# Transformers and BERT

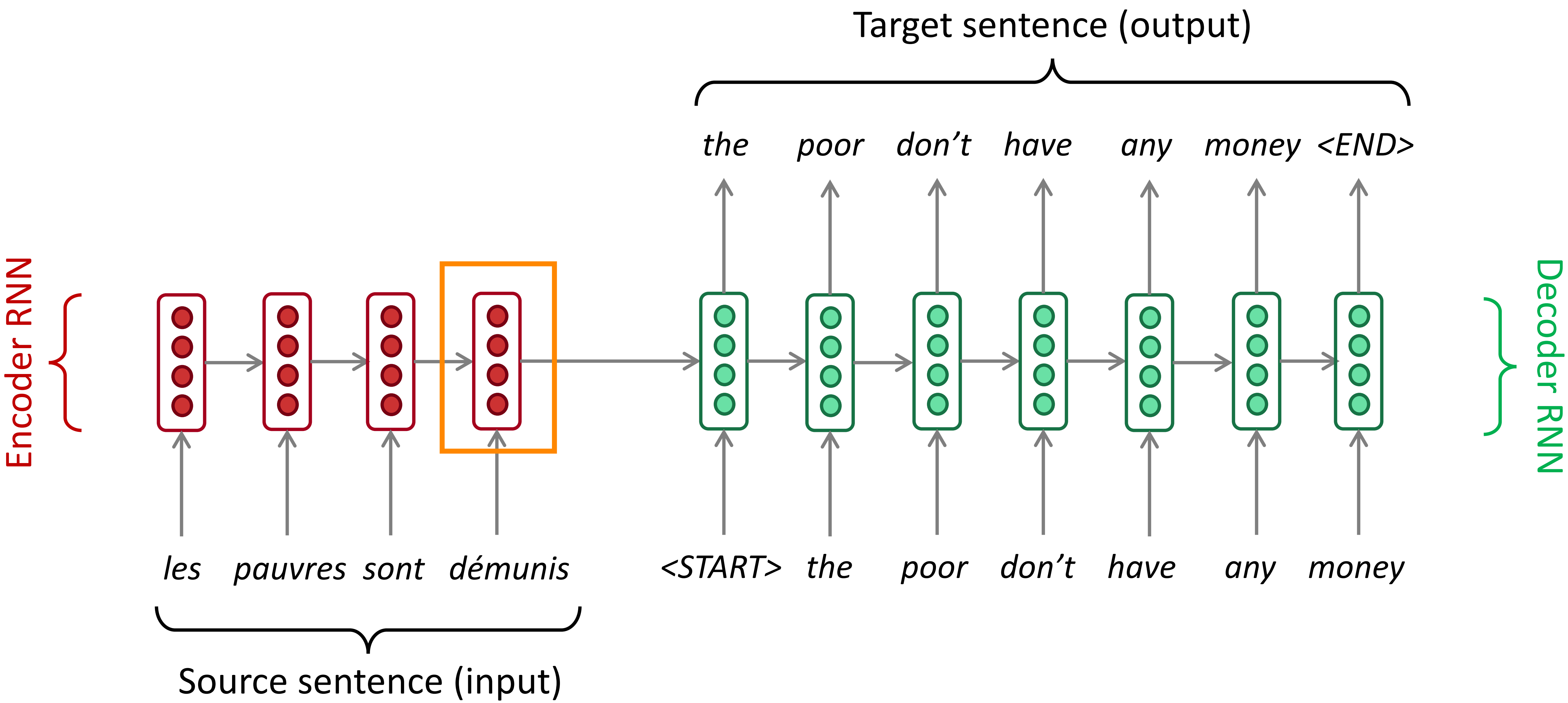
CS 485, Fall 2024
Applications of Natural Language Processing

Brendan O'Connor
College of Information and Computer Sciences
University of Massachusetts Amherst

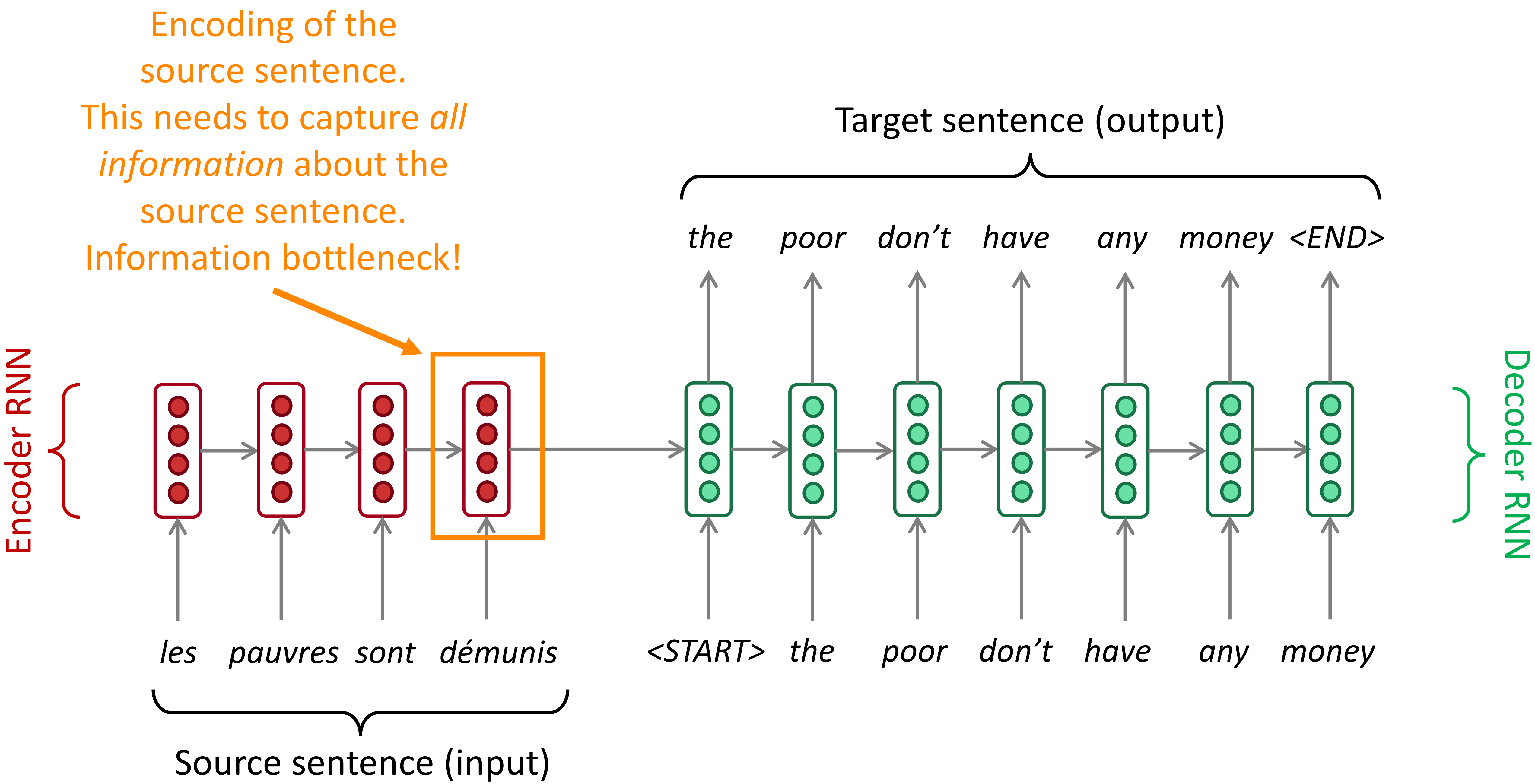
*[Incl. slides from Mohit Iyyer, Richard Socher]*

- Today: introduce "Transformer" network architecture
  - *Attention* for generative LMs, motivated by MT
  - *Self-attention* + feedforward for token embeddings

# Sequence-to-sequence: the bottleneck problem



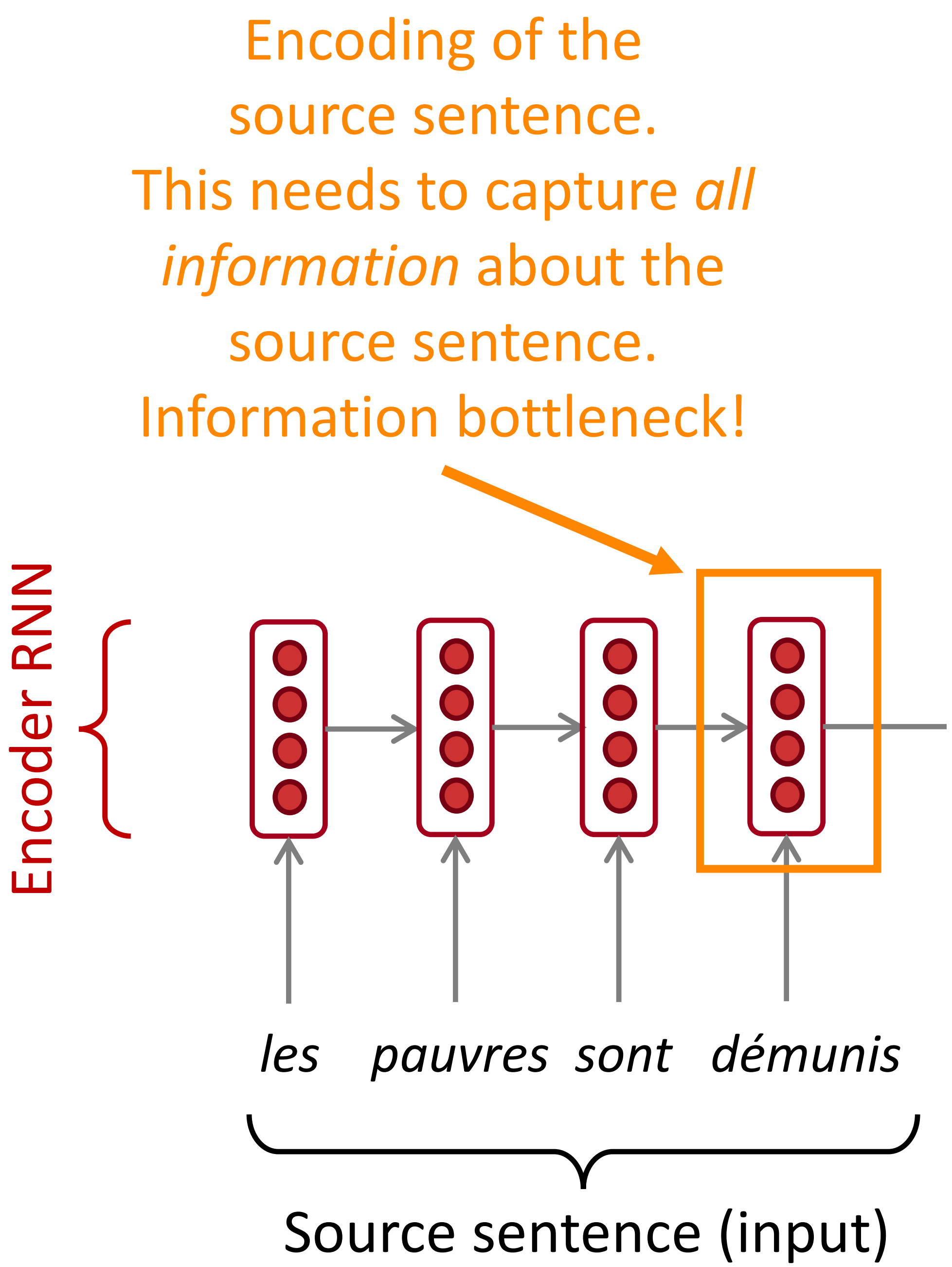
# Sequence-to-sequence: the bottleneck problem



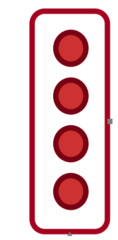
“you can’t cram the meaning of a whole %&@#&ing sentence into a single $*(&@ing vector!”

— Ray Mooney (famous NLP professor at UT Austin)

# idea: what if we use multiple vectors?

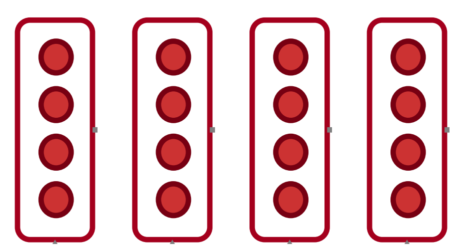



**Instead of:**

les pauvres sont démunis = 

**Let's try:**

les pauvres sont démunis =

 (all 4 hidden states!)

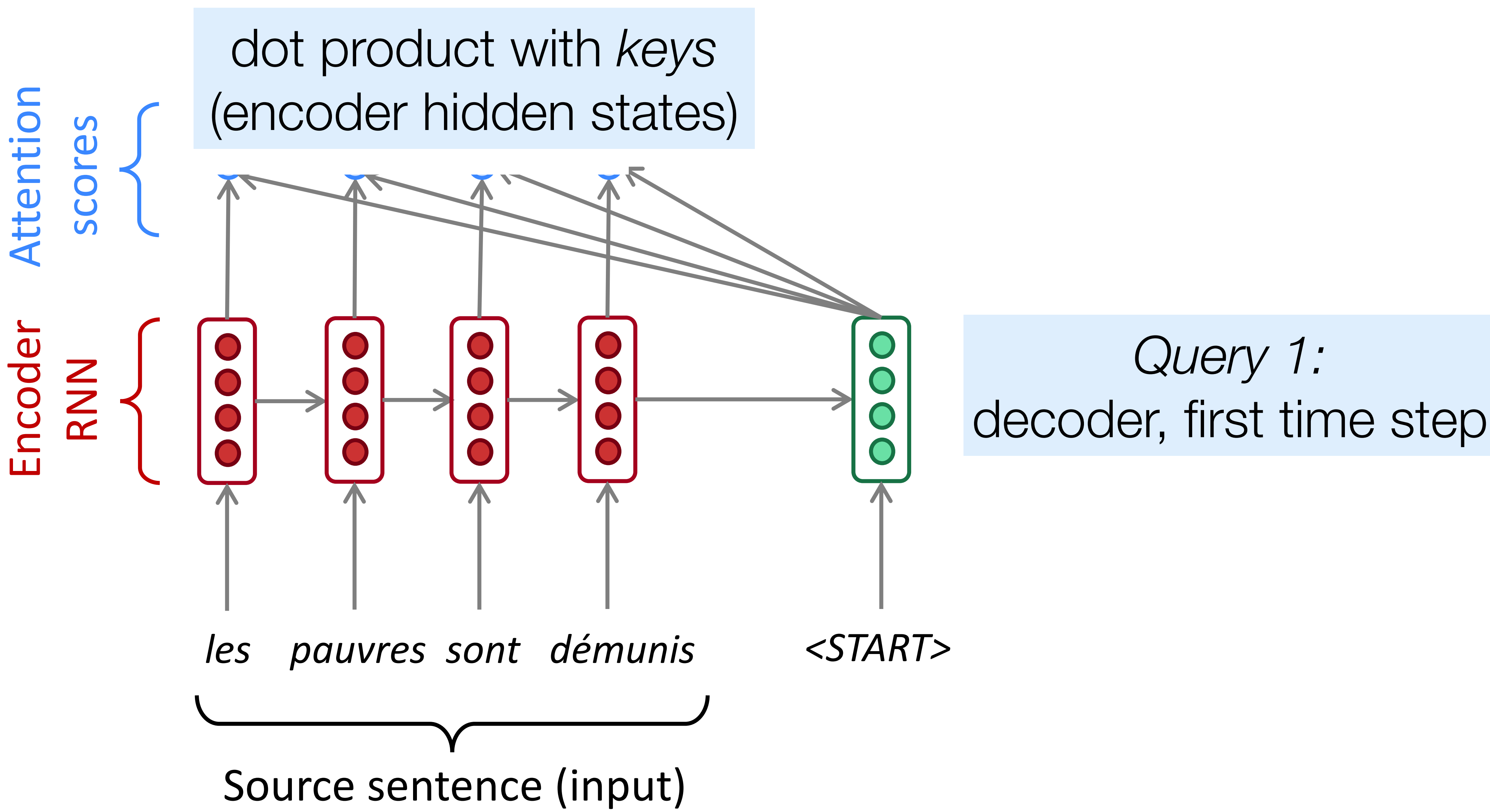
# The solution: **attention**

- **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* in earlier MT models
    - For MT, can model differences in word order between languages

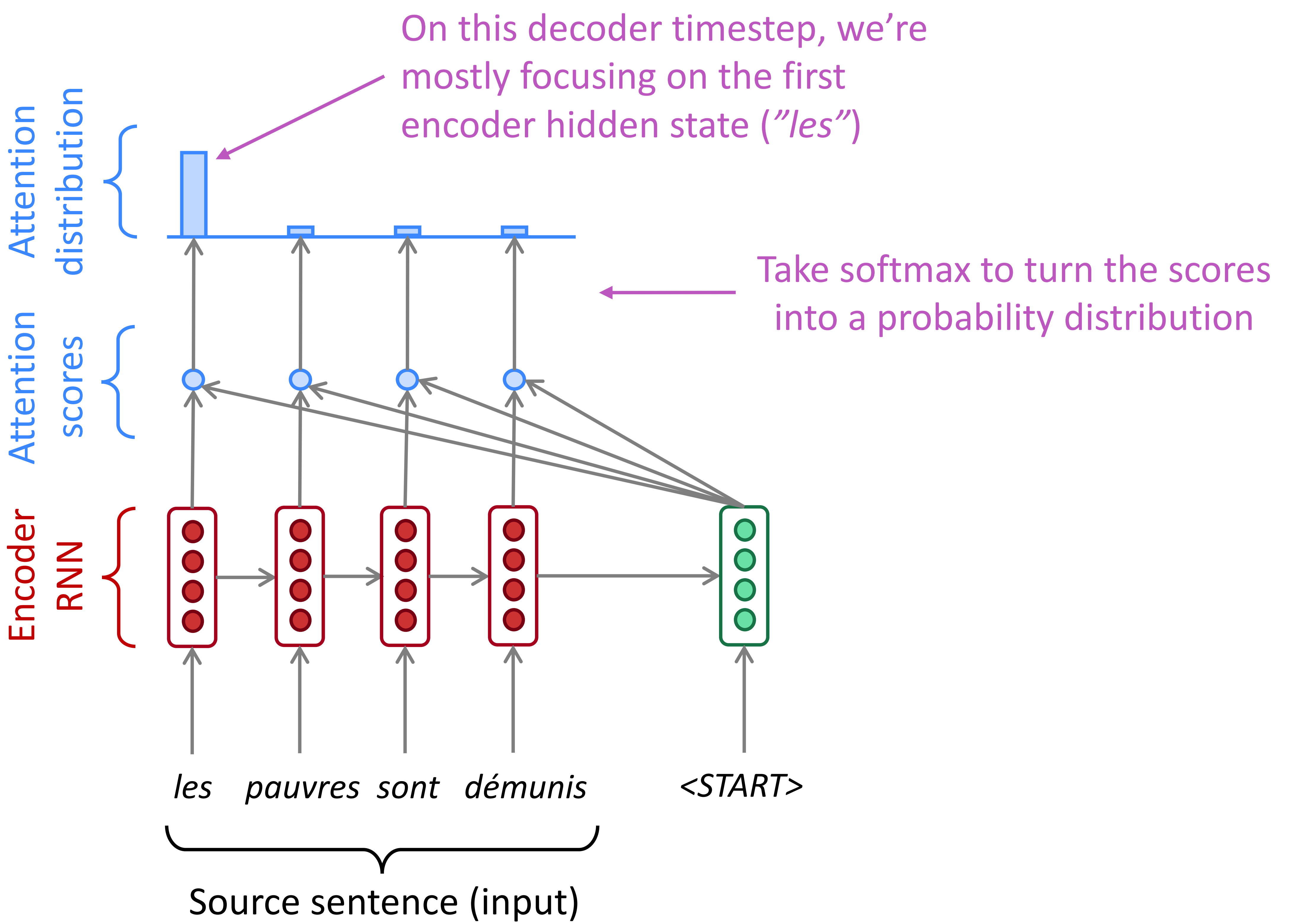
# How does it work?

- in general, we have a single *query* vector and multiple *key* vectors. We want to score each query-key pair
  - Attention score based on query-key similarity
  - New representation = softmax-weighted average of token embeddings

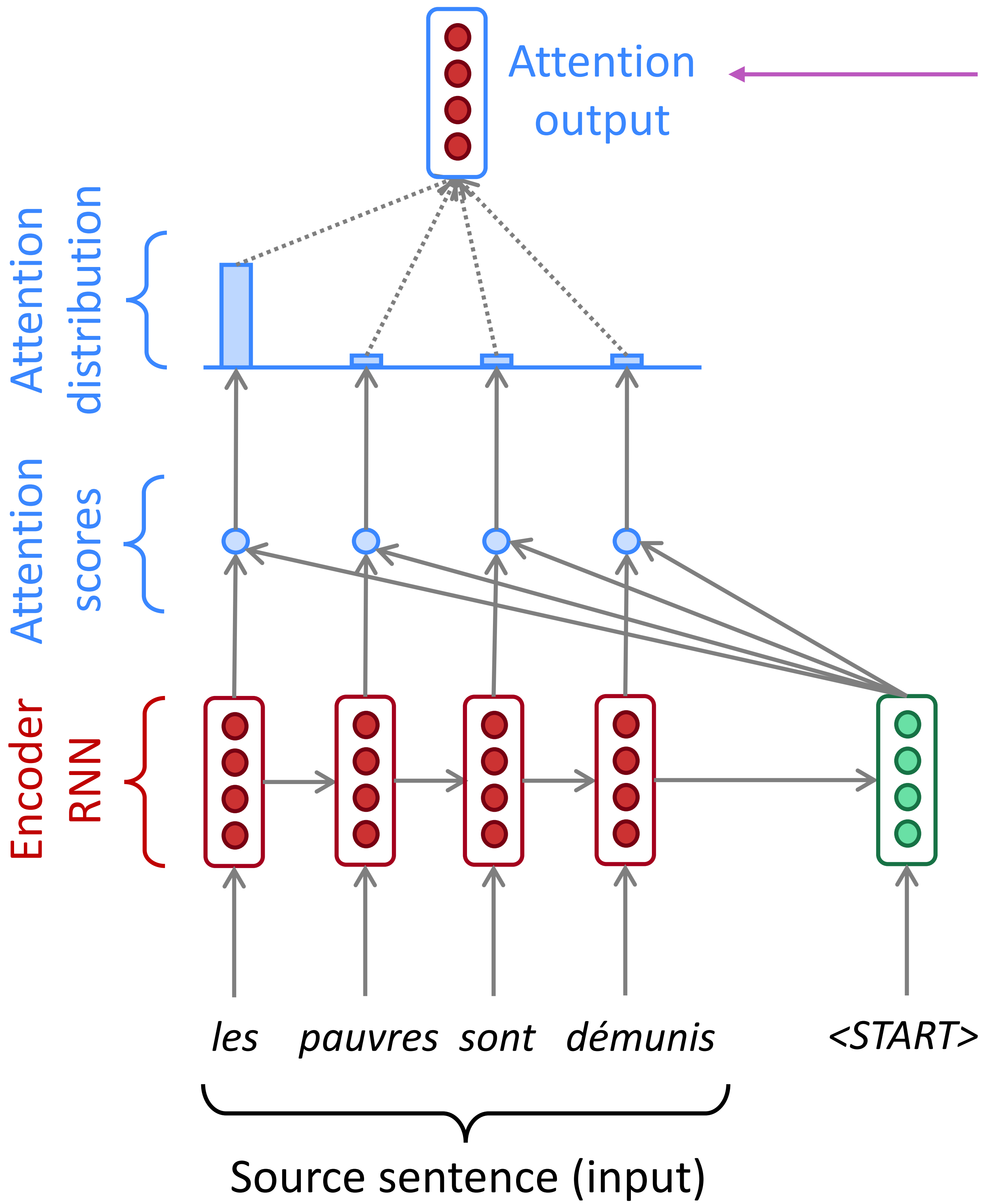
# Sequence-to-sequence with attention



# Sequence-to-sequence with attention



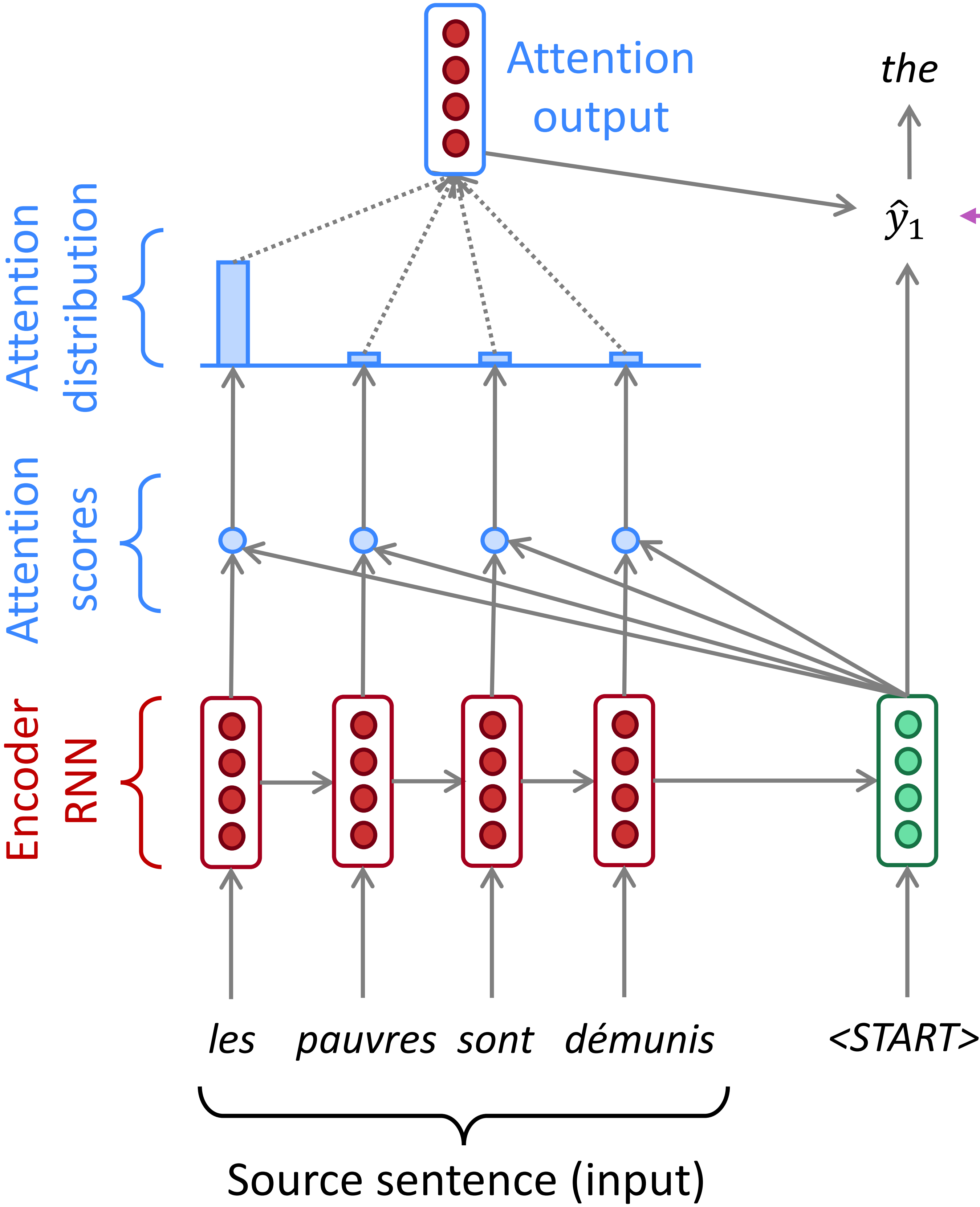
# Sequence-to-sequence with attention




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

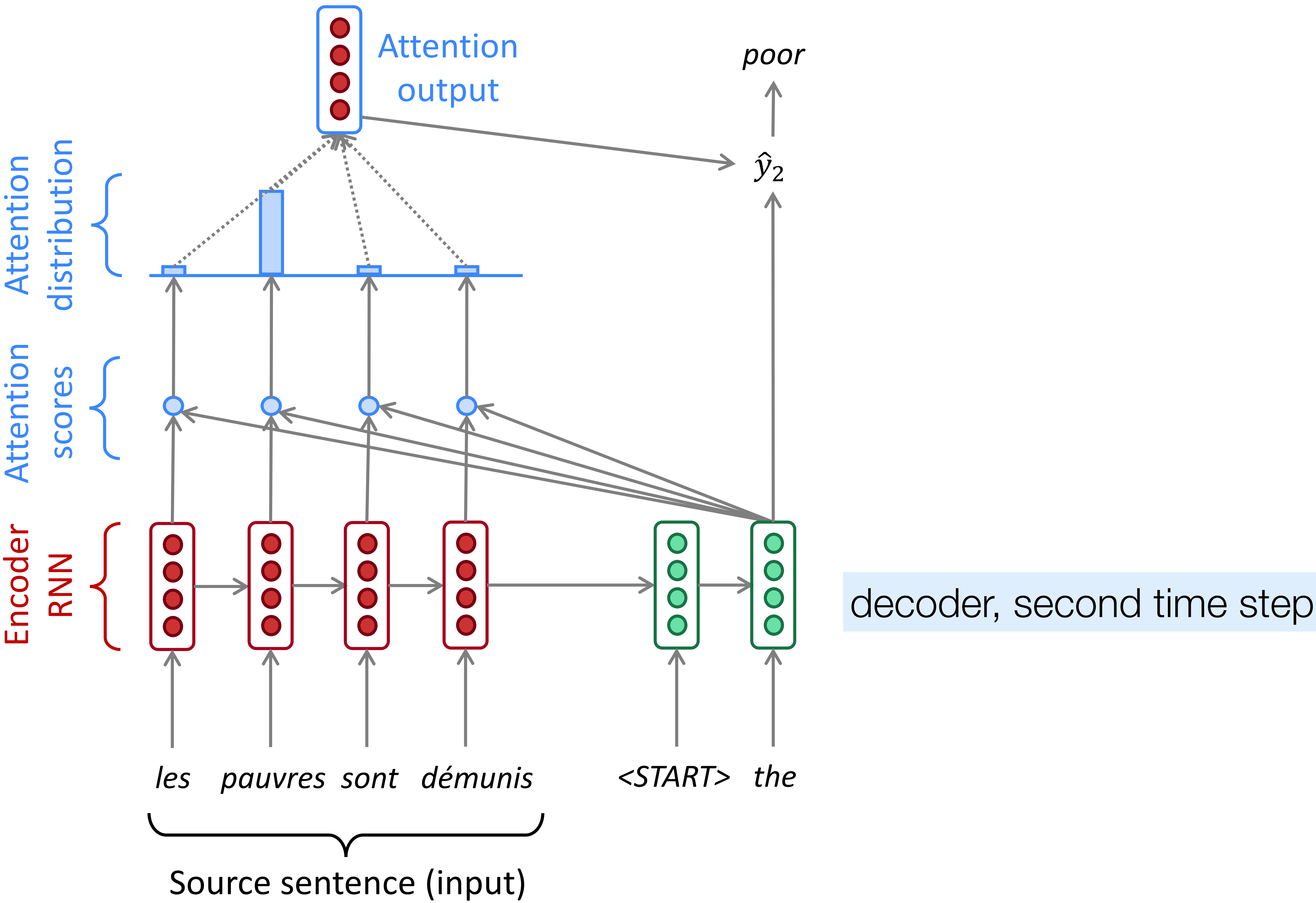
The attention output mostly contains information the hidden states that received high attention.

# Sequence-to-sequence with attention




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

# Sequence-to-sequence with attention



# Many variants of attention

- Dot product: $$a(\mathbf{q}, \mathbf{k}) = \mathbf{q}^T\mathbf{k}$$ Luong et al., 2015

- Scaled dot product: $$a(\mathbf{q}, \mathbf{k}) = \frac{\mathbf{q}^T\mathbf{k}}{\sqrt{|\mathbf{k}|}}$$ Vaswani et al., 2017

  - Attention score based on query-key similarity
  - New representation = softmax-weighted average of token embeddings

# Transformer

- **Self-attention**: token-to-token attention, within a sentence or same text *(Vaswani et al. 2017)*
    - Use to iteratively refine a token's embedding
- e.g. for left-to-right LM:

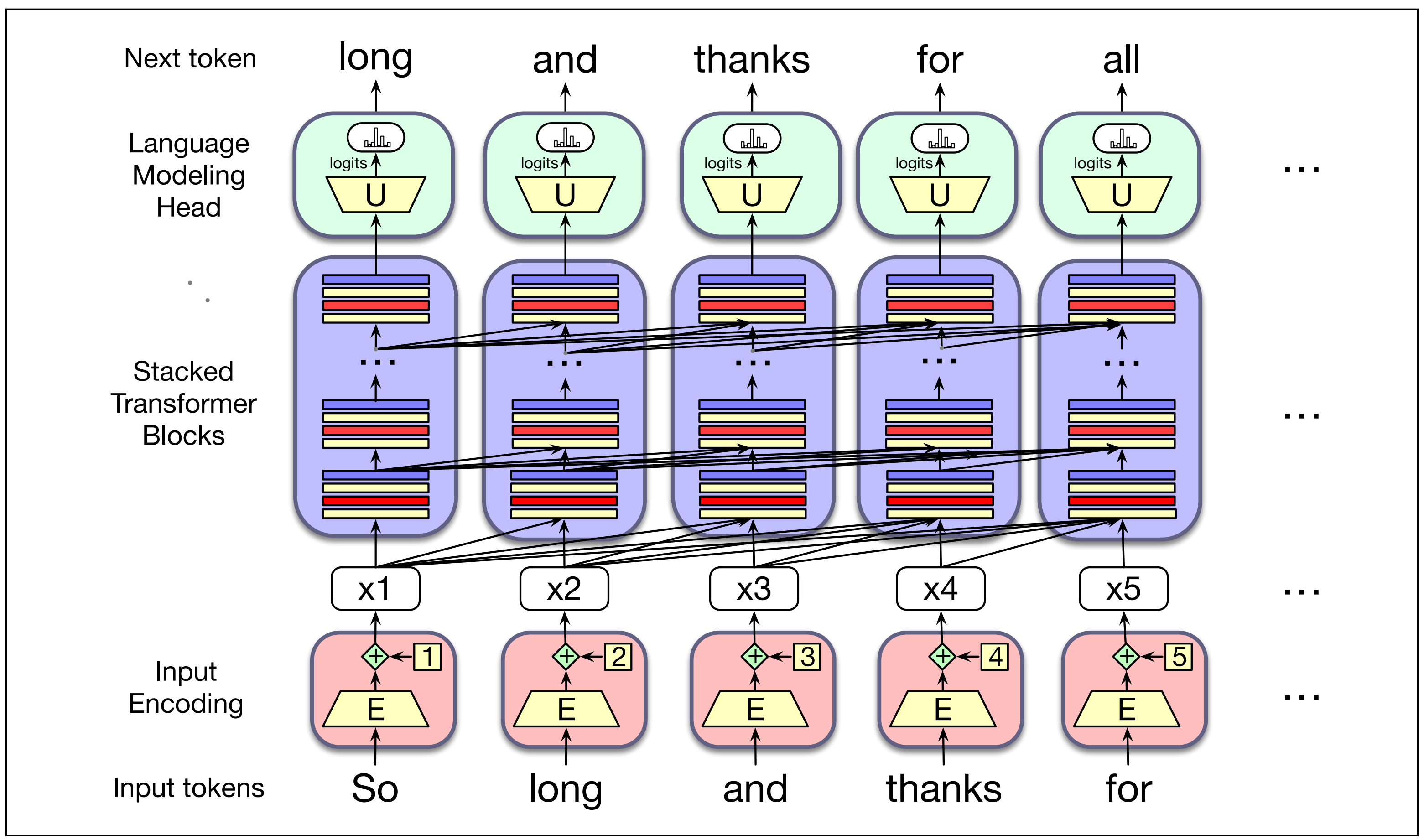



**Figure 9.1** The architecture of a (left-to-right) transformer, showing how each input token get encoded, passed through a set of stacked transformer blocks, and then a language model head that predicts the next token.



*[SLP3 ch. 9]*

# Transformer

- e.g. for masked LM:

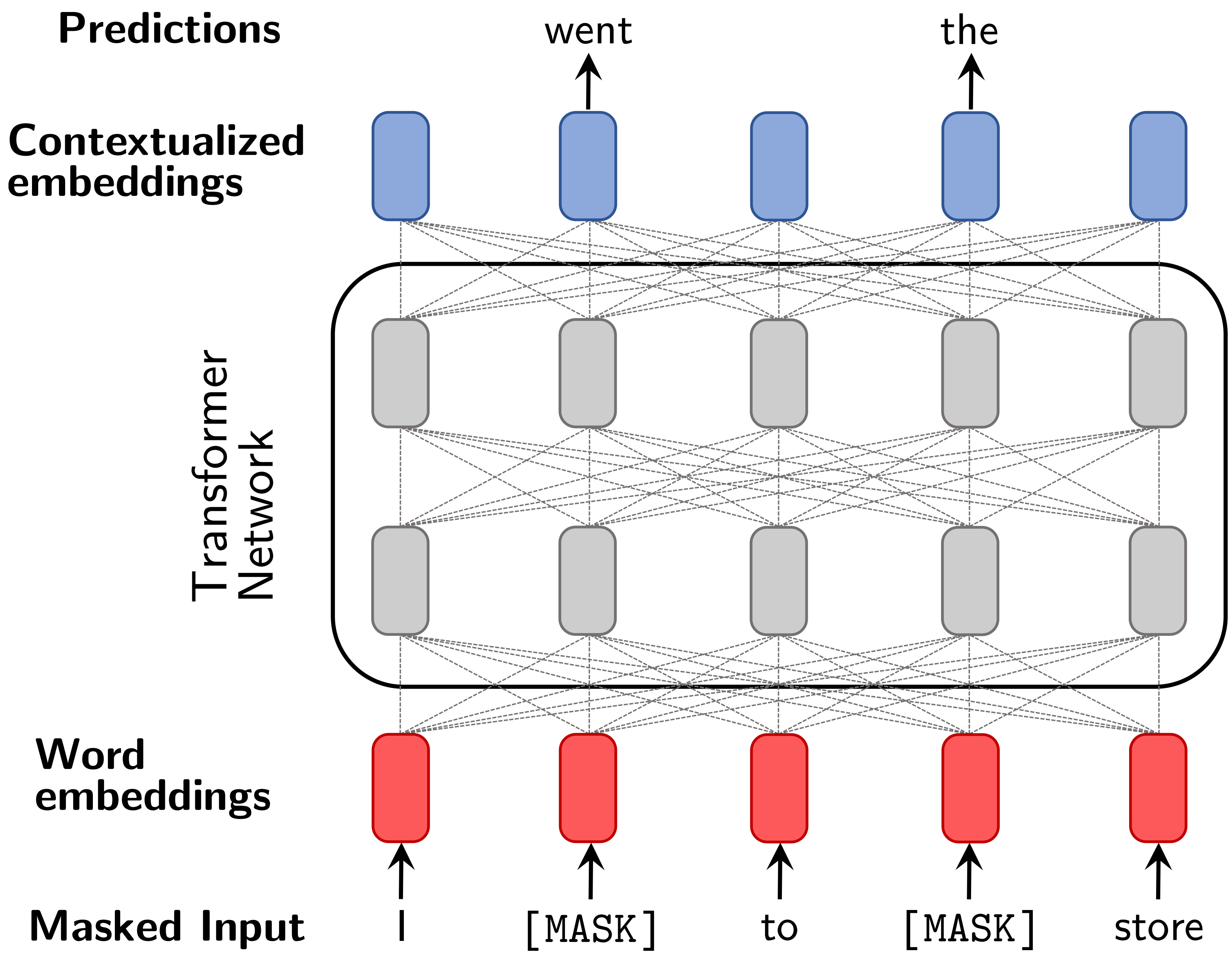



**Fig. 3.** A high-level illustration of BERT. Words in the input sequence are randomly masked out and then all words are embedded as vectors in $\mathbb{R}^d$. A Transformer network applies multiple layers of multiheaded attention to the representations. The final representations are used to predict the identities of the masked-out input words.

*[Manning et al., 2020]*

# Why self-attention?

- *The **keys** to the cabinet {are / is} on the table*

- *The **chicken** didn't cross the **road** because **it** was too {tired / wide}*

- Idea: LM-relevant contextual information may be pretty far away!
  - *The **keys** to the cabinet, which I love so much and are important and I think about all the time, [....] {are / is} on the table*

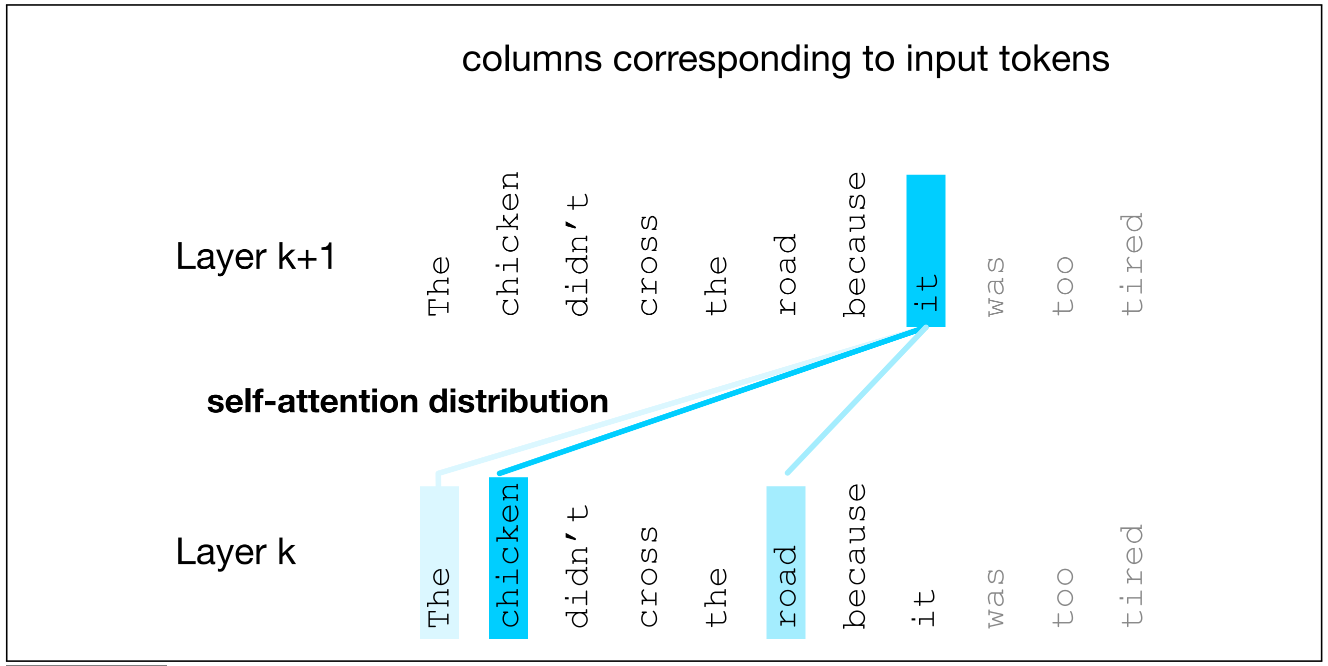
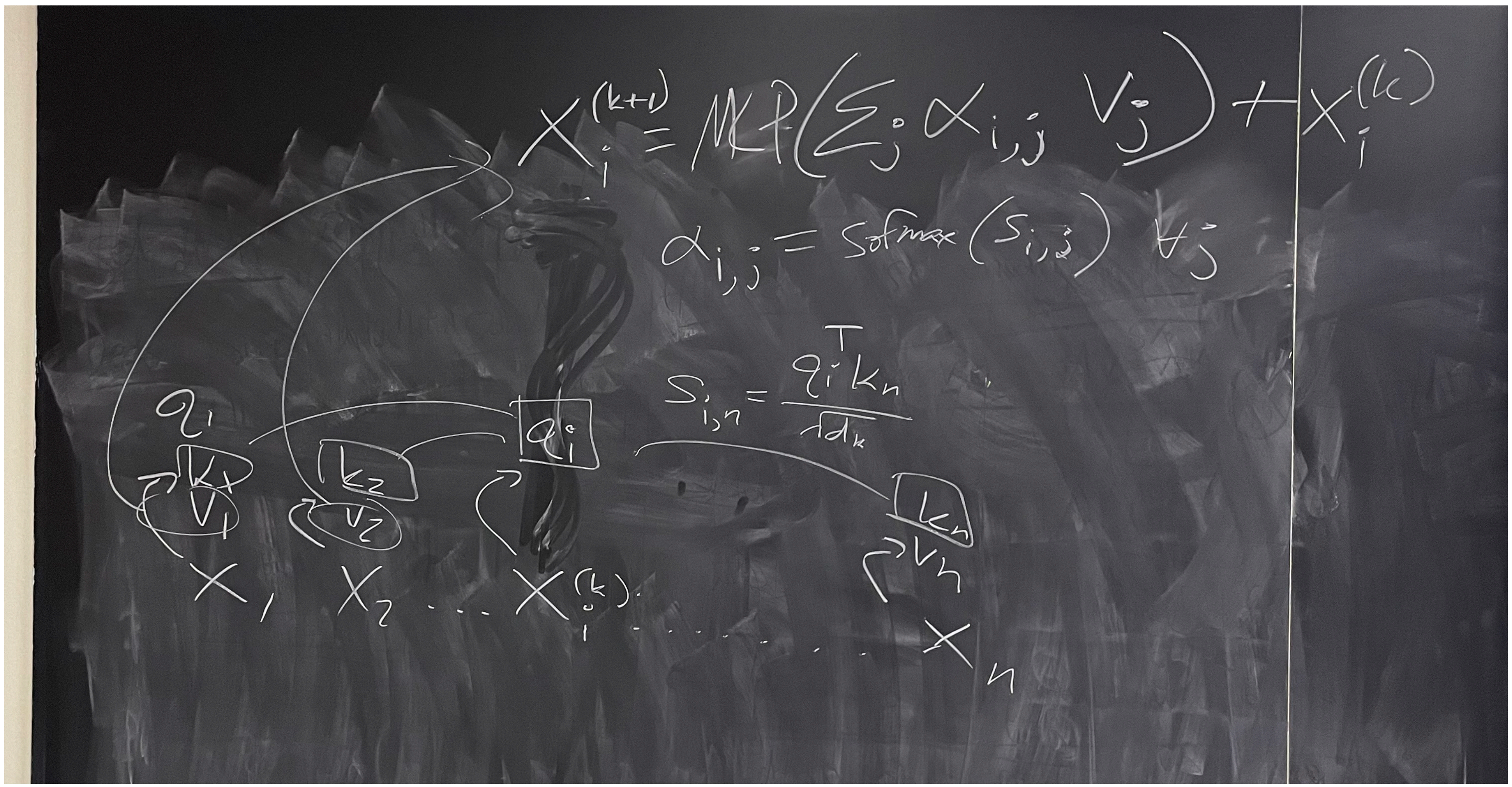



**Figure 9.2** The self-attention weight distribution $\alpha$ that is part of the computation of the representation for the word *it* at layer $k+1$. In computing the representation for *it*, we attend differently to the various words at layer $l$, with darker shades indicating higher self-attention values. Note that the transformer is attending highly to the columns corresponding to the tokens *chicken* and *road* , a sensible result, since at the point where *it* occurs, it could plausibly corefers with the chicken or the road, and hence we'd like the representation for *it* to draw on the representation for these earlier words. Figure adapted from Uszkoreit (2017).

$$
\begin{aligned}
\mathbf{q}_i = \mathbf{x}_i\mathbf{W}^{\mathbf{Q}}; \quad \mathbf{k}_j &= \mathbf{x}_j\mathbf{W}^{\mathbf{K}}; \quad \mathbf{v}_j = \mathbf{x}_j\mathbf{W}^{\mathbf{V}} \\
\text{score}(\mathbf{x}_i, \mathbf{x}_j) &= \frac{\mathbf{q}_i \cdot \mathbf{k}_j}{\sqrt{d_k}} \\
\alpha_{ij} &= \text{softmax}(\text{score}(\mathbf{x}_i, \mathbf{x}_j)) \;\; \forall j \leq i \\
\mathbf{a}_i &= \sum_{j \leq i} \alpha_{ij}\mathbf{v}_j
\end{aligned}
$$




- And in addition to the self-attention mechanism,
  - This was one head = ($W^Q$, $W^K$, $W^V$) tuple. There are multiple heads which can specialize for different attention behaviors
  - Feedforward + residual layer

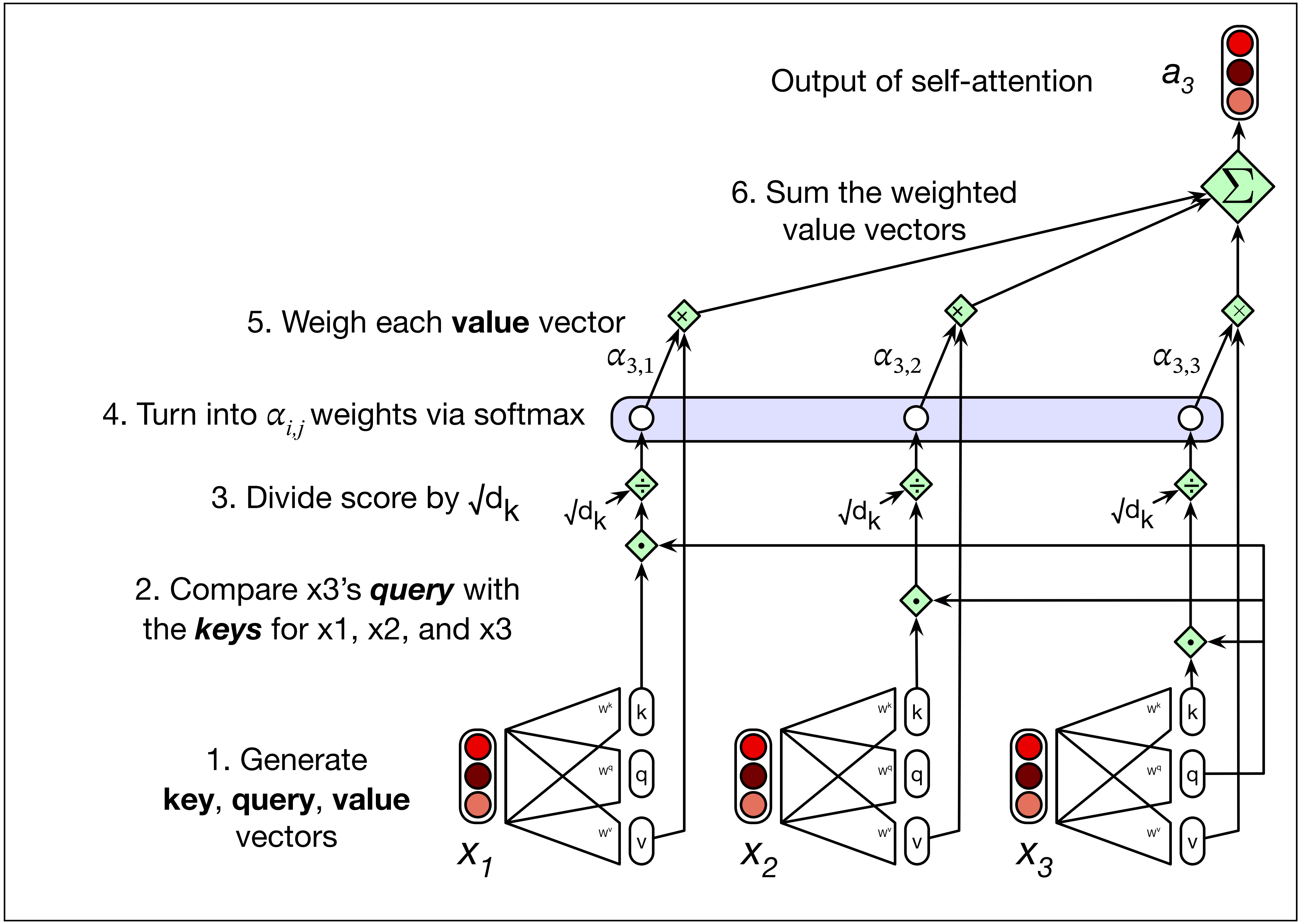



**Figure 9.4** Calculating the value of $\mathbf{a}_3$, the third element of a sequence using causal (left-to-right) self-attention.

# BERT

- "**<u>B</u>idirectional** <u>e</u>ncoder <u>r</u>epresentations from **<u>T</u>ransformers**"
  - Transformer: a **"*self-attention*"** neural net architecture that infers **context-aware** token embeddings
  - Bidirectional: Pretraining with a ***masked LM***, predicting missing word(s) from rest of words in sentence
- Usage
  - 1. **Pretrain** the network via masked language model on a large corpus
  - 2. **Fine-tune**: further learn better parameters for a specific task, e.g. classification
- BERT (+ variants) are really useful, and work because they learn both word embeddings and linguistic structure from pretraining
  - many implementations: https://huggingface.co/docs/transformers/index

# What does BERT learn?

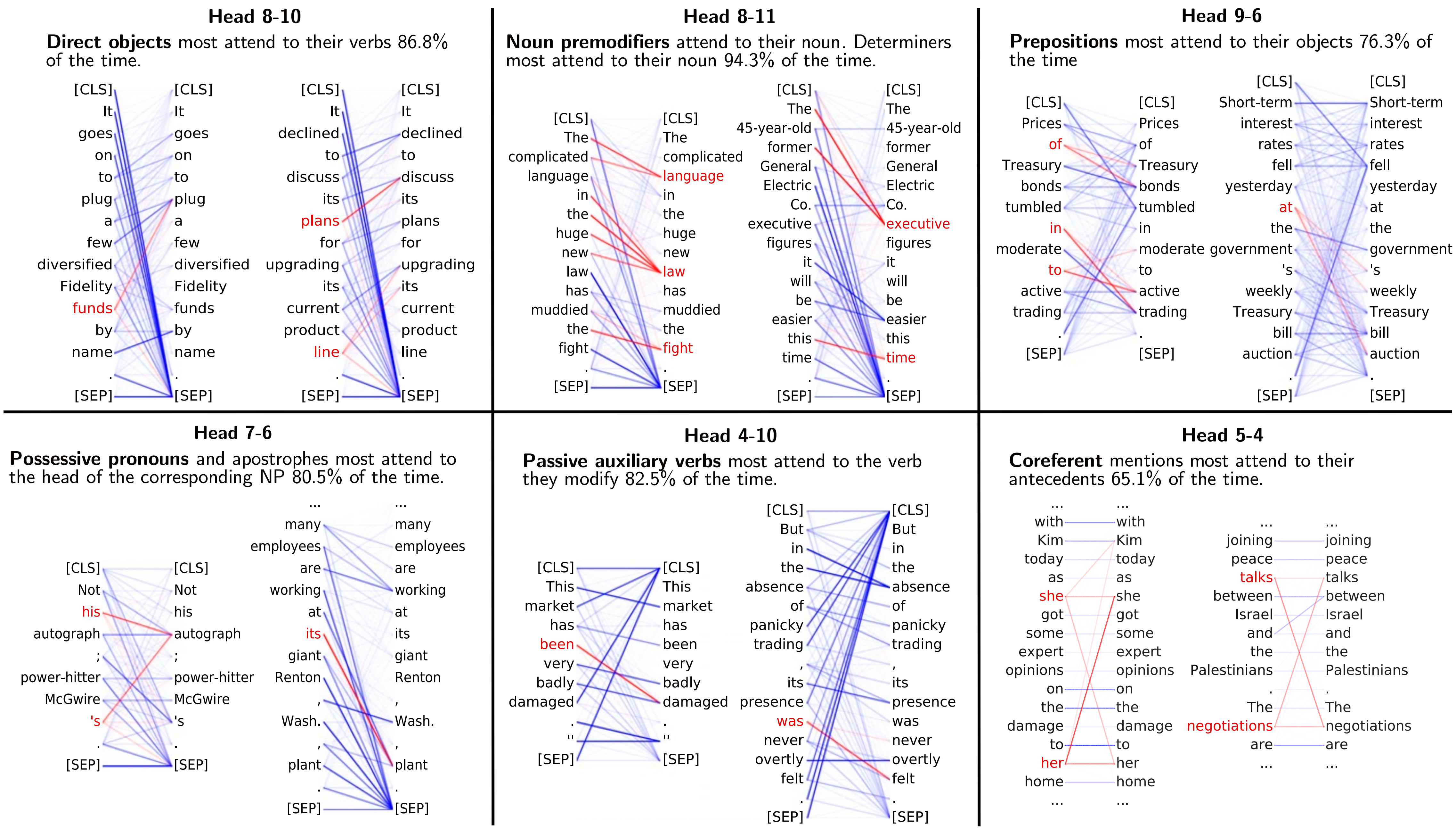



**Fig. 6.** Some BERT attention heads that appear sensitive to linguistic phenomena, despite not being explicitly trained on linguistic annotations. In the example attention maps, the darkness of a line indicates the size of the attention weight. All attention to/from red words is colored red; these words are chosen to highlight certain of the attention heads' behaviors. [CLS] (classification) and [SEP] (separator) are special tokens BERT adds to the input during preprocessing. Attention heads are numbered by their layer and index in BERT. Reprinted with permission from ref. 59, which is licensed under CC BY 4.0.



[Manning et al., 2020]

# What does BERT learn?

- BERT is typically the highest accuracy way to predict POS, syntax, NER, etc.
- If you use multiple layers to predict linguistic structures, what layers encode the information?

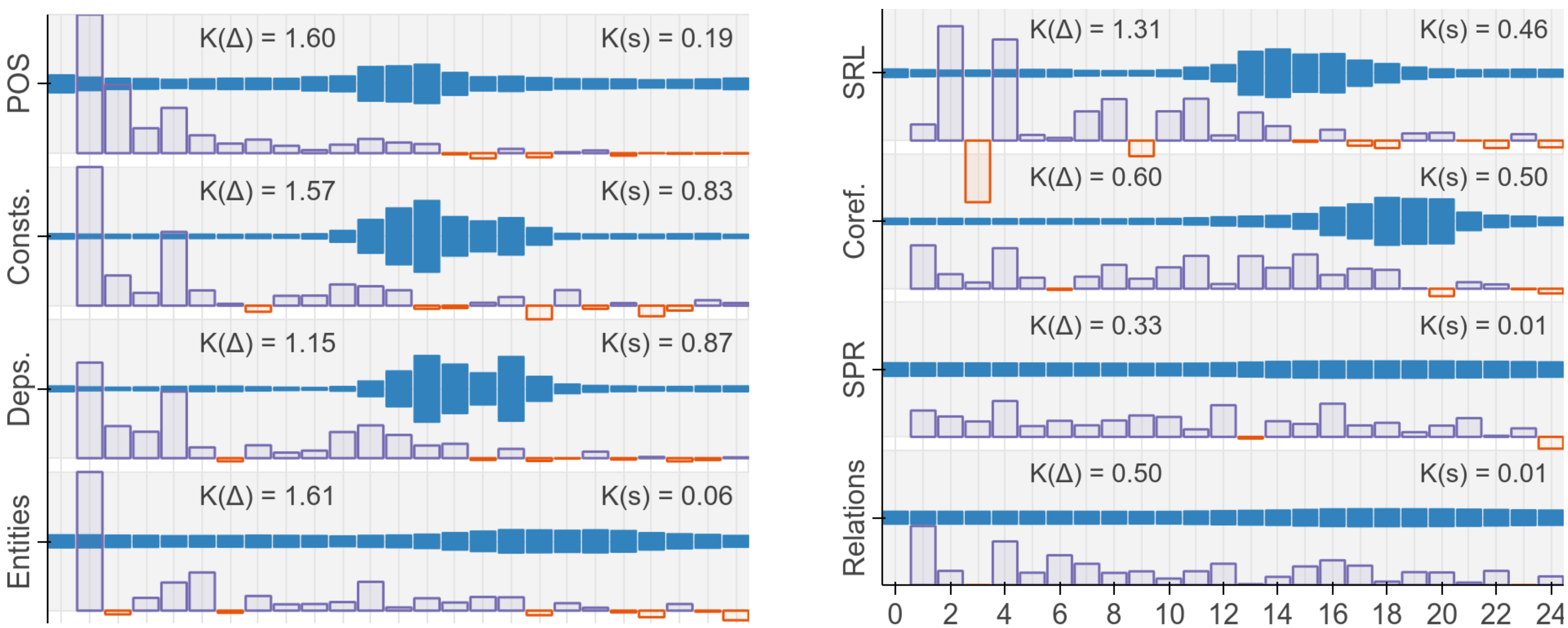



Figure 2: Layer-wise metrics on BERT-large. Solid (blue) are mixing weights $s_\tau^{(\ell)}$ (§3.1); outlined (purple) are differential scores $\Delta_\tau^{(\ell)}$ (§3.2), normalized for each task. Horizontal axis is encoder layer.



[Tenney et al., 2019]

# What does BERT learn?

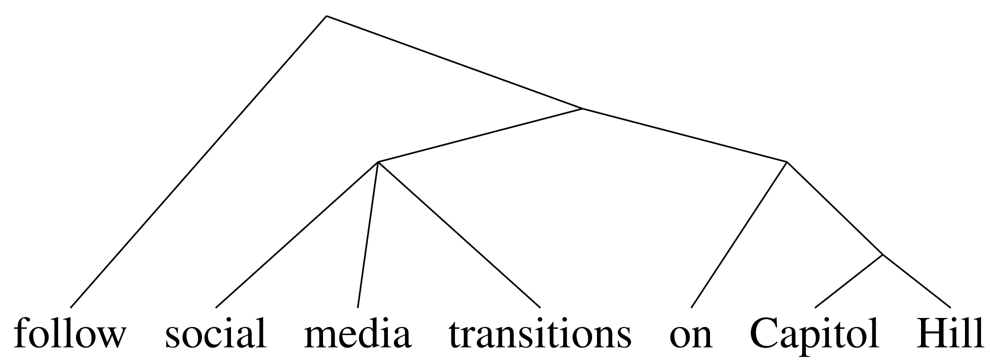



Figure 1: Parameter-free probe for syntactic knowledge: words sharing syntactic subtrees have larger impact on each other in the MLM prediction (Wu et al., 2020).



[Rogers et al., 2020]

# Using BERT

- You get
  - Per-token embeddings
  - Multiple layers of embeddings (!)
  - Embedding for per-sentence “[CLS]” symbol
- Use as input for tasks
  - Fine-tuning: add a prediction head, then backprop through the actual BERT model itself
    - The transformer network (with fine-tuned parameters) *is* your final classifier/tagger
  - Less common: directly use embeddings

# Byte pair encoding (BPE)

- BERT is a neural LM designed to be used on arbitrary text later. But what should the vocabulary be?
- 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
  - Poor tokenization can cause many problems in practice

| 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

# Application

- Fine-tuned BERT is one of the most accurate ways to train a text classifier or tagger if you have a moderate (>100) amount of labeled data
- "BERT" sometimes means the original release, but sometimes means the general class of models (!)
- Many pretrained BERT-like, MLM-trained models are available
    - RoBERTa is a good, general-purpose one
    - mBERT and XLM-R: multilingual models
    - Many specific languages or language families (AfriBERTa, LatinBERT, ...)
    - Many domains (LegalBERT, BERTweet, SciBERT, ...)
- Check out HuggingFaces' examples
    - https://huggingface.co/transformers/examples.html

# SentenceBERT

- Also there are many released BERT-likes tuned for specific tasks - sentiment, toxicity, etc.
- **SentenceBERT** is worth mentioning: designed to encode sentences to embedding vectors
  - Cosine similarity often works very well!
  - The model is trained to give high cosine similarity to human-annotated pairs of similar sentences
- https://sbert.net/

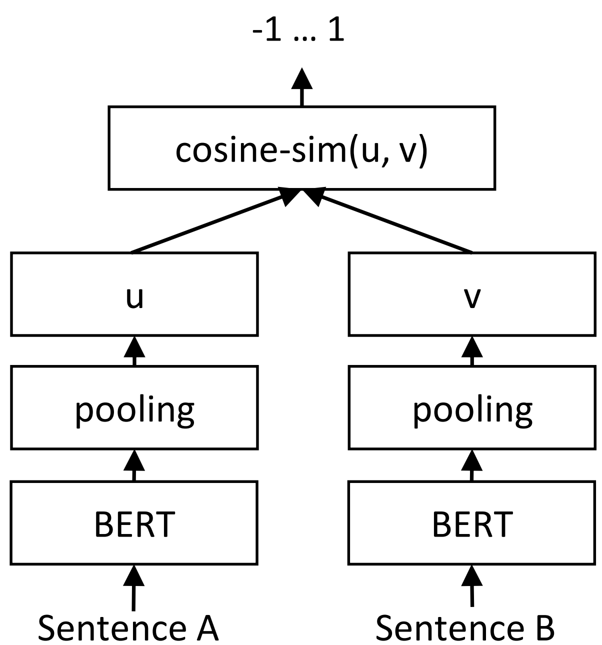



Figure 2: SBERT architecture at inference, for example, to compute similarity scores. This architecture is also used with the regression objective function.

[Reimers and Gurevich, 2019]

# Challenges

- Some issues
  - Bidirectional models can't generate
  - BERT fails to model plenty of tricky phenomena
  - How to collect a large pretraining corpus?
  - Why does all this work?

- BERT fine-tuning is often the best classifier you can make.
  - Note widely used variants: RoBERTa, DeBERTa