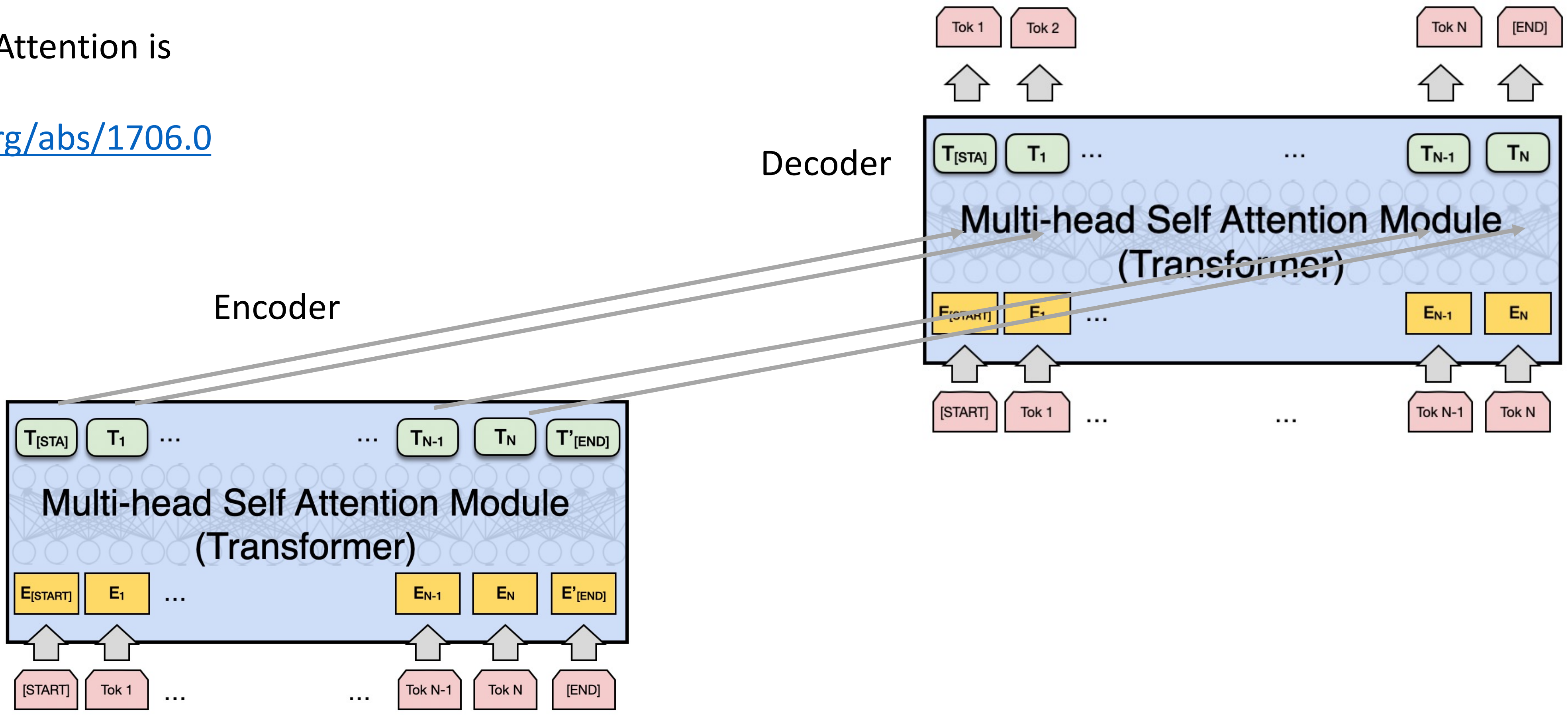
convert to 49 vectors of size 512 and those become h_i

output tensor of size CxHxW e.g. 512x7x7


A
bird
flying
over
a
body
of
water
.

Attention is All you Need (no RNNs)

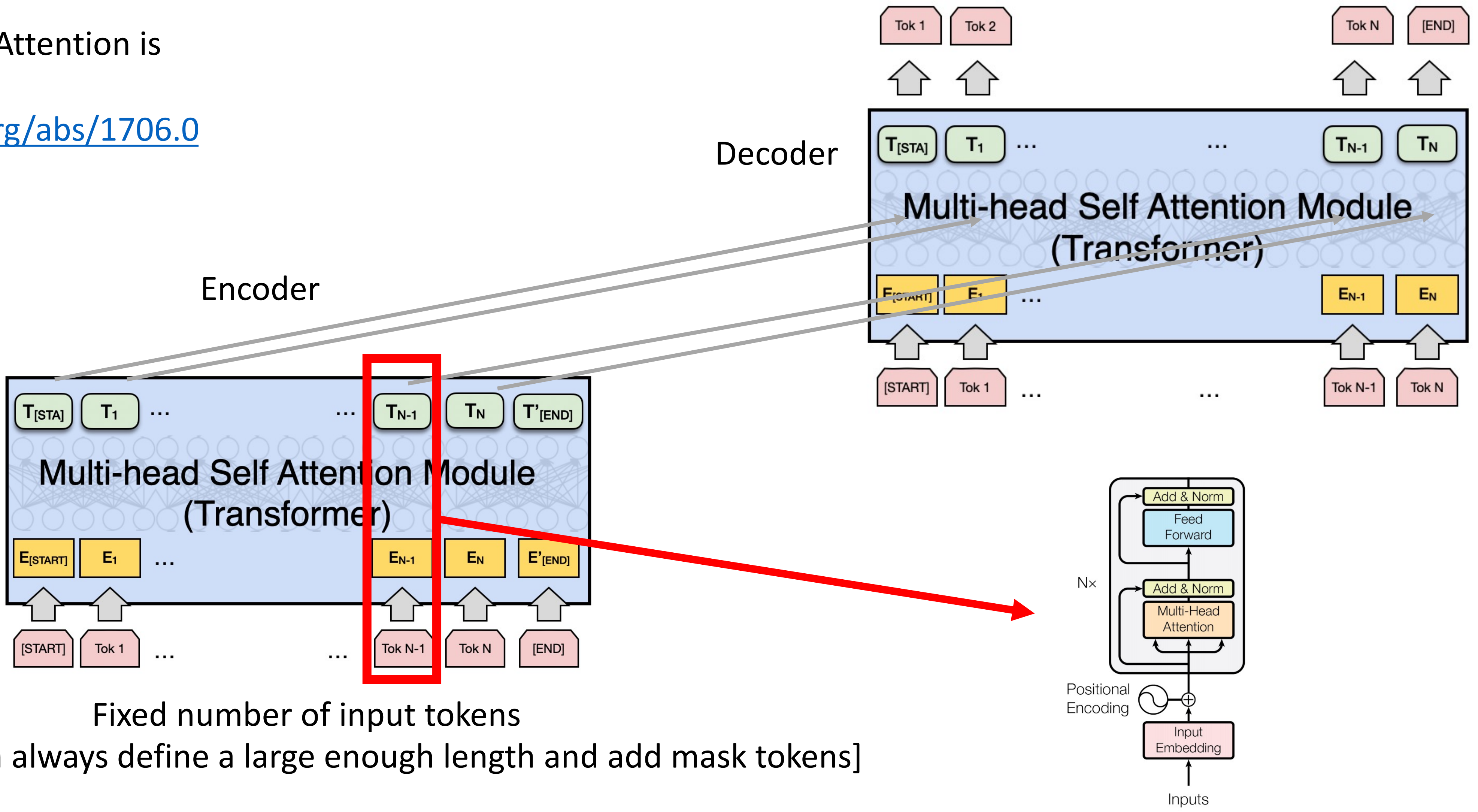
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762



Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]

Attention is All you Need (no RNNs)

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

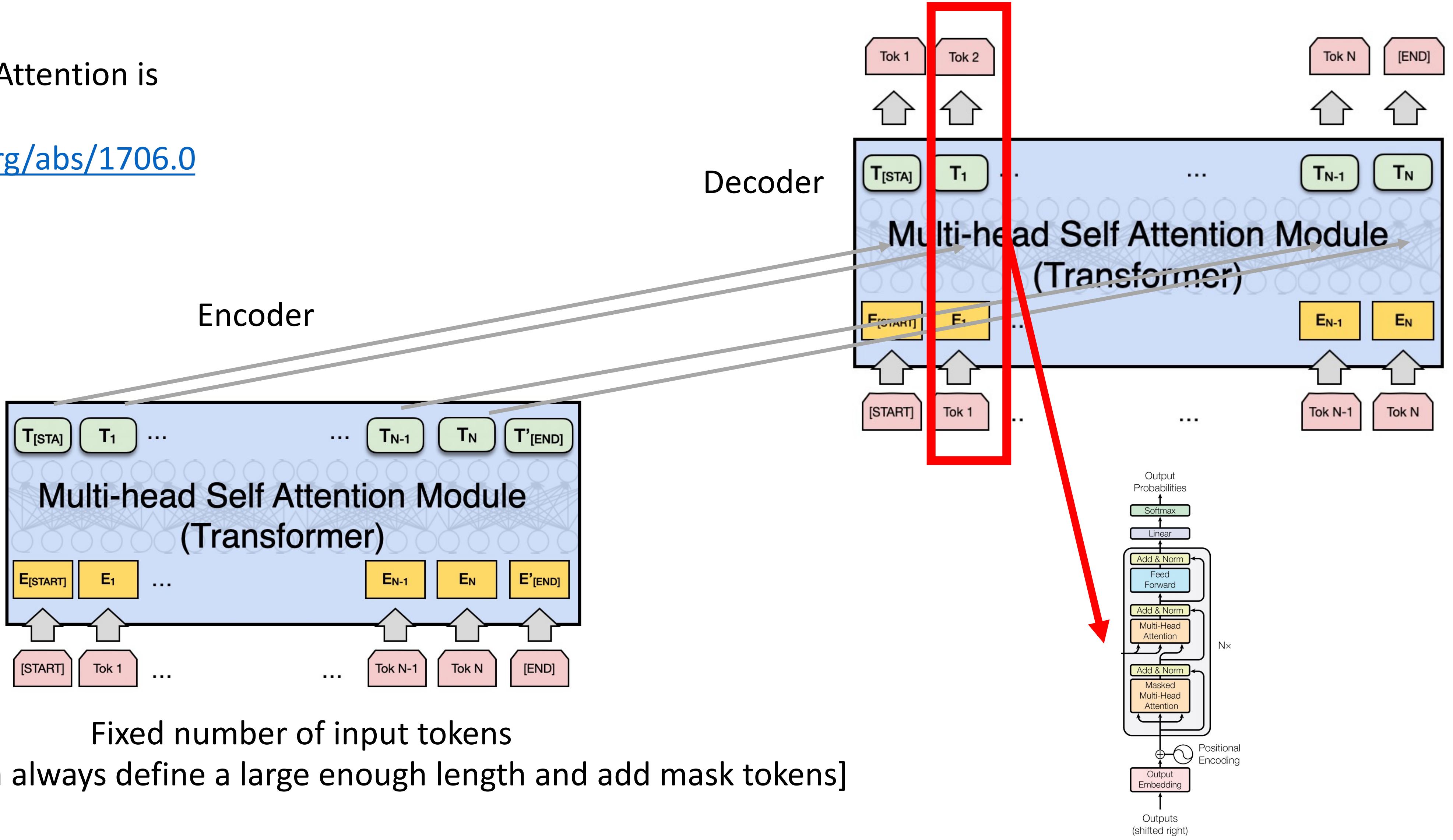


Fixed number of input tokens

[but hey! we can always define a large enough length and add mask tokens]

Attention is All you Need (no RNNs)

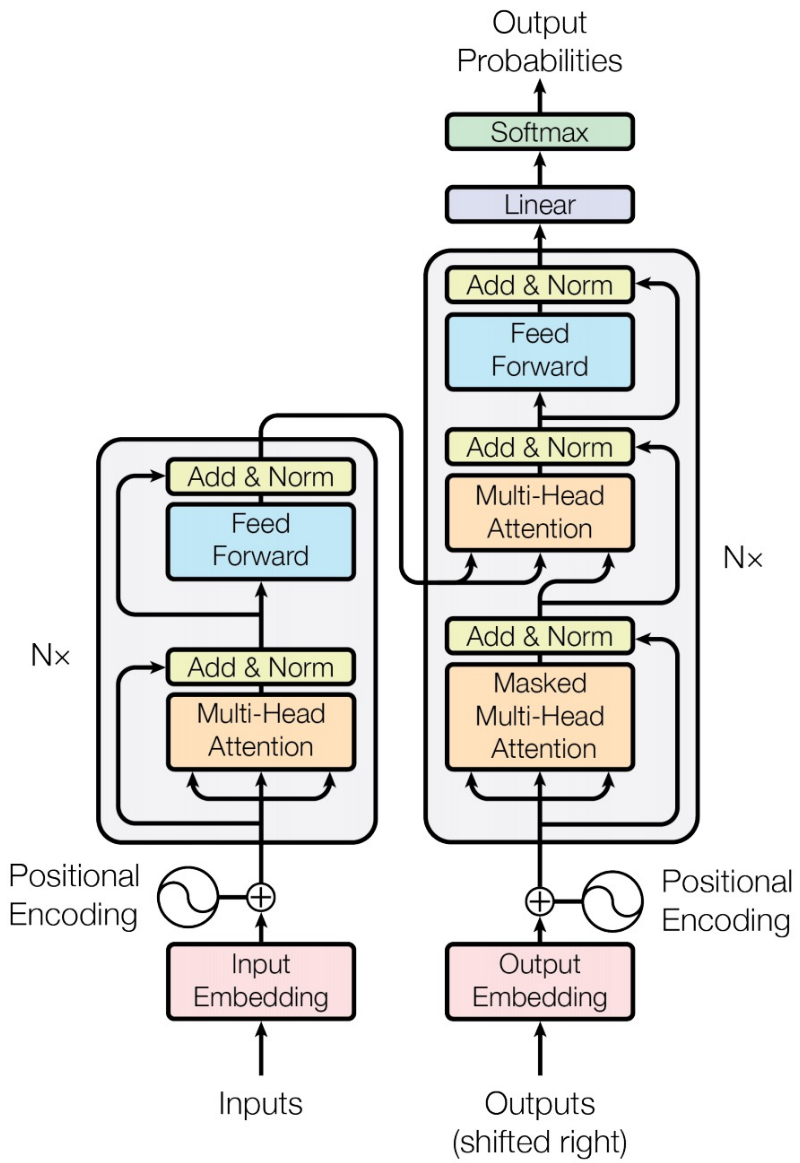
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762



Fixed number of input tokens
[but hey! we can always define a large enough length and add mask tokens]

We can also draw this as in the paper:

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

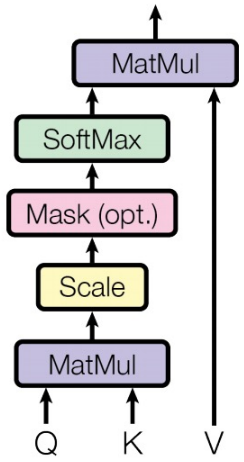


Regular Attention: + Scaling factor

Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

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

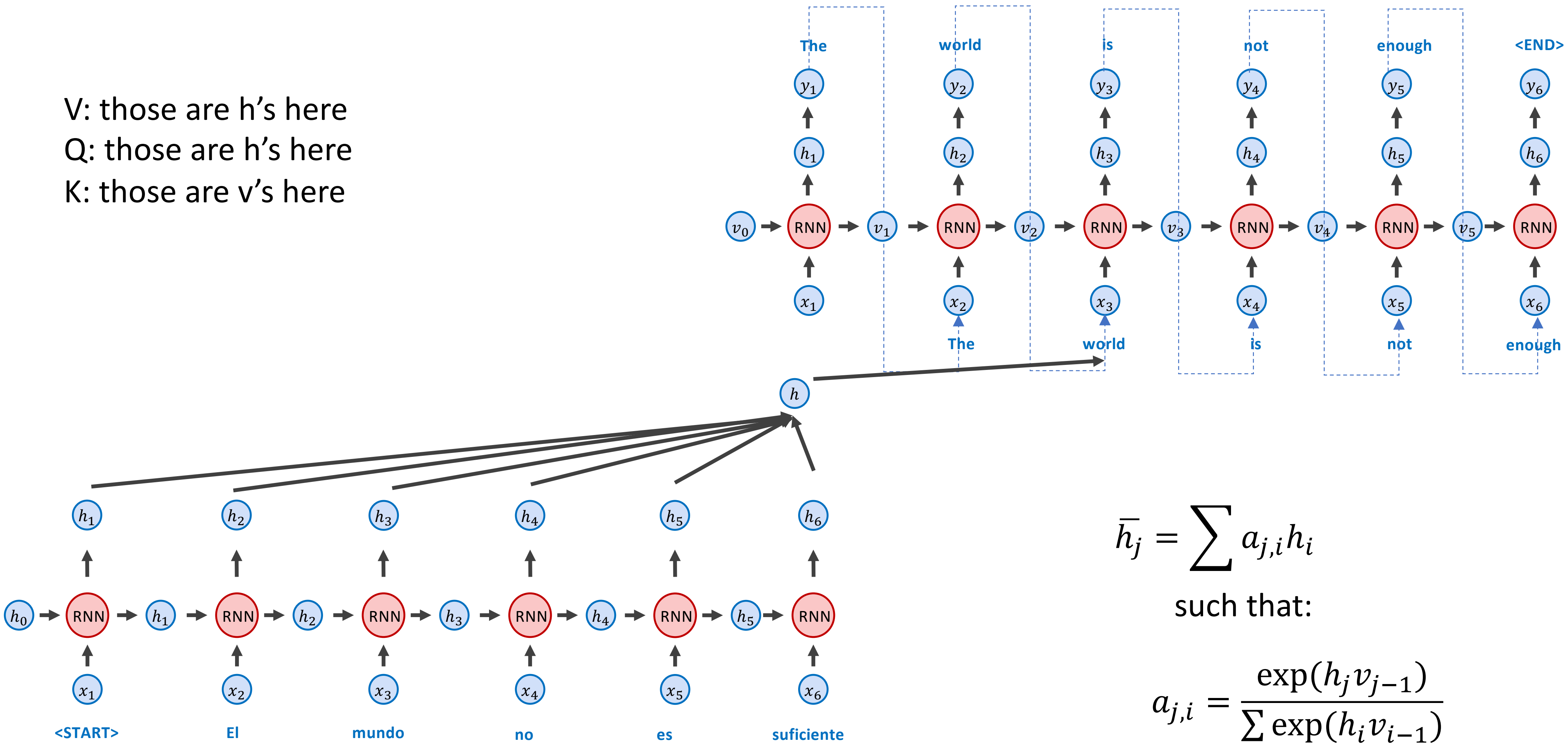
Scaled Dot-Product Attention



This is not unlike what we already used before

Only showing the third time step encoder-decoder connection

V: those are h's here
Q: those are h's here
K: those are v's here



$$\bar{h_j} = \sum a_{j,i} h_i$$

such that:

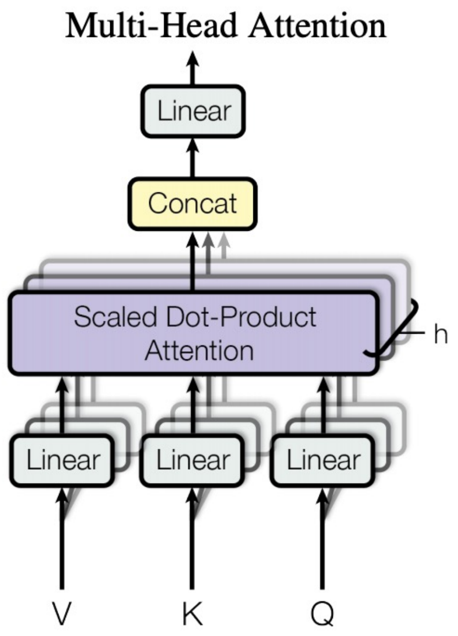
$$a_{j,i} = \frac{\exp(h_j v_{j-1})}{\sum \exp(h_i v_{i-1})}$$

Multi-head Attention: Do not settle for just one set of attention weights.

Vaswani et al. Attention is all you need
https://arxiv.org/abs/1706.03762

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}(\mathrm{head_1}, ..., \mathrm{head_h})W^O$$
$$\text{where } \mathrm{head_i} = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

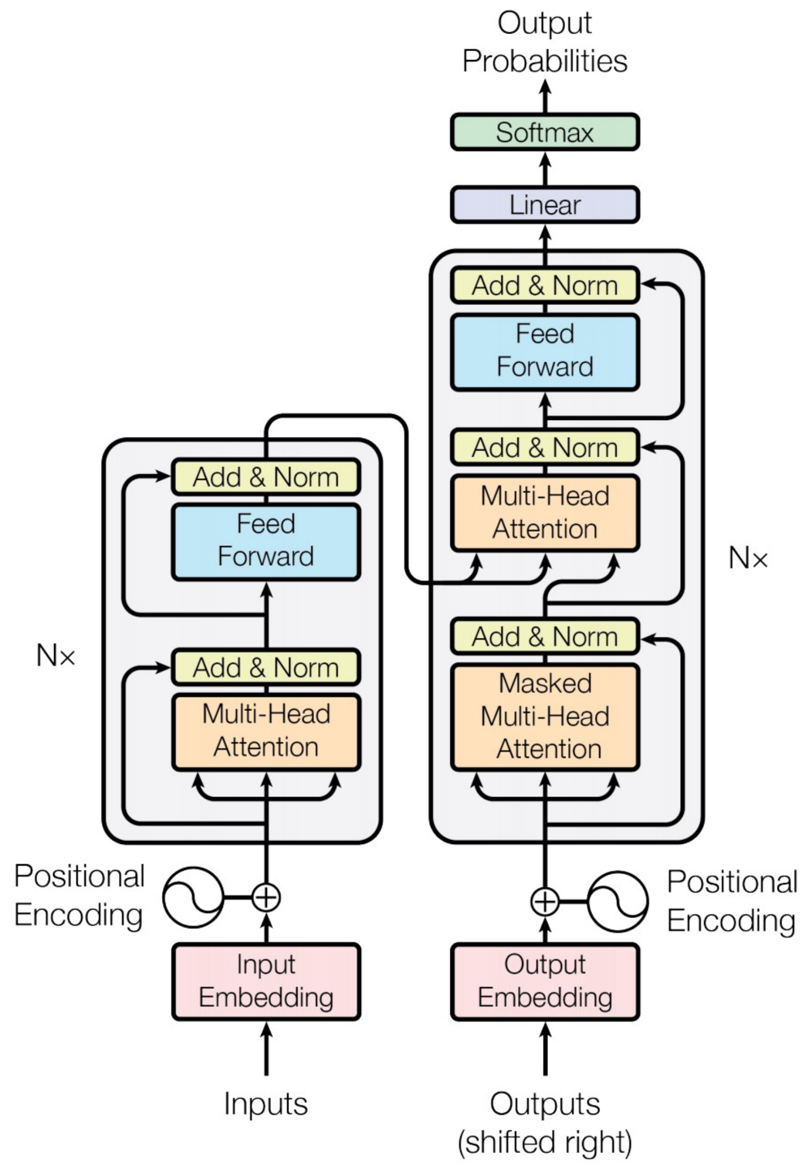
Where the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$.



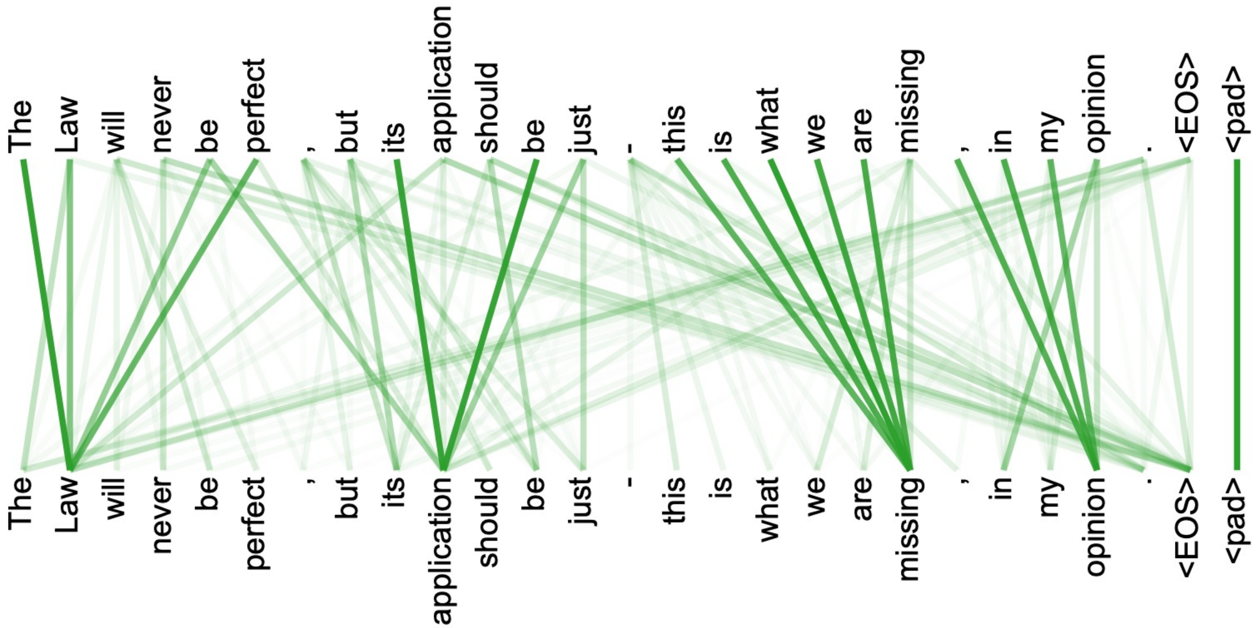
We can lose track of position since we are aggregating across all locations

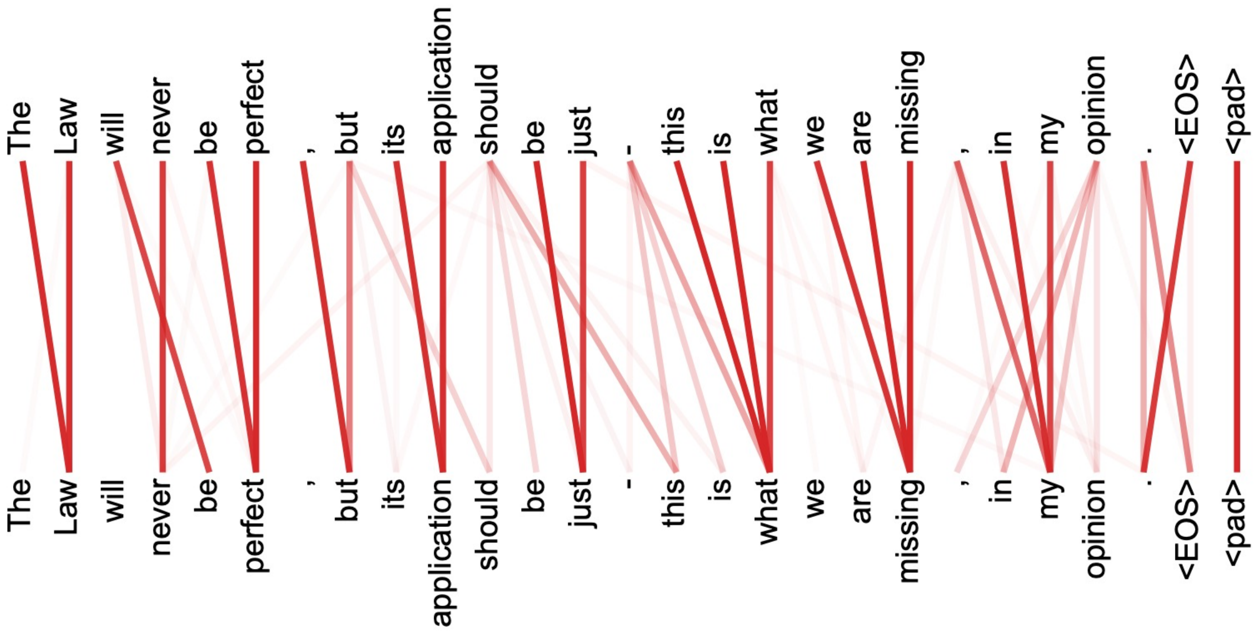
Vaswani et al. Attention is all you need https://arxiv.org/abs/1706.03762

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Multi-headed attention weights are harder to interpret obviously



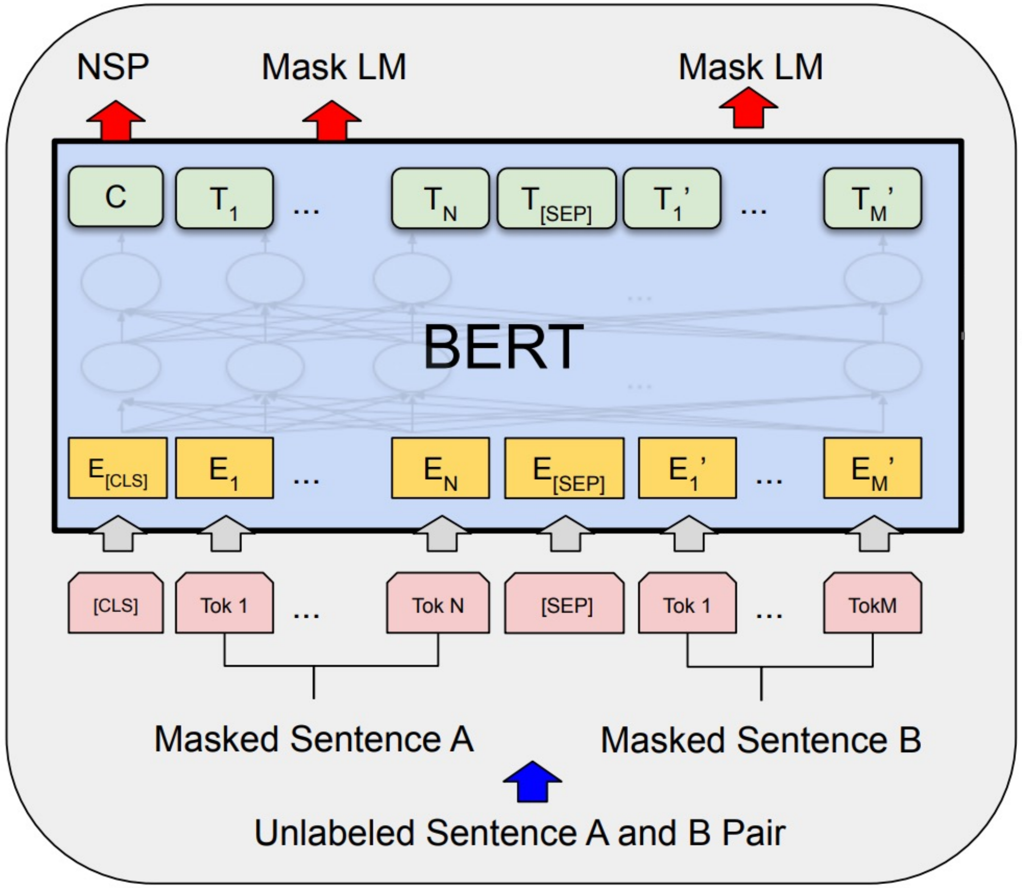


The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805

Important things to know

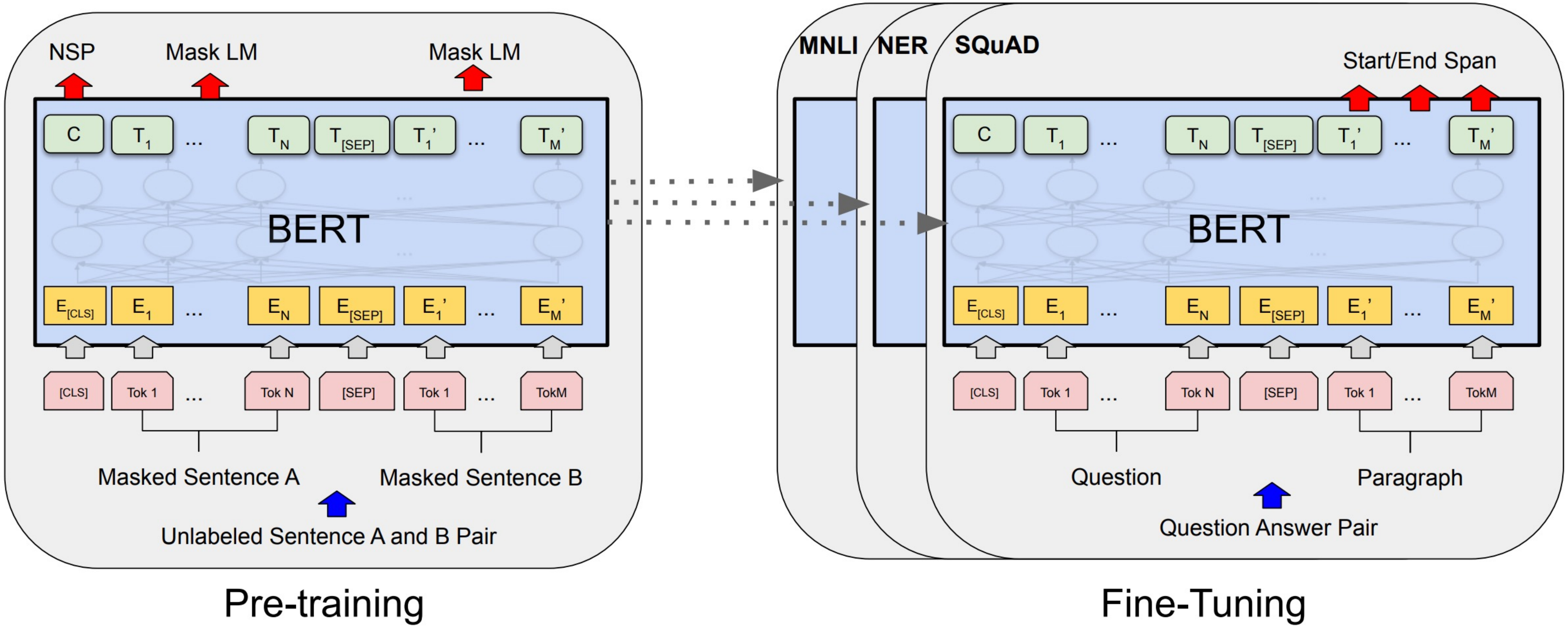
- No decoder
- Train the model to fill-in-the-blank by masking some of the input tokens and trying to recover the full sentence.
- The input is not one sentence but two sentences separated by a [SEP] token.
- Also try to predict whether these two input sentences are consecutive or not.



Pre-training

The BERT Encoder Model

Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding . https://arxiv.org/abs/1810.04805



Questions?