Neural Language Models and BERT

CS 490A, Fall 2020

Applications of Natural Language Processing

https://people.cs.umass.edu/~brenocon/cs490a_f20/

Brendan O’Connor

College of Information and Computer Sciences

University of Massachusetts Amherst

including slides from Mohit Iyyer and Richard Socher

Language Models

- LM = probabilistically predict words
  - We've seen probabilistic word prediction already. Where?
- Use nearby words as context
  - Next-word prediction: give probability to a *sequence*

Why model language?

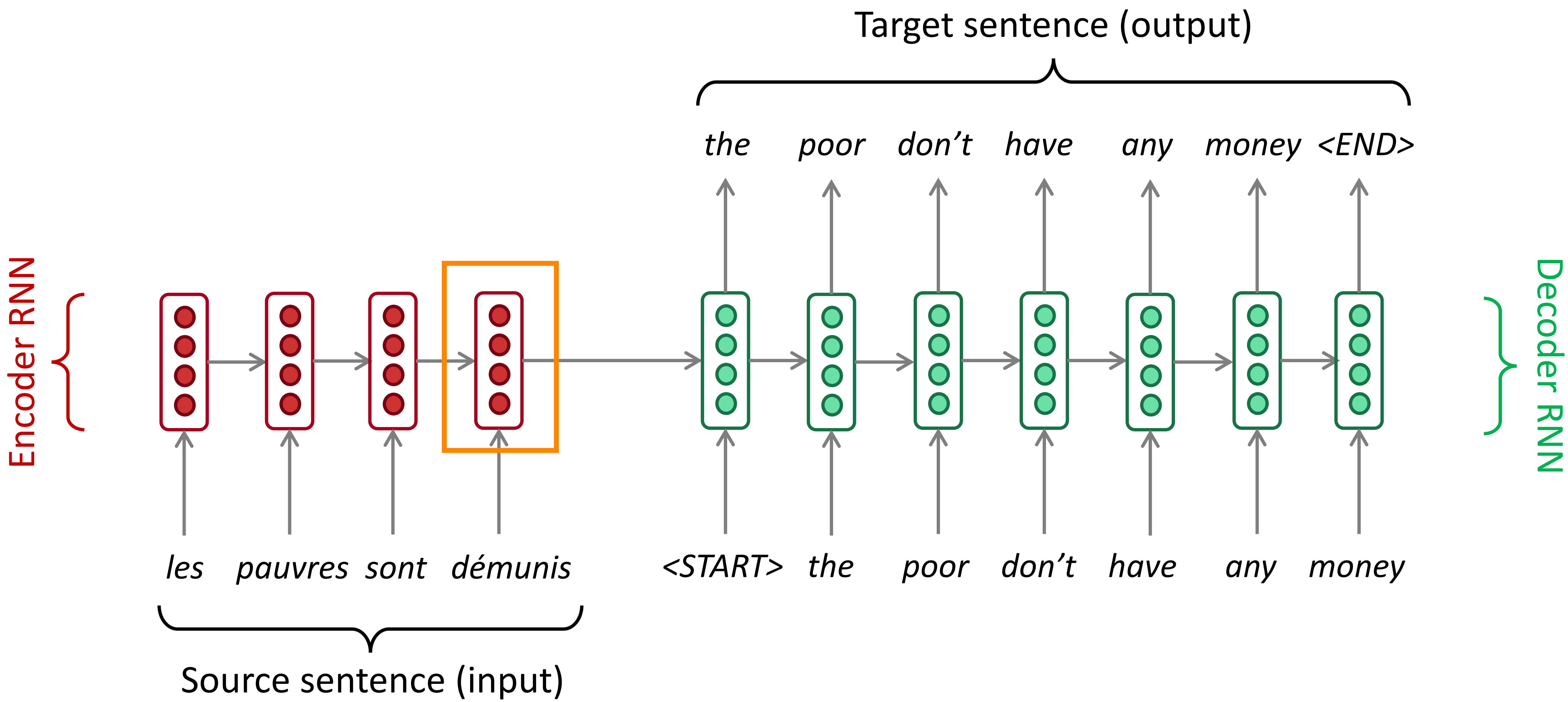
- Train LM -> get *word embeddings*

- LM probabilities for tasks
  - Score quality of proposed translations
  - Predict/score grammatical corrections
  - Generate language

- Train LM, infer on new doc -> get *token embeddings*

Attention mechanisms: background

Sequence-to-sequence: the bottleneck problem

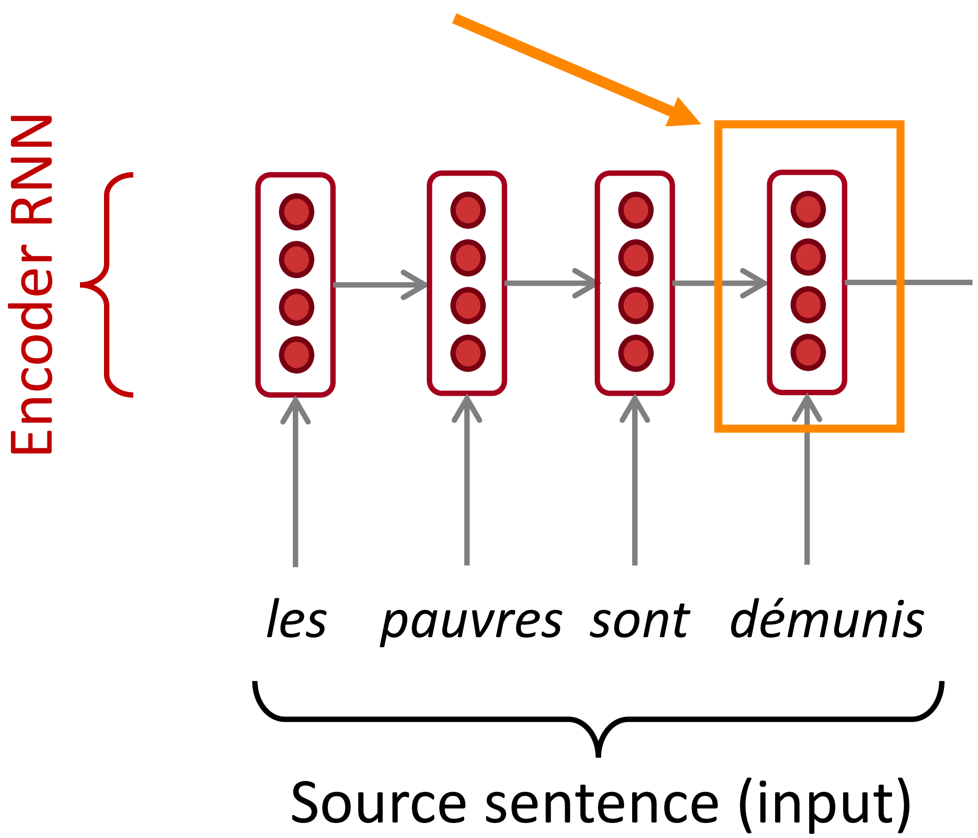


“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!”

— Ray Mooney (famous NLP professor at UT Austin)

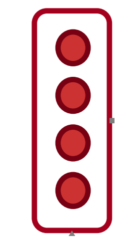
idea: what if we use multiple vectors?

Encoding of the source sentence. This needs to capture *all information* about the source sentence. Information bottleneck!



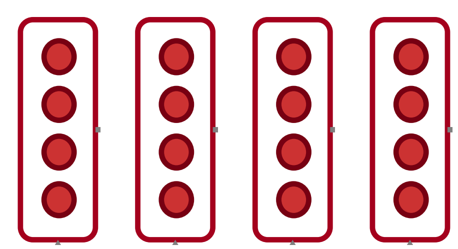
Instead of:

les pauvres sont démunis =



Let's try:

les pauvres sont démunis =



(all 4 hidden states!)

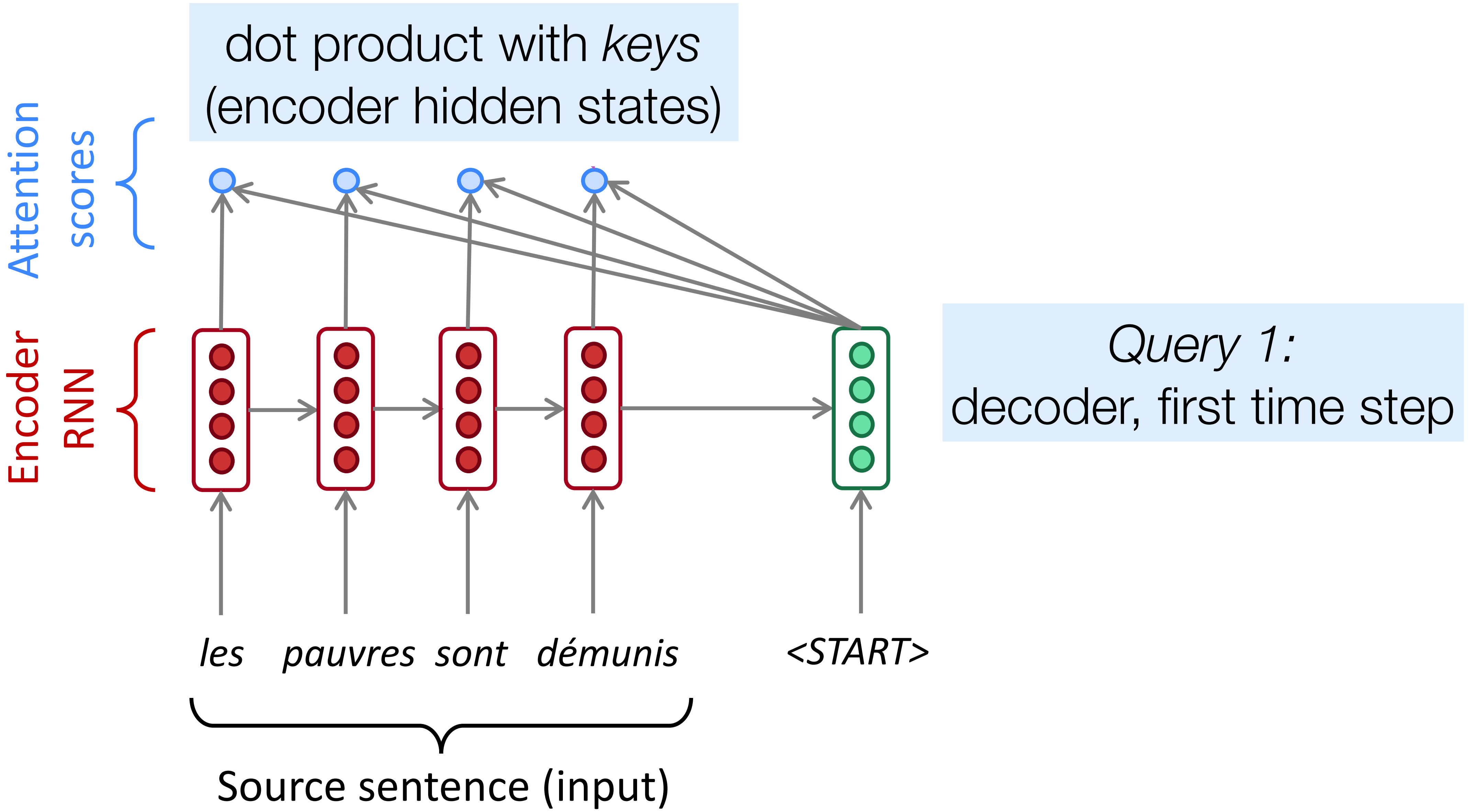
The solution: **attention**

- **Attention mechanisms** (Bahdanau et al., 2015) allow the decoder to focus on a particular part of the source sequence at each time step
  - Conceptually similar to *word alignments*

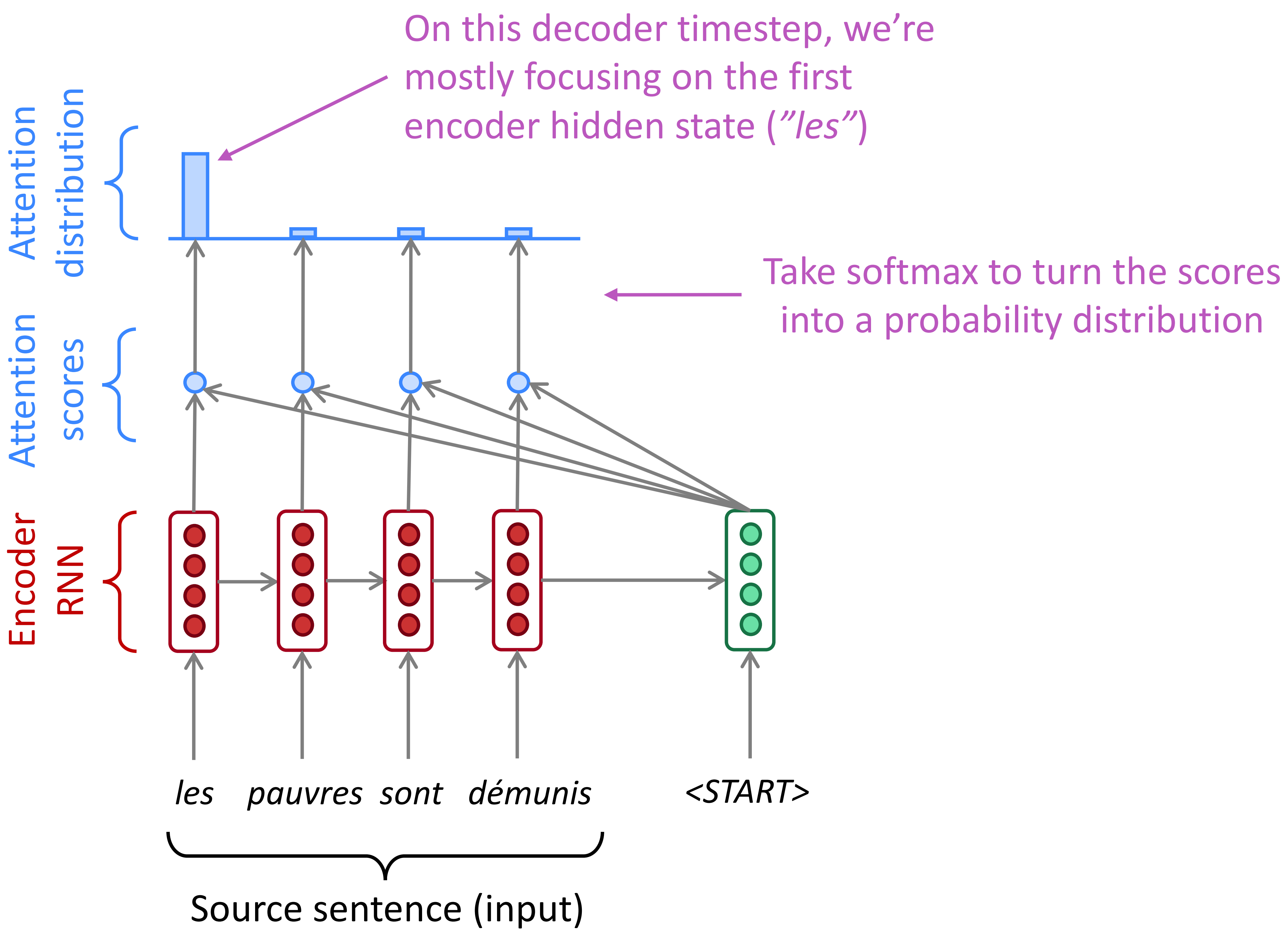
How does it work?

- in general, we have a single *query* vector and multiple *key* vectors. We want to score each query-key pair

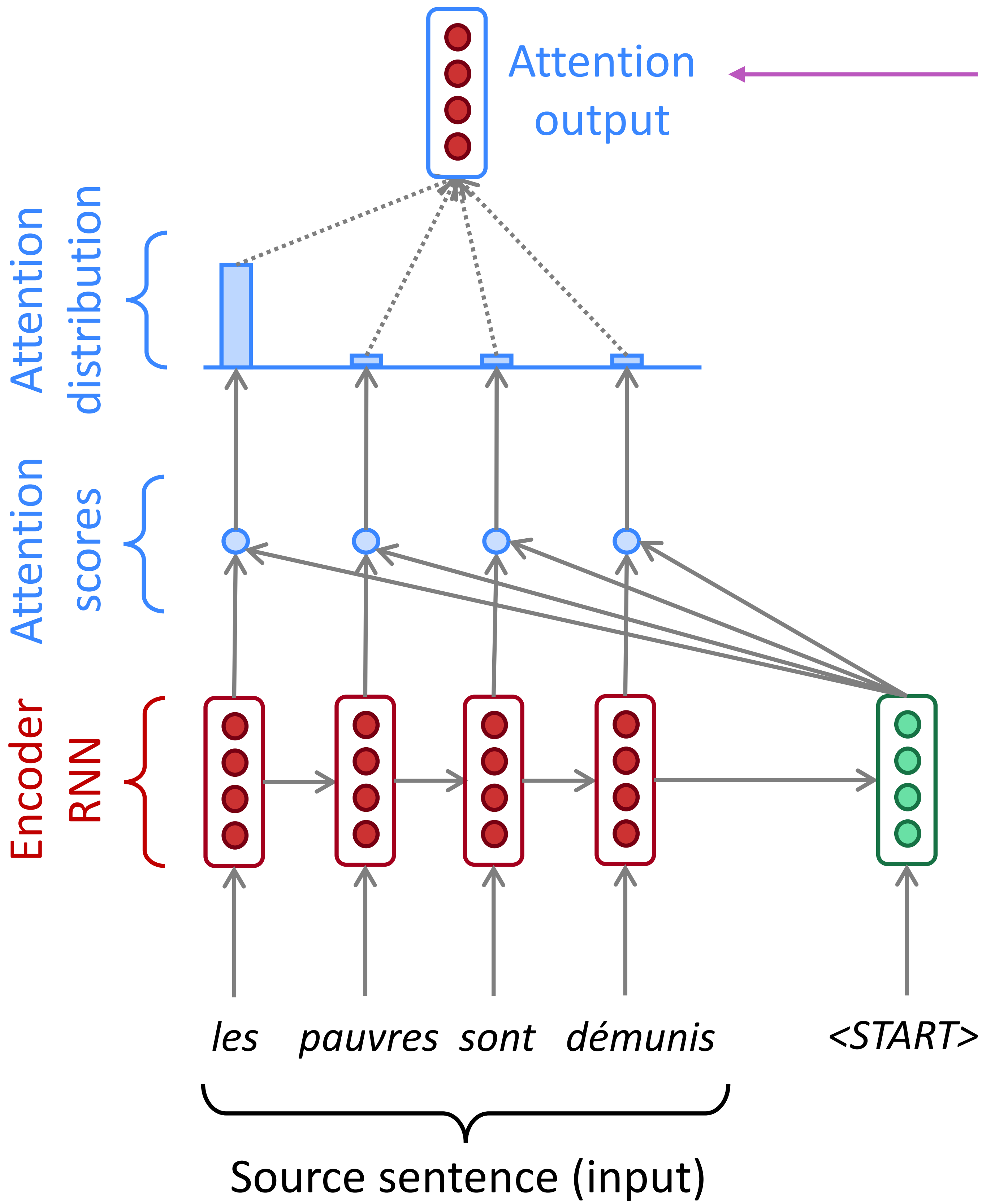
Sequence-to-sequence with attention



Sequence-to-sequence with attention



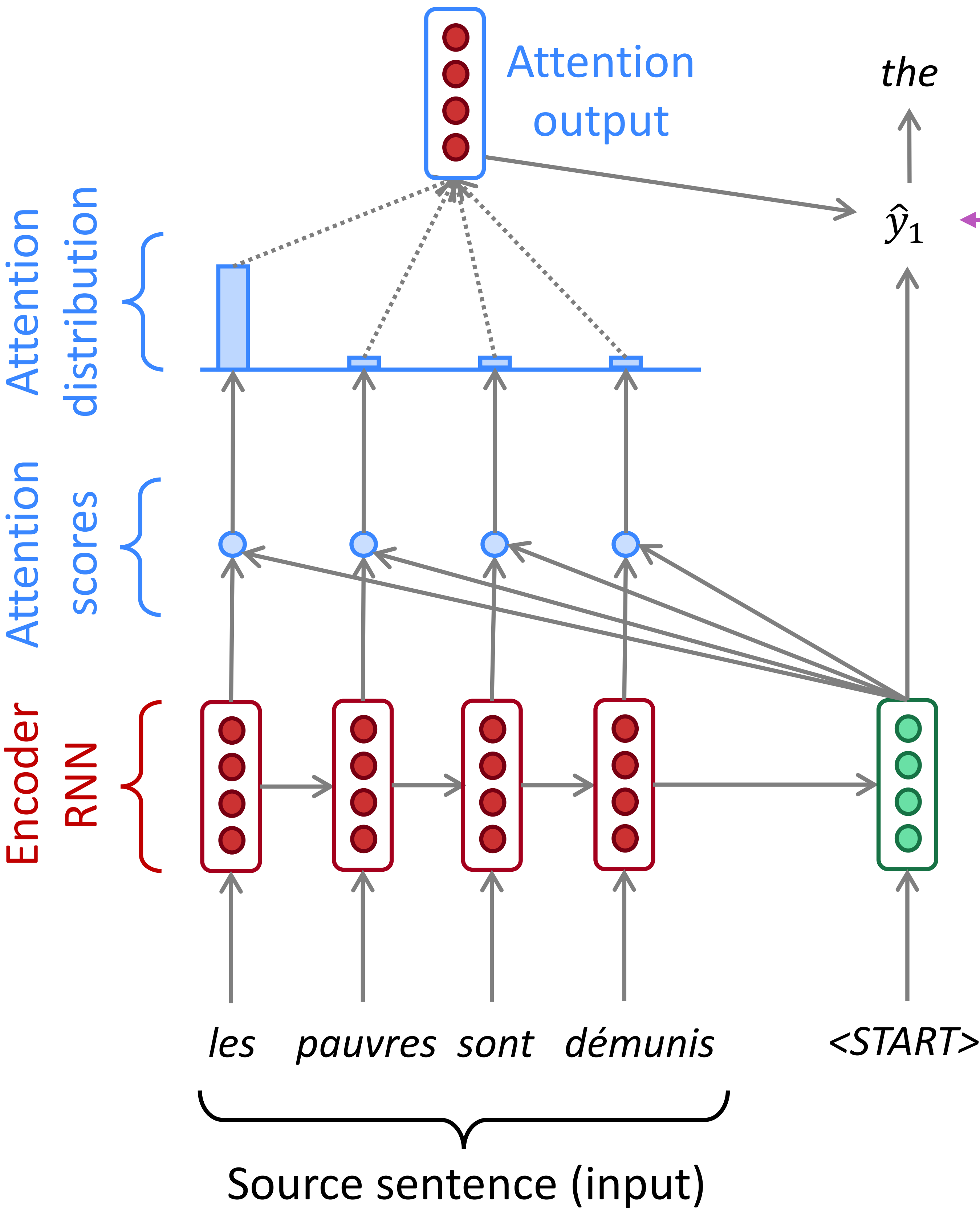
Sequence-to-sequence with attention



Use the attention distribution to take a **weighted sum** of the encoder hidden states.

The attention output mostly contains information the hidden states that received high attention.

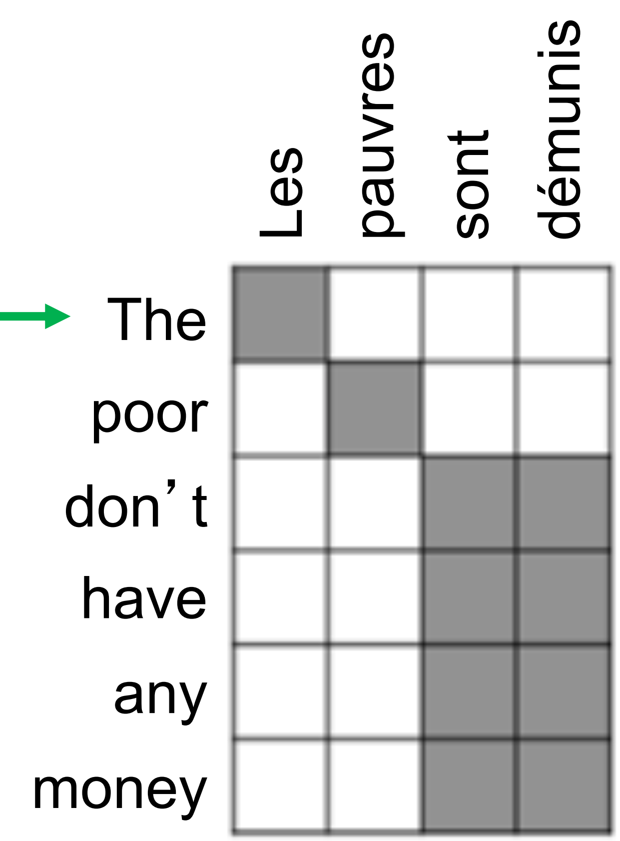
Sequence-to-sequence with attention



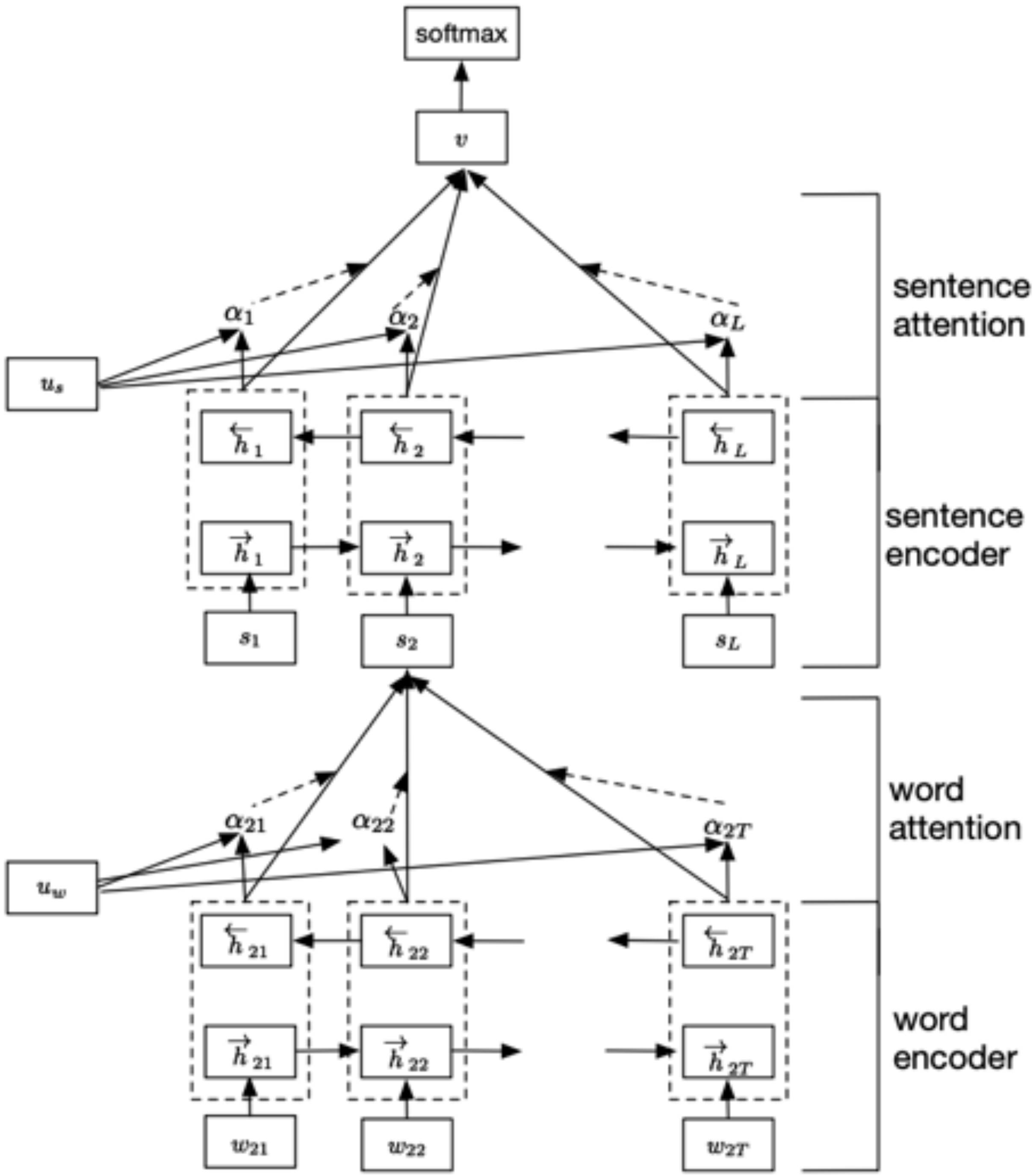
Concatenate attention output with decoder hidden state, then use to compute $\hat{y}_1$ as before

Attention is great

- Attention significantly improves NMT performance
  - It’s very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
  - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
  - Provides shortcut to faraway states
- Attention provides some interpretability
  - By inspecting attention distribution, we can see what the decoder was focusing on
  - We get alignment for free!
  - This is cool because we never explicitly trained an alignment system
  - The network just learned alignment by itself



Hierarchical attention



pork belly = delicious . || scallops? || I don't even like scallops, and these were a-m-a-z-i-n-g . || fun and tasty cocktails. || next time I in Phoenix, I will go back here. || Highly recommend.

Figure 1: A simple example review from Yelp 2013 that consists of five sentences, delimited by period, question mark. The first and third sentence delivers stronger meaning and inside, the word *delicious, a-m-a-z-i-n-g* contributes the most in defining sentiment of the two sentences.

- Yang et al., 2016: hierarchical attention for document classification

Corpus attention

Kelvin Guu
@kelvin_guu

New from Google Research! REALM:
realm.page.link/paper

We pretrain an LM that sparsely attends over all of Wikipedia as extra context. We backprop through a latent retrieval step on 13M docs. Yields new SOTA results for open domain QA, breaking 40 on NaturalQuestions-Open!

Name	Architectures	Pre-training	NQ (79k/4k)	WQ (3k/2k)	CT (1k /1k)	# params
BERT-Baseline (Lee et al., 2019)	Sparse Retr.+Transformer	BERT	26.5	17.7	21.3	110m
T5 (base) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	27.0	29.1	-	223m
T5 (large) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	29.8	32.2	-	738m
T5 (11b) (Roberts et al., 2020)	Transformer Seq2Seq	T5 (Multitask)	34.5	37.4	-	11318m
DrQA (Chen et al., 2017)	Sparse Retr.+DocReader	N/A	-	20.7	25.7	34m
HardEM (Min et al., 2019a)	Sparse Retr.+Transformer	BERT	28.1	-	-	110m
GraphRetriever (Min et al., 2019b)	GraphRetriever+Transformer	BERT	31.8	31.6	-	110m
PathRetriever (Asai et al., 2019)	PathRetriever+Transformer	MLM	32.6	-	-	110m
ORQA (Lee et al., 2019)	Dense Retr.+Transformer	ICT+BERT	33.3	36.4	30.1	330m
Ours ($\mathcal{X}$ = Wikipedia, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	39.2	40.2	**46.8**	330m
Ours ($\mathcal{X}$ = CC-News, $\mathcal{Z}$ = Wikipedia)	Dense Retr.+Transformer	REALM	**40.4**	**40.7**	42.9	330m

6:47 PM · Feb 11, 2020 · Twitter Web App

Transformers: Self-attention

Attention for LM

BERT

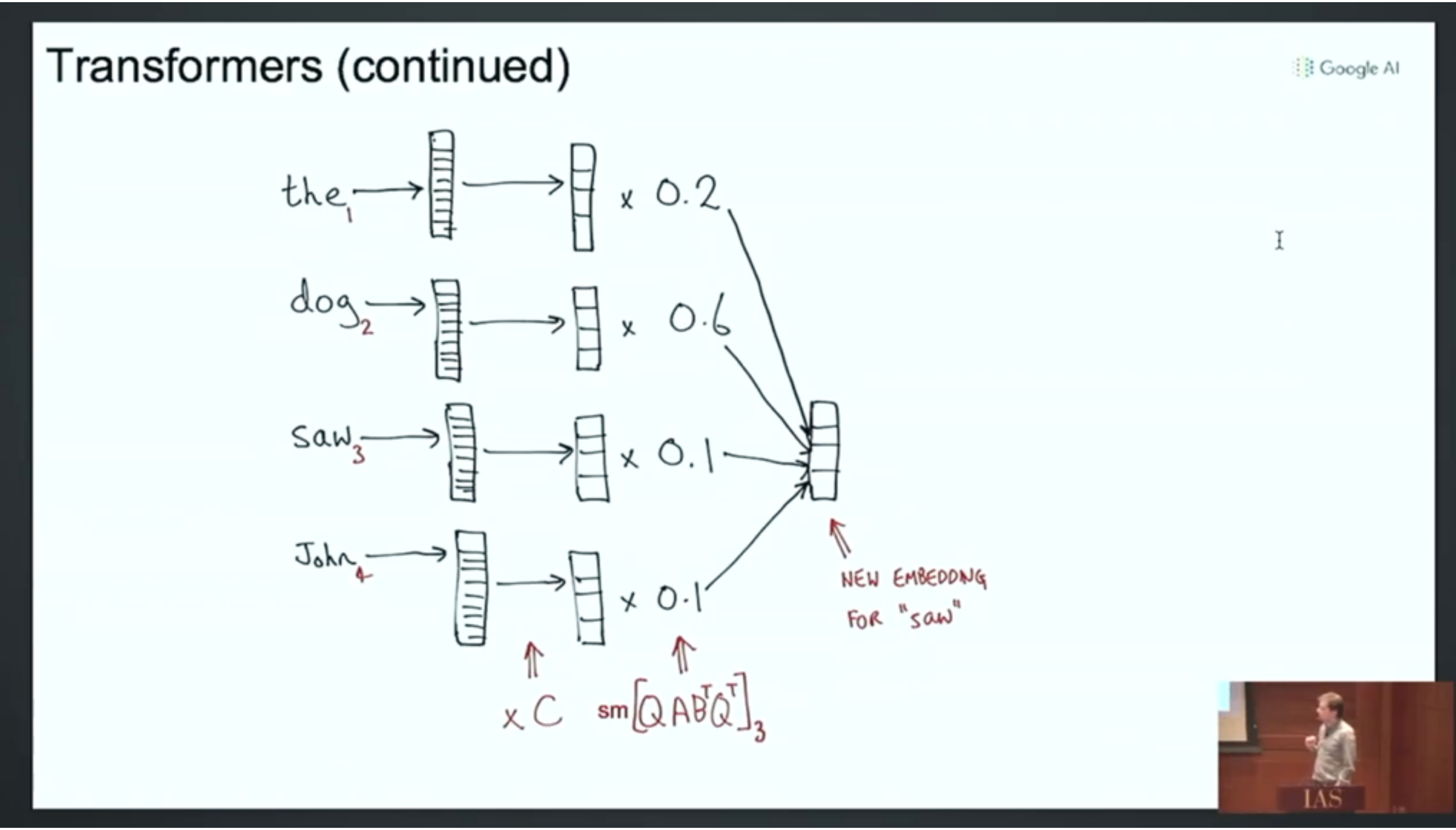
- "Bidirectional… Transformers"
  - Transformer: a specific neural net architecture for token sequences, that uses attention and token embeddings
  - Bidirectional: The core model is a *masked LM*, predicting missing word(s) from rest of words in sentence
- Intended for *pretraining* pipeline
  - Initially train on a gazillion documents (using a GPU-days)
  - Then apply pretrained model on *new* data to calculate *token*-level embeddings. (No word prediction at all any more!) They turn out to be useful!
- BERT (+ variants) is incredibly successful at many classification, tagging, and generation tasks
  - This space changes very rapidly, so who knows how long it's SOTA. Two years is longer than I would have guessed though?

Transformers (Attention is All You Need, Vaswani et al. 2017)

Google AI

- Assume we have a sequence of words $w_1 \ldots w_n$
- We can map this to a sequence of vectors $x_1 \ldots x_n$ where each $x_i \in \mathbb{R}^d$ (e.g., $d = 512$), and each x_i is the word embedding for w_i
- How do we map this to a new sequence $z_1 \ldots z_n$ where each $z_i \in \mathbb{R}^d$, where z_i's now take context into account?

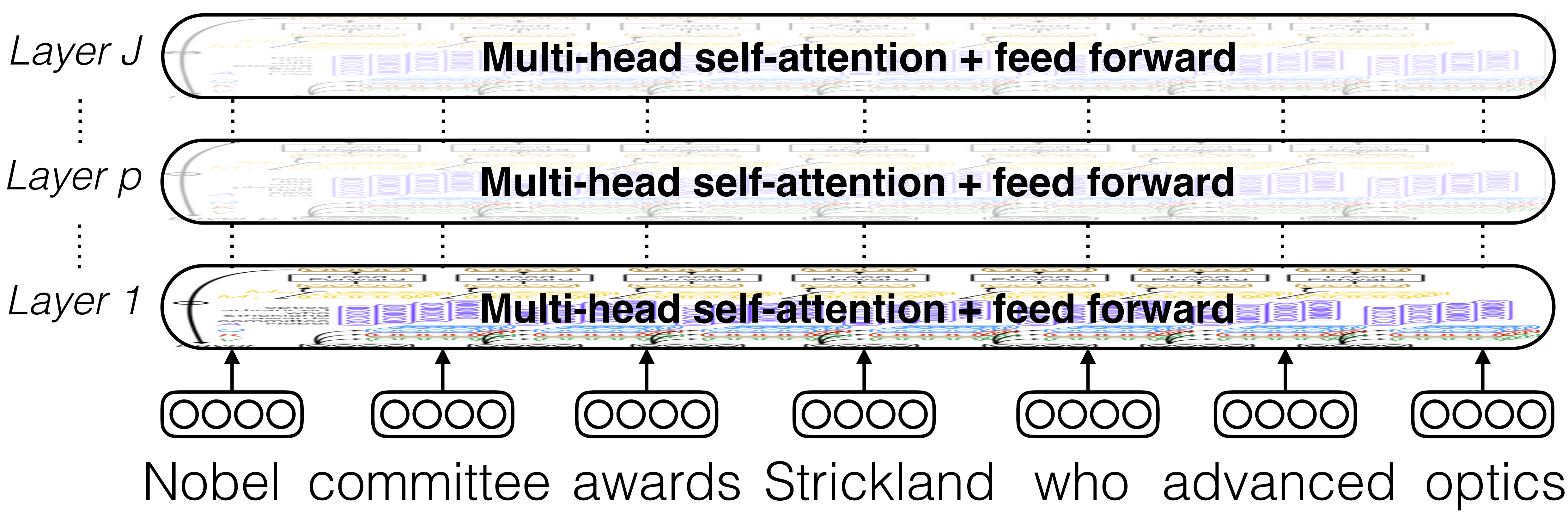
[Michael Collins 2019 lecture]



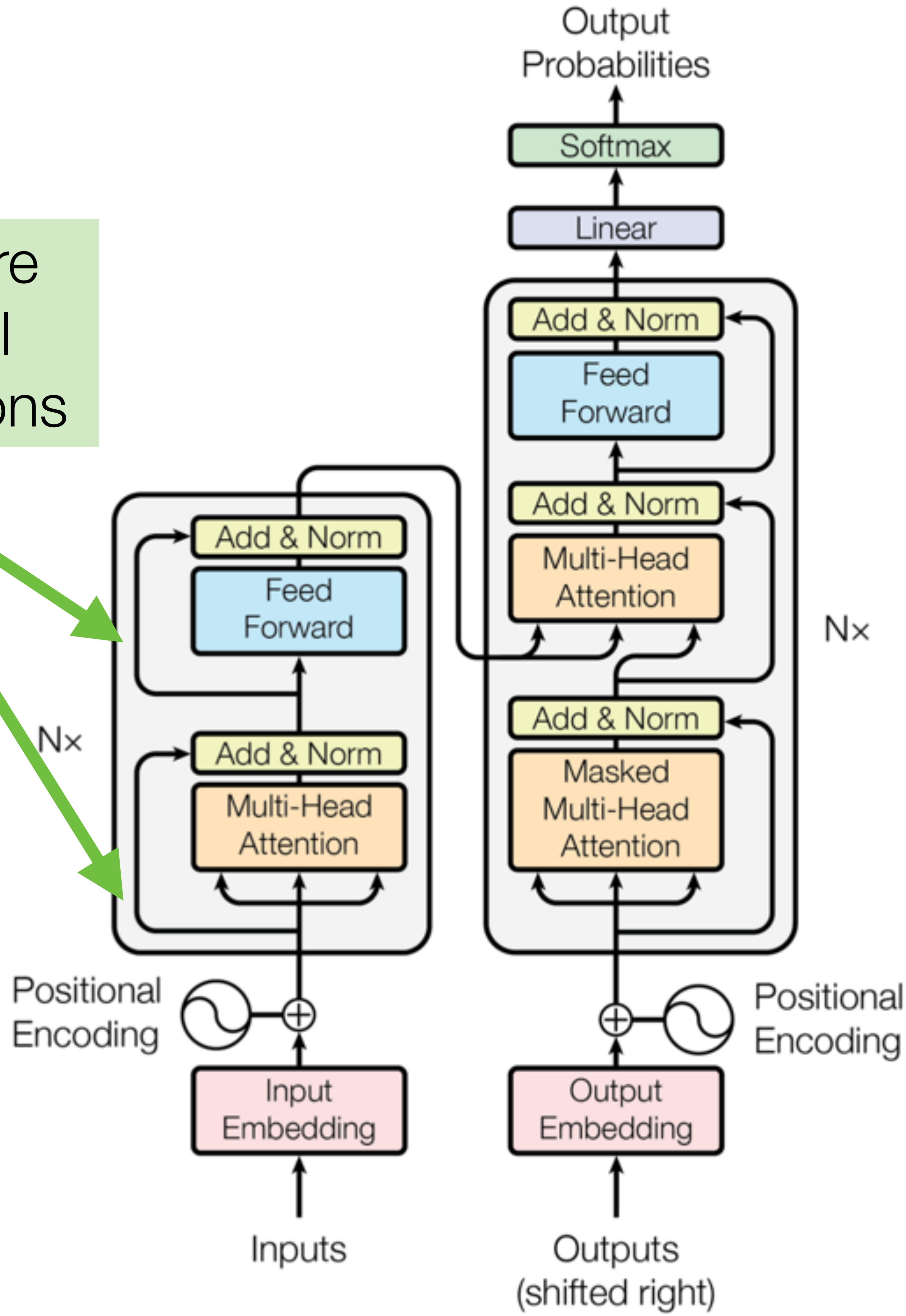
[Michael Collins 2019 lecture]

Multi-head self-attention

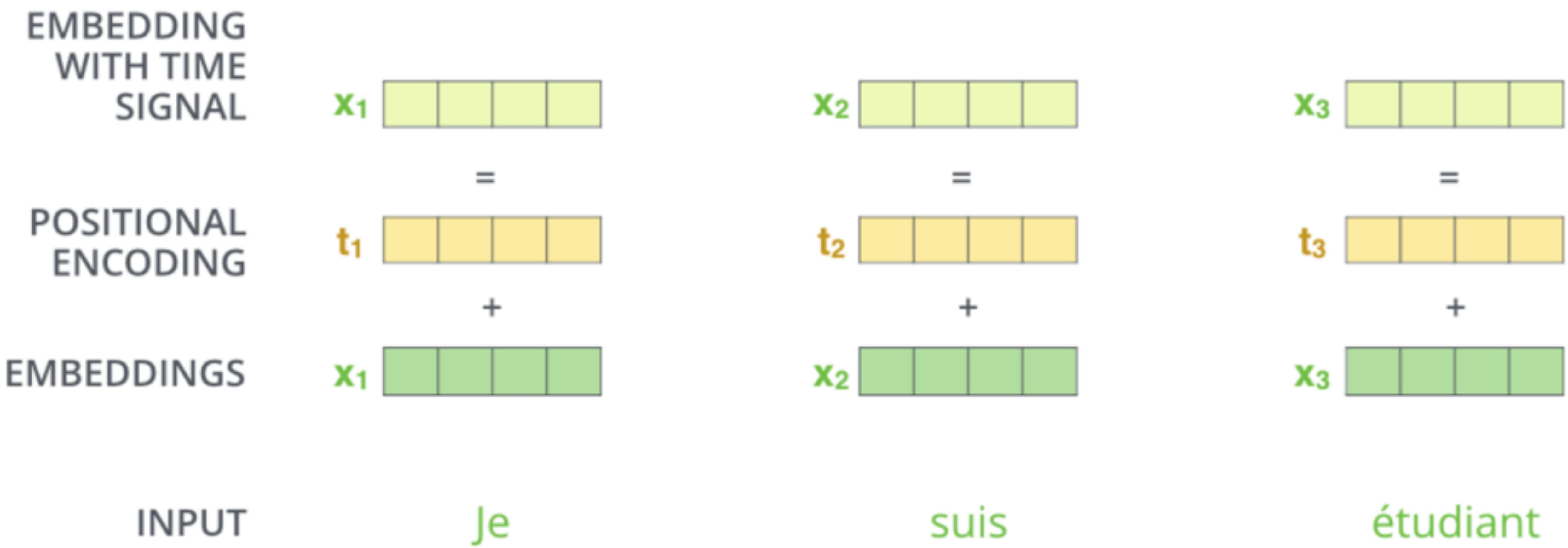
[Vaswani et al. 2017 original notation]



These are residual connections



Positional encoding

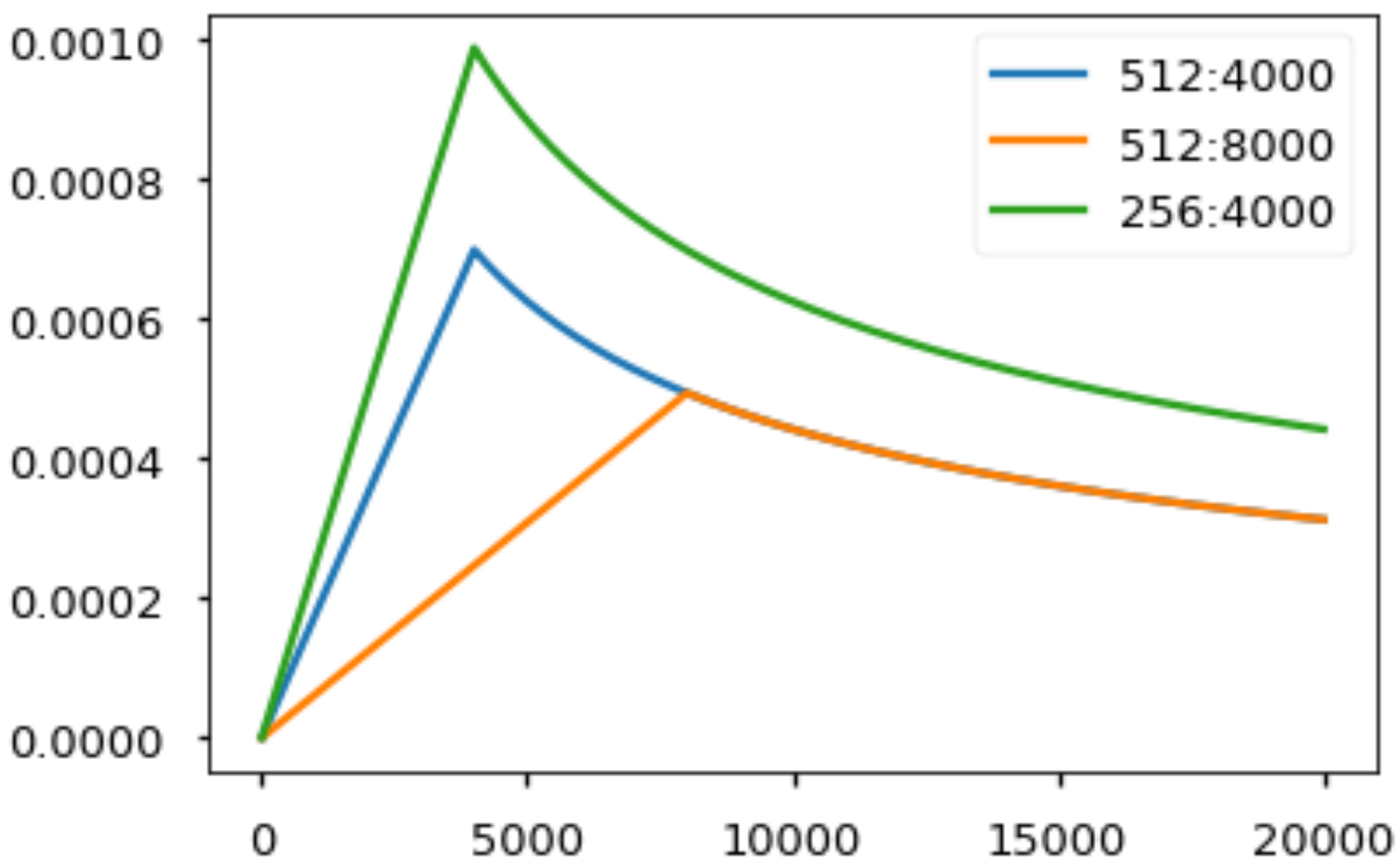


Hacks to get it to work:

Optimizer

We used the Adam optimizer (cite) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula: $lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$ This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

> *Note: This part is very important. Need to train with this setup of the model.*



Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has* `confidence` *of the correct word and the rest of the* `smoothing` *mass distributed throughout the vocabulary.*

I went to class and took ___

cats	TV	notes	took	sofa
0	0	1	0	0
0.025	0.025	0.9	0.025	0.025

with label smoothing

Byte pair encoding (BPE)

- Deal with rare words / large vocabulary by using *subword* tokenization
  - Initial analysis step iteratively merges frequent character n-grams to form the vocabulary
  - Confusing name comes from data compression literature - not actually about bytes for us

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
WDict	Forschungsinstitute
C2-50k	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE-60k	Gesundheits|forsch|ungsinstitu|ten
BPE-J90k	Gesundheits|forsch|ungsin|stitute
source	asinine situation
reference	dumme Situation
WDict	asinine situation → UNK → asinine
C2-50k	as|in|in|e situation → As|in|en|si|tu|at|io|n
BPE-60k	as|in|ine situation → A|in|line-|Situation
BPE-J90K	as|in|ine situation → As|in|in-|Situation

Sennrich et al., ACL 2016

Using BERT

- You get
  - Per-token embeddings
  - Multiple layers at each
  - Embedding for per-sentence “[CLS]” symbol
- Use as input for tasks. Two learning approaches
  - “Frozen”: use them as input features
  - Fine-tuning: backprop through the actual BERT model itself

- Many pretrained BERT or BERT-like models are available (especially for English and other high-resource languages…)
- Check out HuggingFaces' examples
    - https://huggingface.co/transformers/examples.html
- Many other frameworks too - e.g. AllenNLP

Fine-Tuning Pretrained Language Models: Weight Initializations, Data Orders, and Early Stopping

Jesse Dodge [1,2] **Gabriel Ilharco** [3] **Roy Schwartz** [2,3] **Ali Farhadi** [2,3,4] **Hannaneh Hajishirzi** [2,3] **Noah Smith** [2,3]

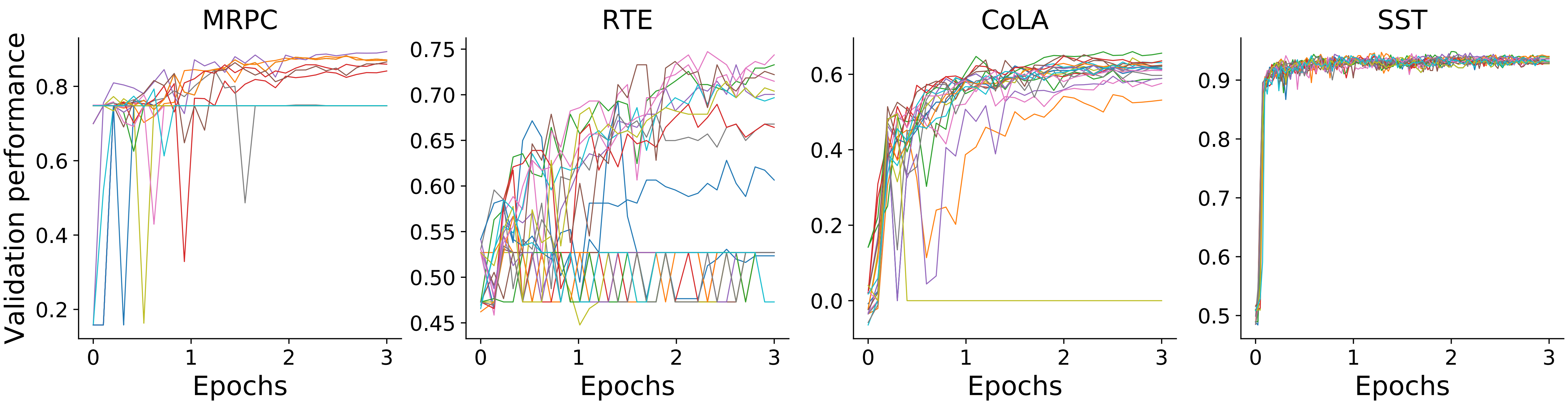
Abstract

Fine-tuning pretrained contextual word embedding models to supervised downstream tasks has become commonplace in natural language processing. This process, however, is often brittle: even with the same hyperparameter values, distinct random seeds can lead to substantially different results. To better understand this phenomenon, we experiment with four datasets from the GLUE benchmark, fine-tuning BERT hundreds of times on each while varying only the random seeds. We

On small datasets, we observe that many fine-tuning trials diverge part of the way through training, and we offer best practices for practitioners to stop training less promising runs early. We

	MRPC	RTE	CoLA	SST
BERT (Phang et al., 2018)	90.7	70.0	62.1	92.5
BERT (Liu et al., 2019)	88.0	70.4	60.6	93.2
BERT (ours)	**91.4**	**77.3**	**67.6**	**95.1**
STILTs (Phang et al., 2018)	90.9	83.4	62.1	93.2
XLNet (Yang et al., 2019)	89.2	83.8	63.6	95.6
RoBERTa (Liu et al., 2019)	90.9	86.6	68.0	96.4
ALBERT (Lan et al., 2019)	90.9	89.2	71.4	96.9

Table 1. Fine-tuning BERT multiple times while varying only random seeds leads to substantial improvements over previously published validation results with the same model and experimental



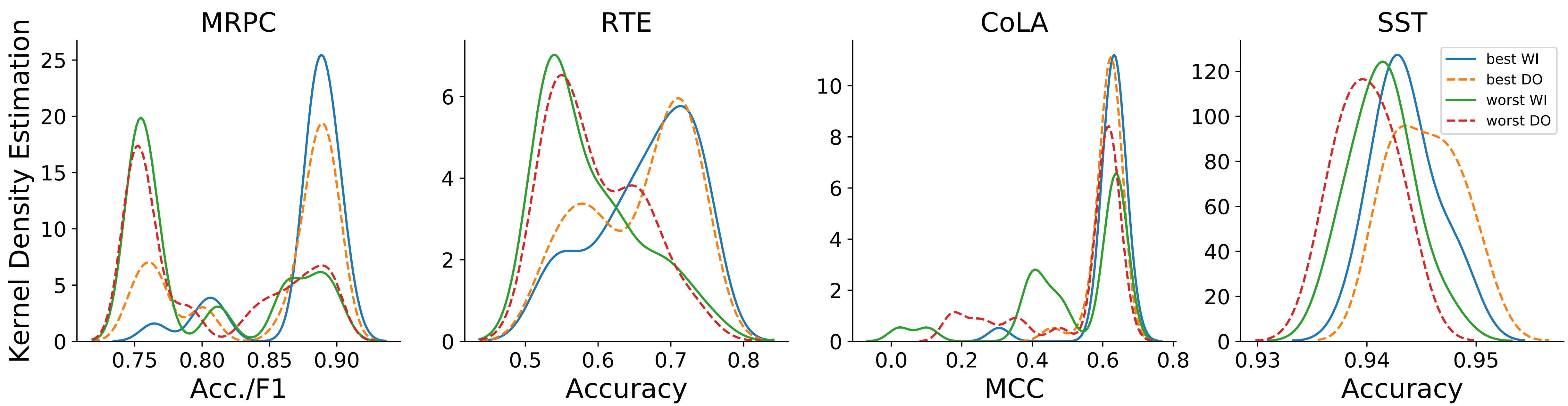


Figure 3. Some seeds are better then others. Plots show the kernel density estimation of the distribution of validation performance for best and worst WI and DO seeds. Curves for DO seeds are shown in dashed lines and for WI in solid lines. MRPC and RTE exhibit pronounced bimodal shapes, where one of the modes represents divergence; models trained with the worst WI and DO are more likely to diverge than learn to predict better than random guessing. Compared to the best seeds, the worst seeds are conspicuously more densely populated in the lower performing regions, for all datasets.