CS769 Advanced NLP

Attention and Transformer

Junjie Hu

UNIVERSITY OF WISCONSIN–MADISON

Slides adapted from Graham, Sergey

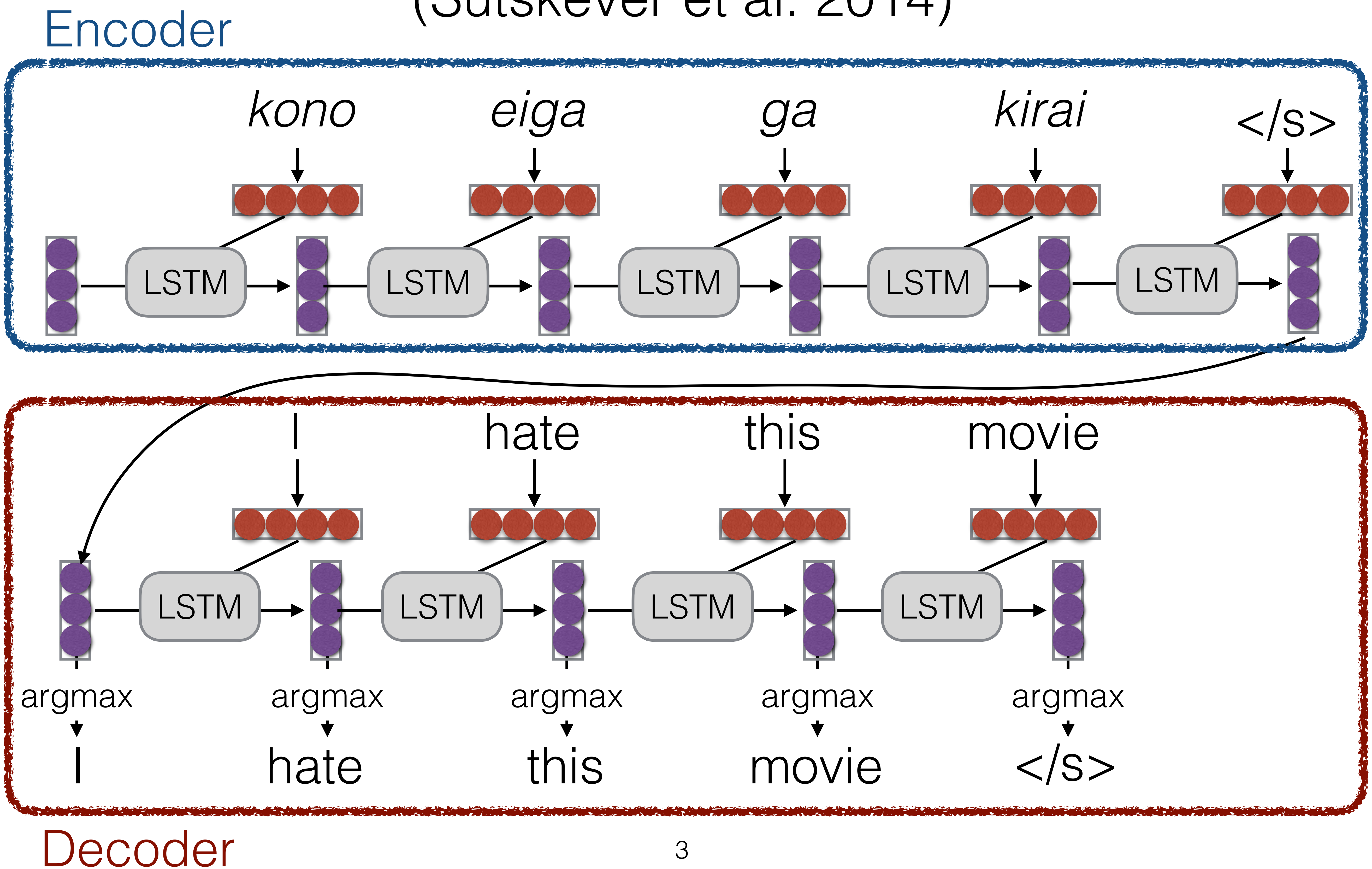
https://junjiehu.github.io/cs769-spring23/

Goals for Today

- Brief Introduction to Attention
- Transformer (Five Key Components)
- Advanced Training And Applications of Attention

Encoder-decoder Models

(Sutskever et al. 2014)

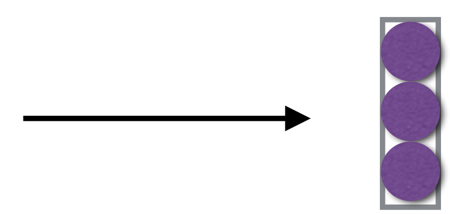


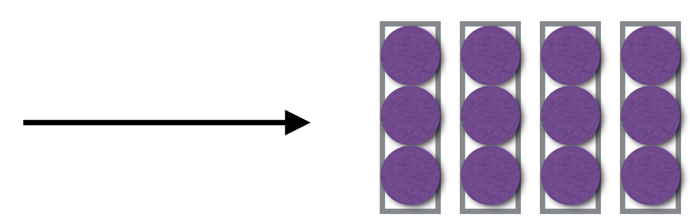
Sentence Representations

<u>Problem!</u>

> It's not ideal to compress the meaning of a sentence with variable length into a single vector.

- But what if we could use multiple vectors, based on the length of the sentence.

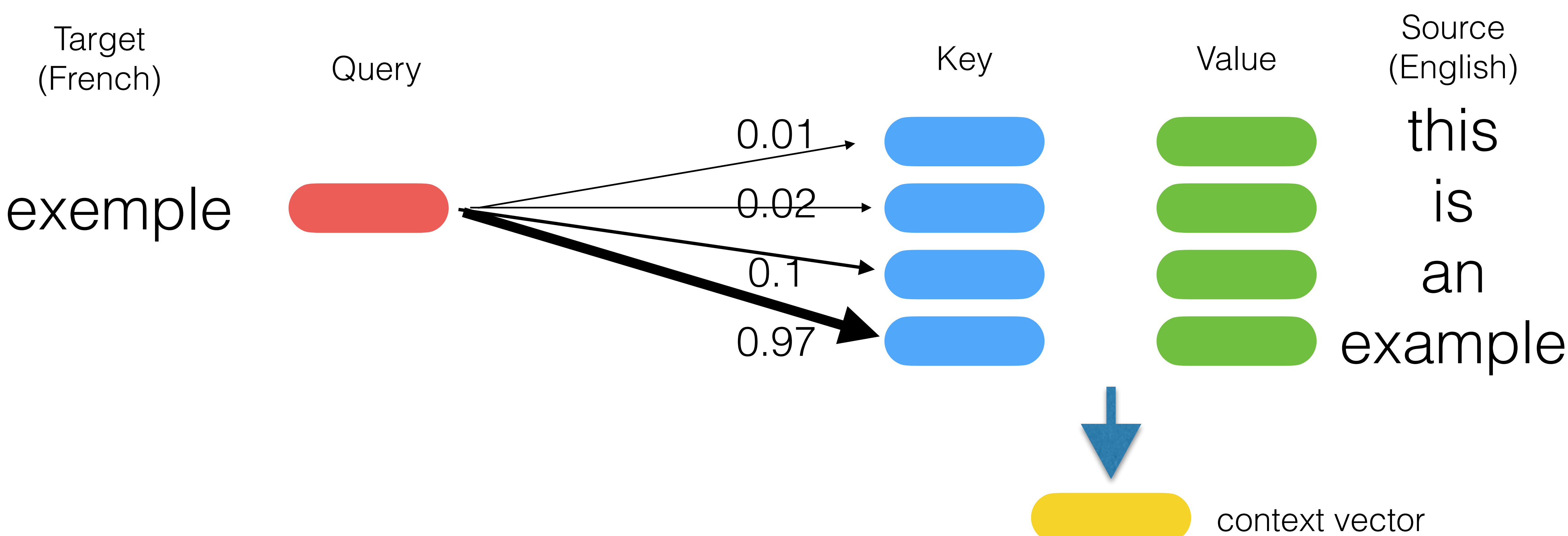
this is an example 

this is an example 

Attention

Basic Idea of Attention

- Embed the **source elements** (e.g., English words) into a dictionary of (**key, value**) vectors
- When a **query** of a target element (e.g., a French word), pick relevant source elements by comparing **query** and **keys**
- Summarize the relevant **values** into **a context vector**



Basic Idea

(Bahdanau et al. 2015)

- Attention is first used in machine translation
- Encode each word in the sentence into a vector
- When decoding, perform a linear combination of these vectors, weighted by “attention weights”
- Use this combination in picking the next word
- In a sequence-to-sequence model, we sometimes call the attention from *the **target** hidden vector (**query**)* to *all the **source** vectors (**keys**)* as “**target-to-source cross attention**”.

Attention: "pick" at the input

Intuition: **Send** the most relevant h_t by $\arg\max_t e_{t,l}$ **to step** l, however argmax operation is **not differentiable!**

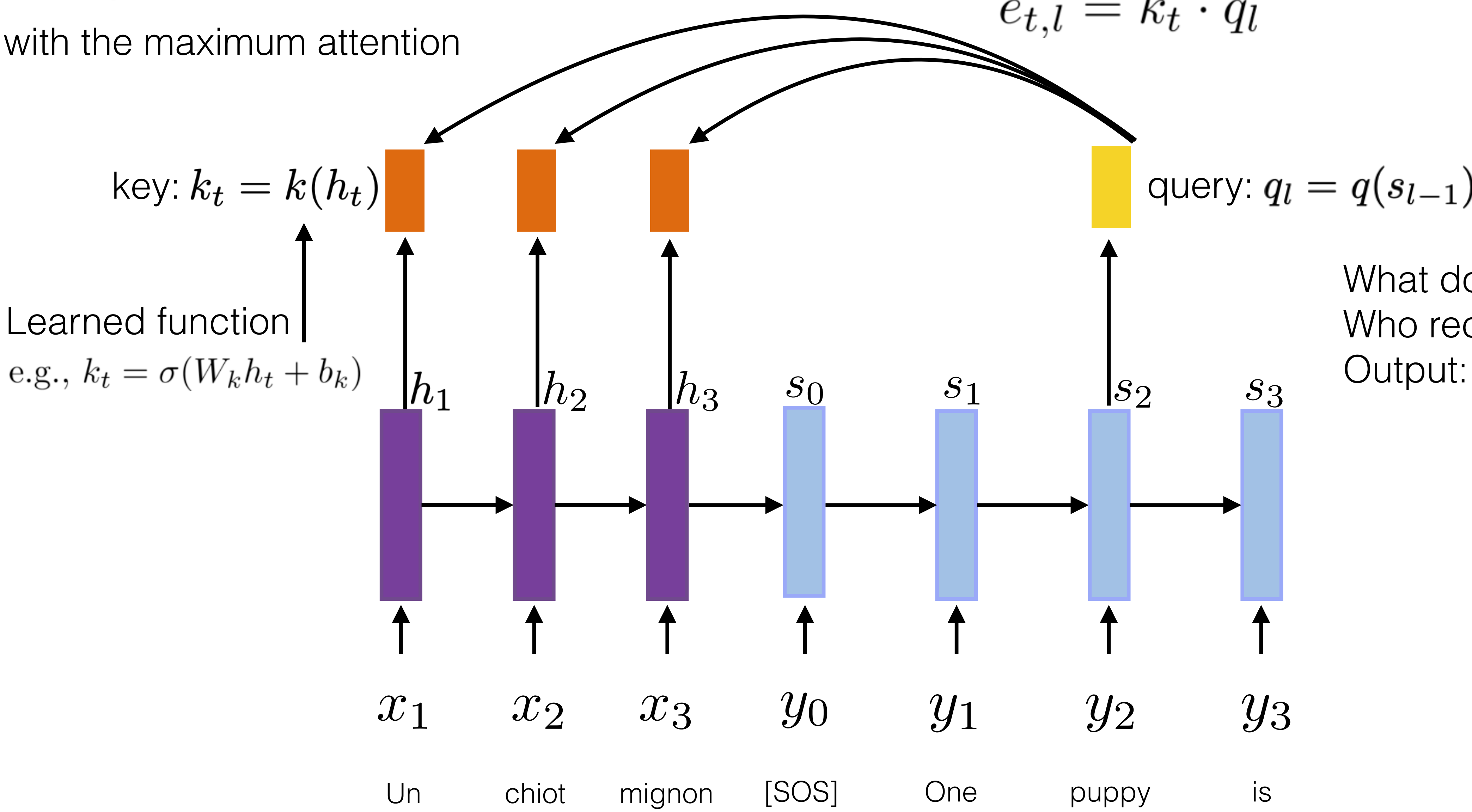
$$\alpha_{\cdot,l} = \text{softmax}(e_{\cdot,l})$$

$$\alpha_{t,l} = \frac{\exp(e_{t,l})}{\sum_{t'} \exp(e_{t',l})}$$

Let $a_l = \sum_t \alpha_{t,l} h_t$ → approximate h_t with the maximum attention

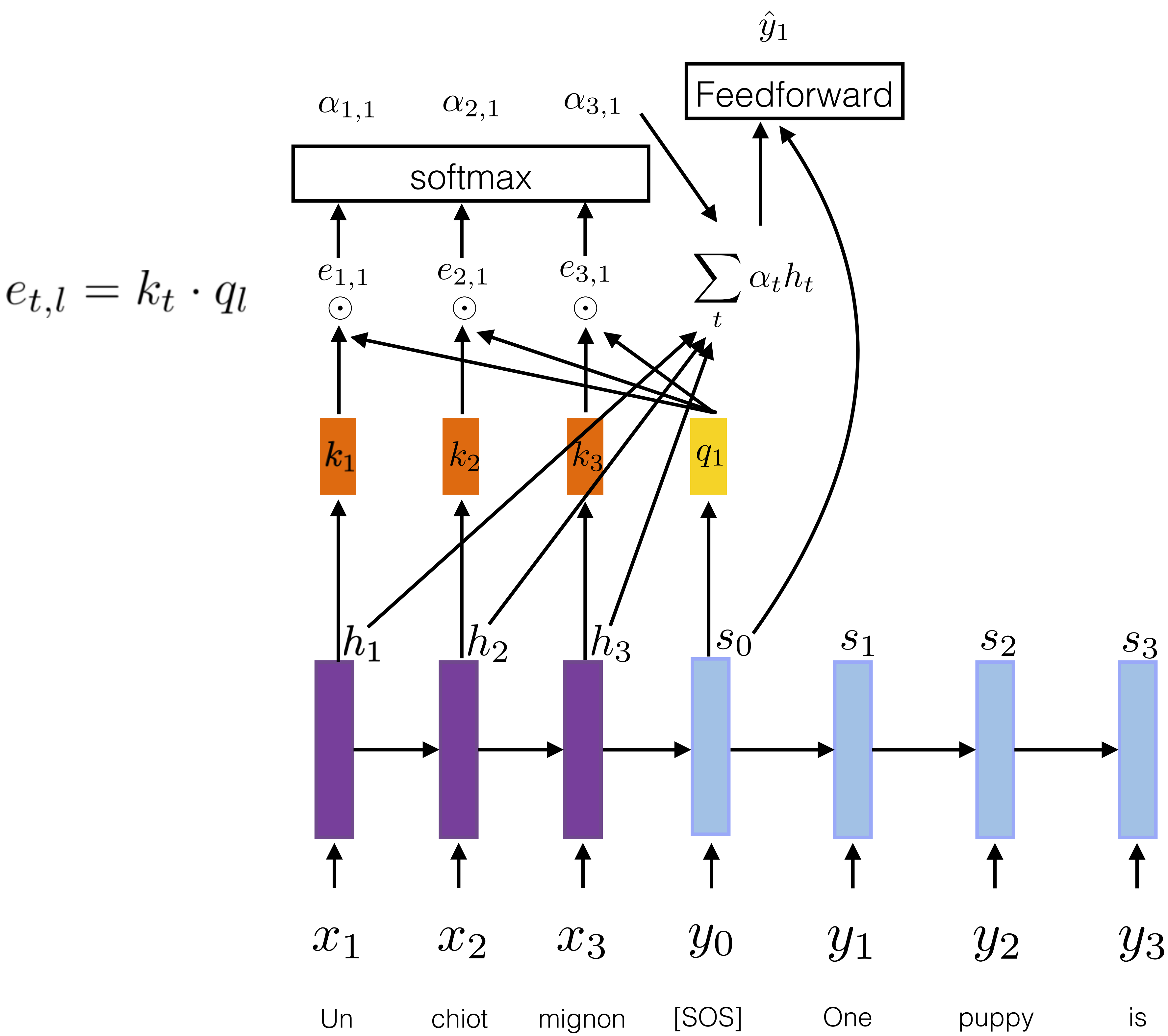
Attention score for (encoder) step t to (decoder) step l

$$e_{t,l} = k_t \cdot q_l$$



What does "send" mean?
Who receives it?
Output: $\hat{y}_l = f(s_{l-1}, a_l)$

Attention (Example)



A Graphical Example

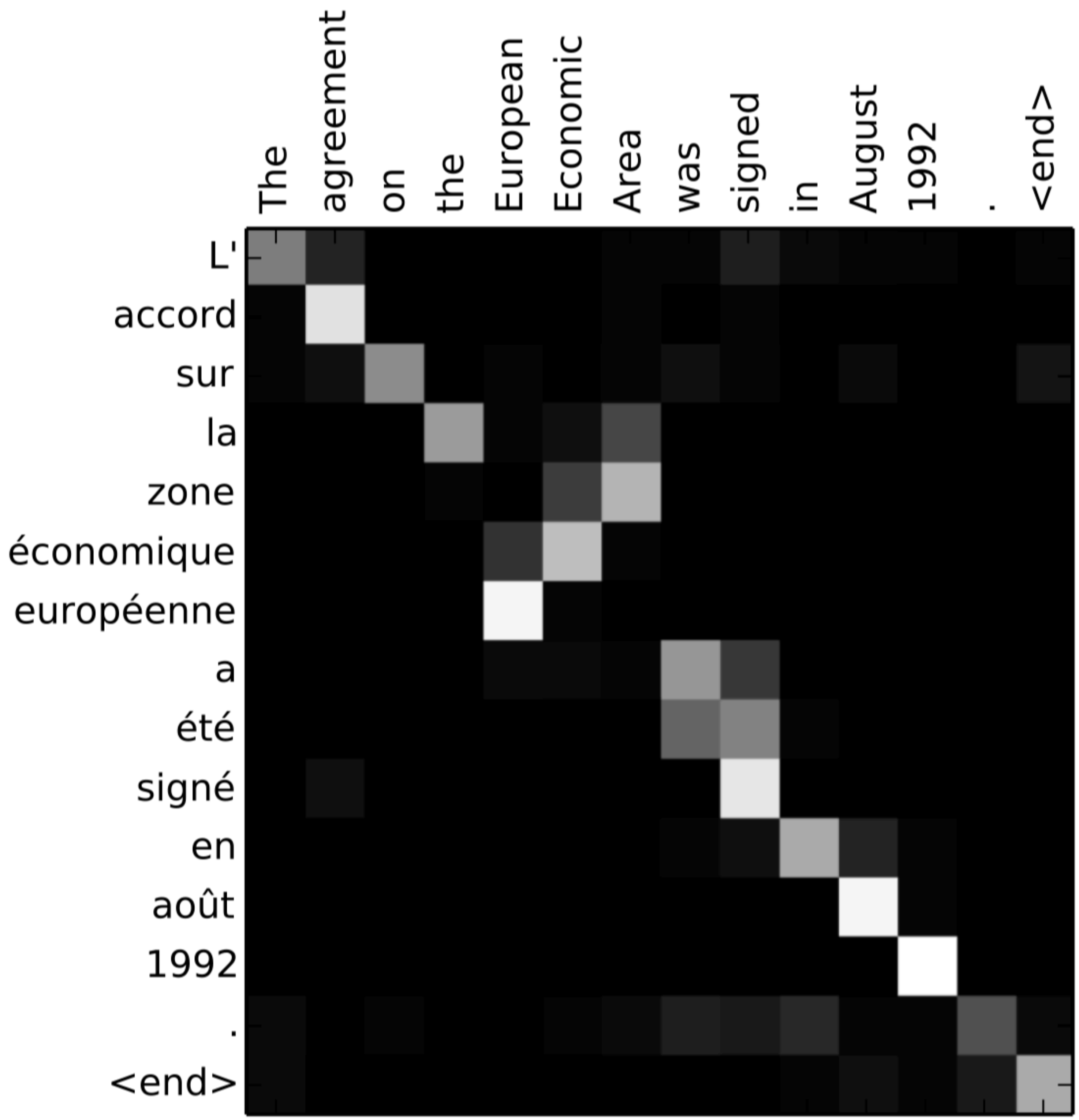


Image from Bahdanau et al. (2015)

Attention Score Functions (1)

- $\boldsymbol{q}$ is the query and $\boldsymbol{k}$ is the key
- **Multi-layer Perceptron** (Bahdanau et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_2^\intercal \tanh(W_1[\boldsymbol{q}; \boldsymbol{k}])$$

  - Flexible, often very good with large data
- **Bilinear** (Luong et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal W \boldsymbol{k}$$

Attention Score Functions (2)

- **Dot Product** (Luong et al. 2015)

$$a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal \boldsymbol{k}$$

  - No parameters! But requires sizes to be the same.

- **Scaled Dot Product** (Vaswani et al. 2017)

  - *Problem:* scale of dot product increases as dimensions get larger

  - *Fix:* scale by size of the vector

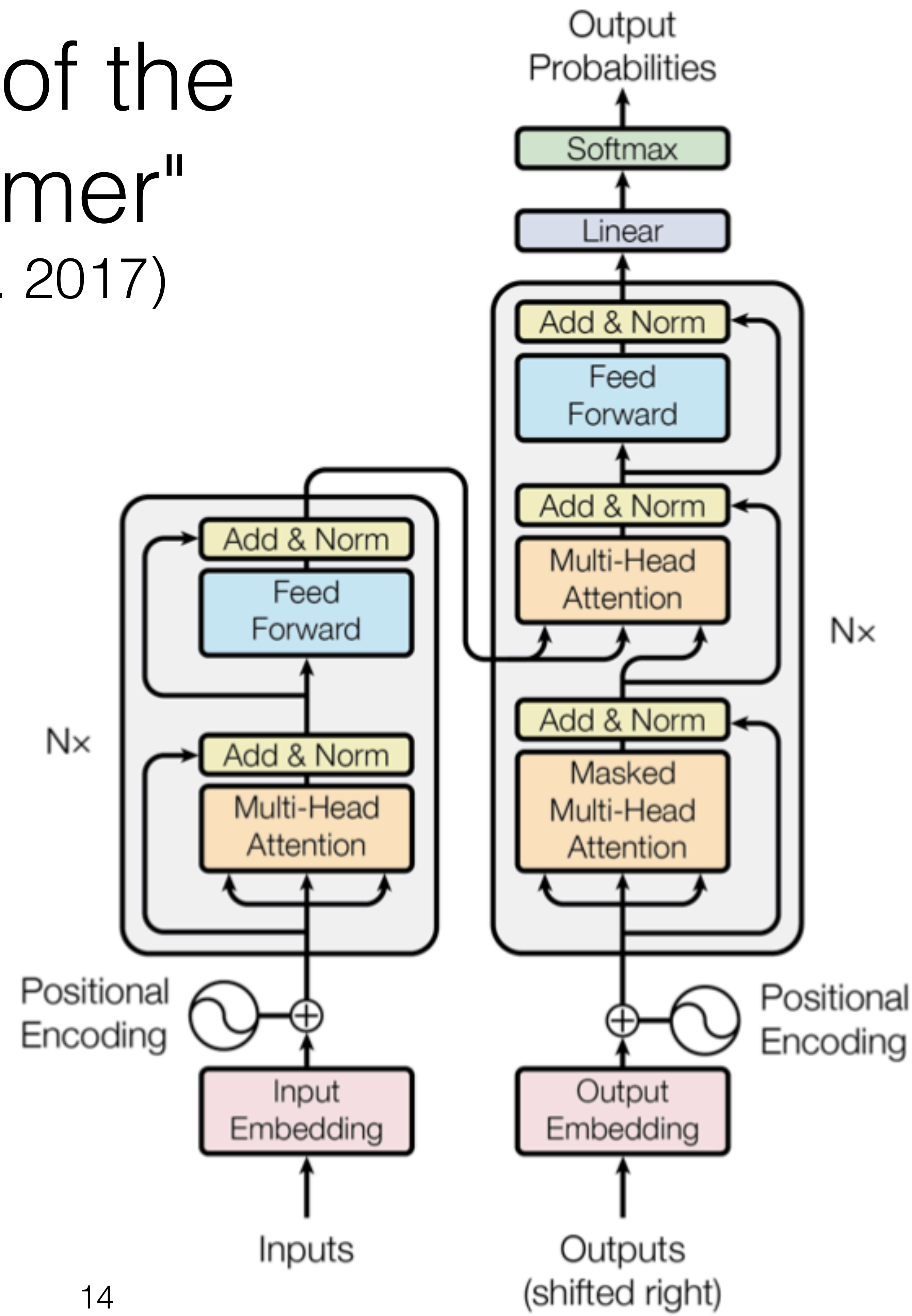
$$a(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\intercal \boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}}$$

Transformer:
“Attention is All You Need”

(Vaswani et al. 2017)

Summary of the "Transformer" (Vaswani et al. 2017)

- A sequence-to-sequence model based entirely on attention
- Strong results on translation, a wide variety of other tasks
- Fast: only matrix multiplications



Transformers

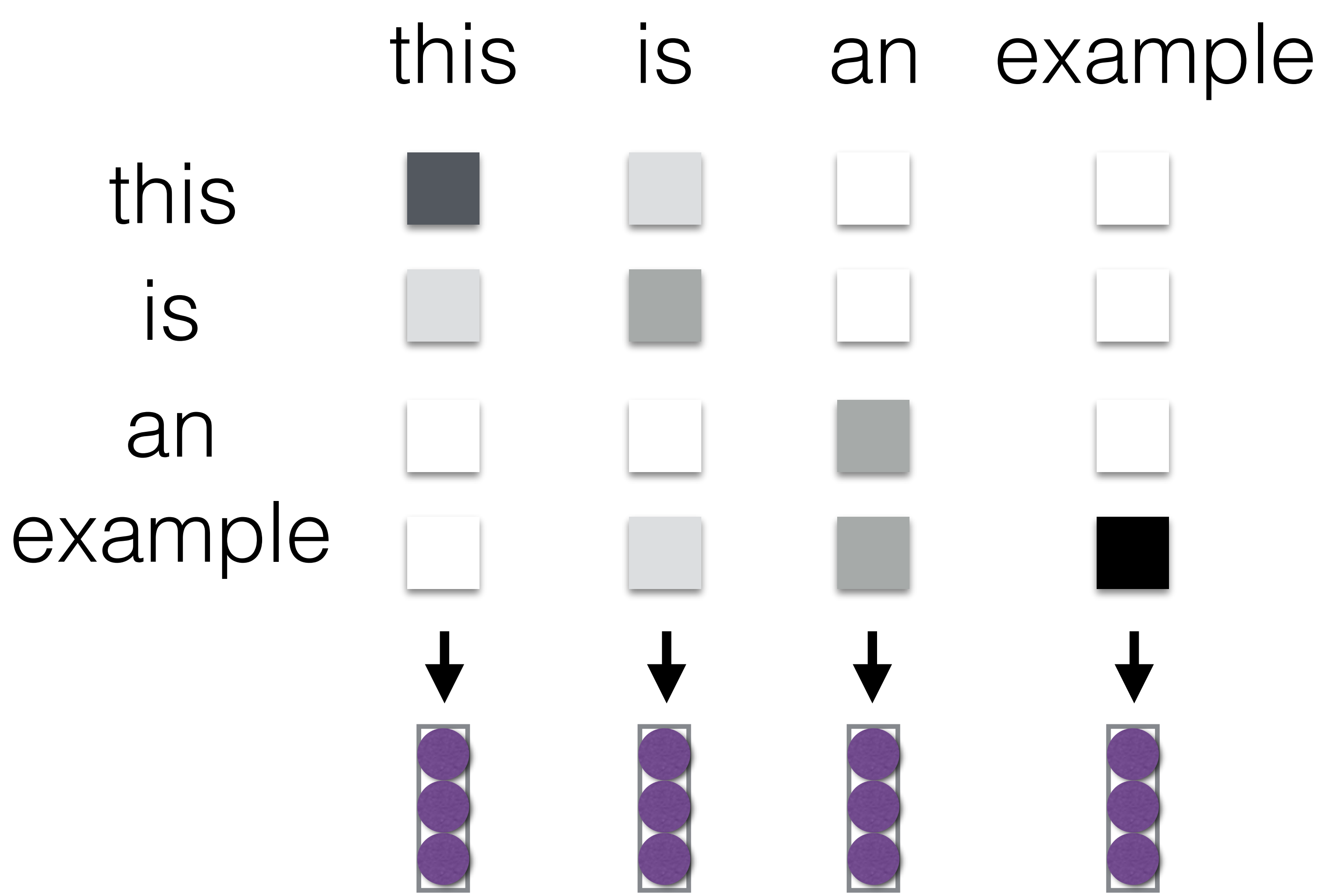
- A few key components to make Transformer work.

    1. **Self-attention** — allows parallel computing of all tokens
    2. **Multi-headed attention** — allows querying multiple positions at each layer
    3. **Position encoding** — adds position information to each token
    4. **Adding nonlinearities** — combines features from a self-attention layer
    5. **Masked decoding** — prevents attention lookups in the future tokens

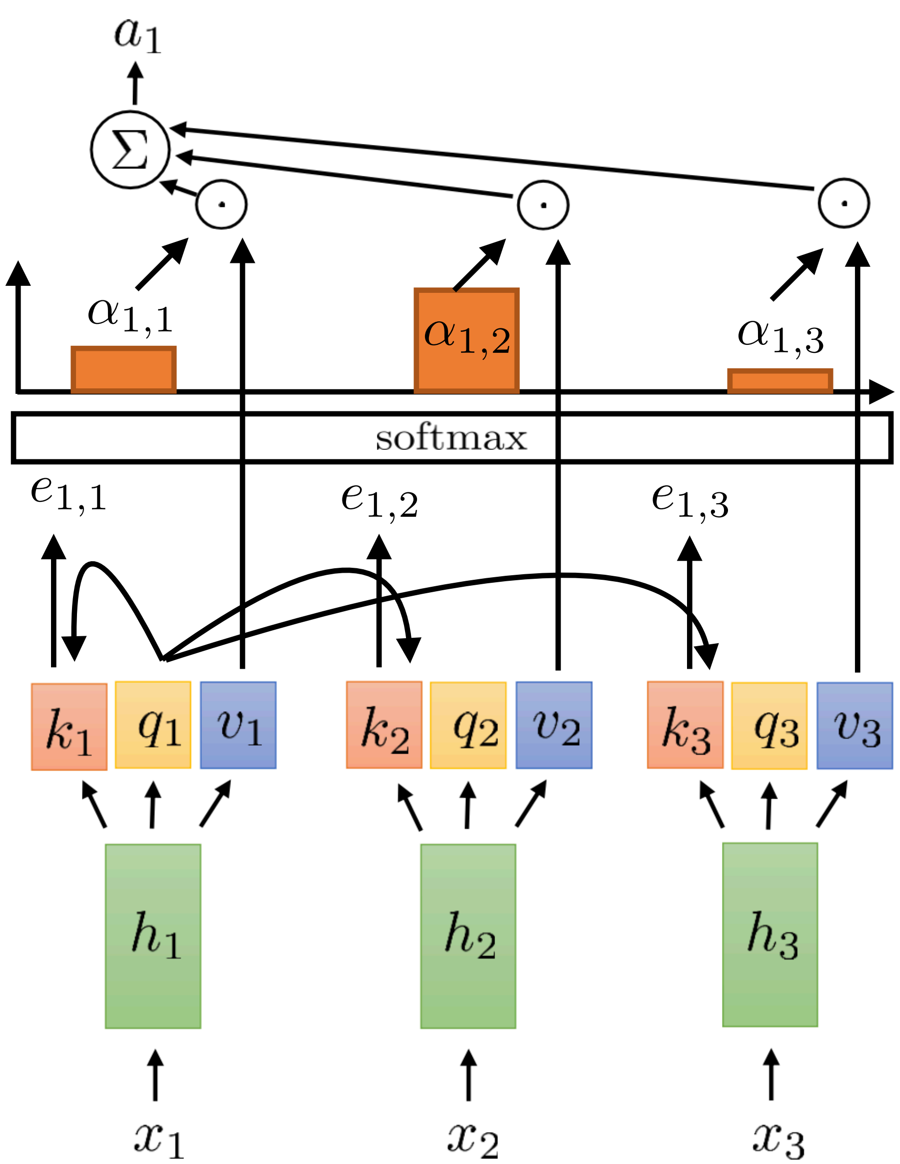
Self Attention
(Cheng et al. 2016, Vaswani et al. 2017)

- **Intuition:** Each element in the sentence attends to all elements including itself → context sensitive encodings!

- Each element will be used as **key, value and query** in self-attention



Self-Attention



Example to compute the attention context for the l-th token

$$a_l = \sum_t \alpha_{l,t} v_t$$
$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$
$$e_{l,t} = q_l \cdot k_t$$

$$v_t = W_v h_t$$
$$k_t = W_k h_t \qquad W_v, W_k, Wq \in \mathbb{R}^{d \times d}$$
$$q_t = W_q h_t, \qquad v_t, k_t, q_t, h_t \in \mathbb{R}^d,$$

this is *not* a recurrent model!

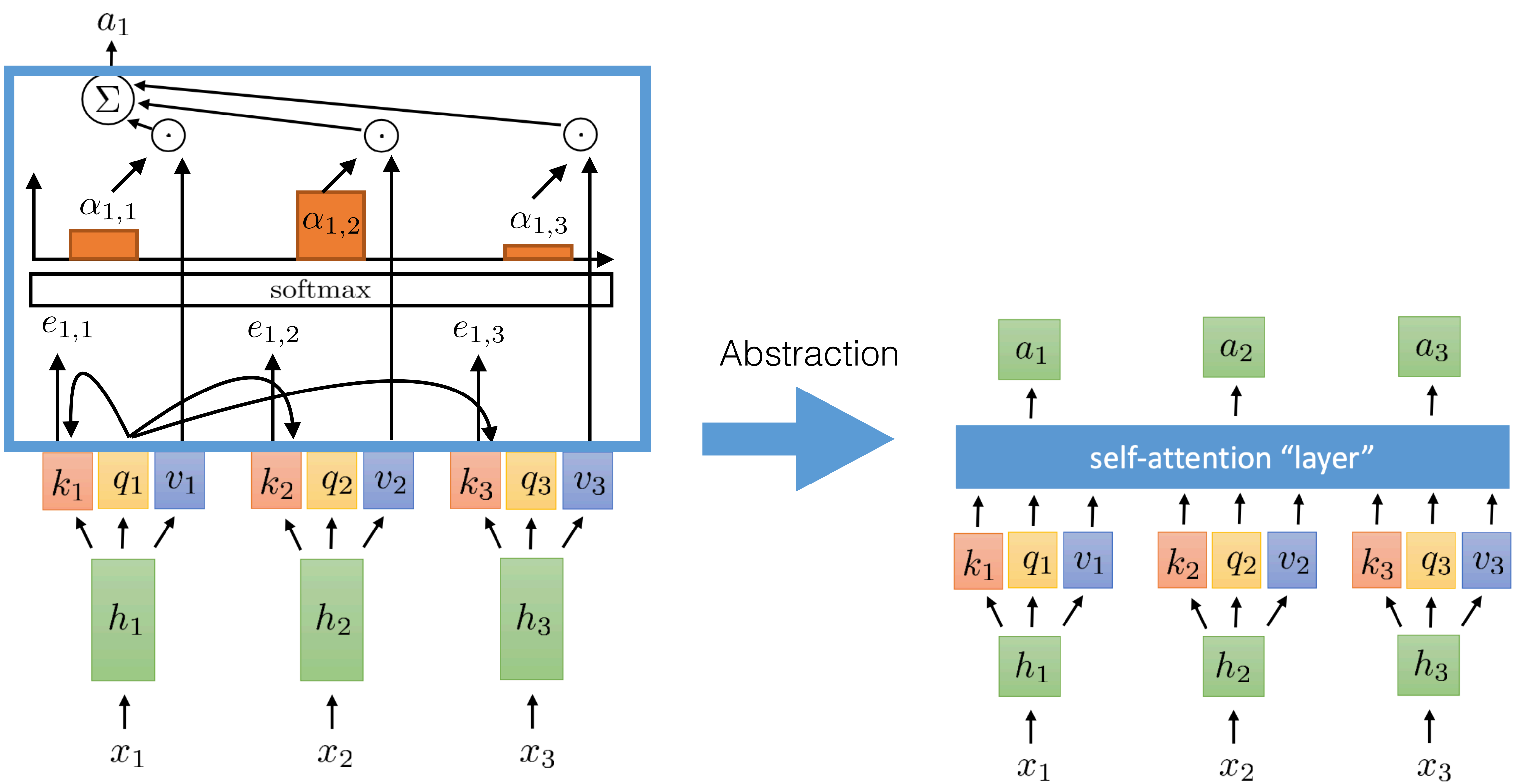
but still weight sharing:

$$h_t = \sigma(W x_t + b)$$

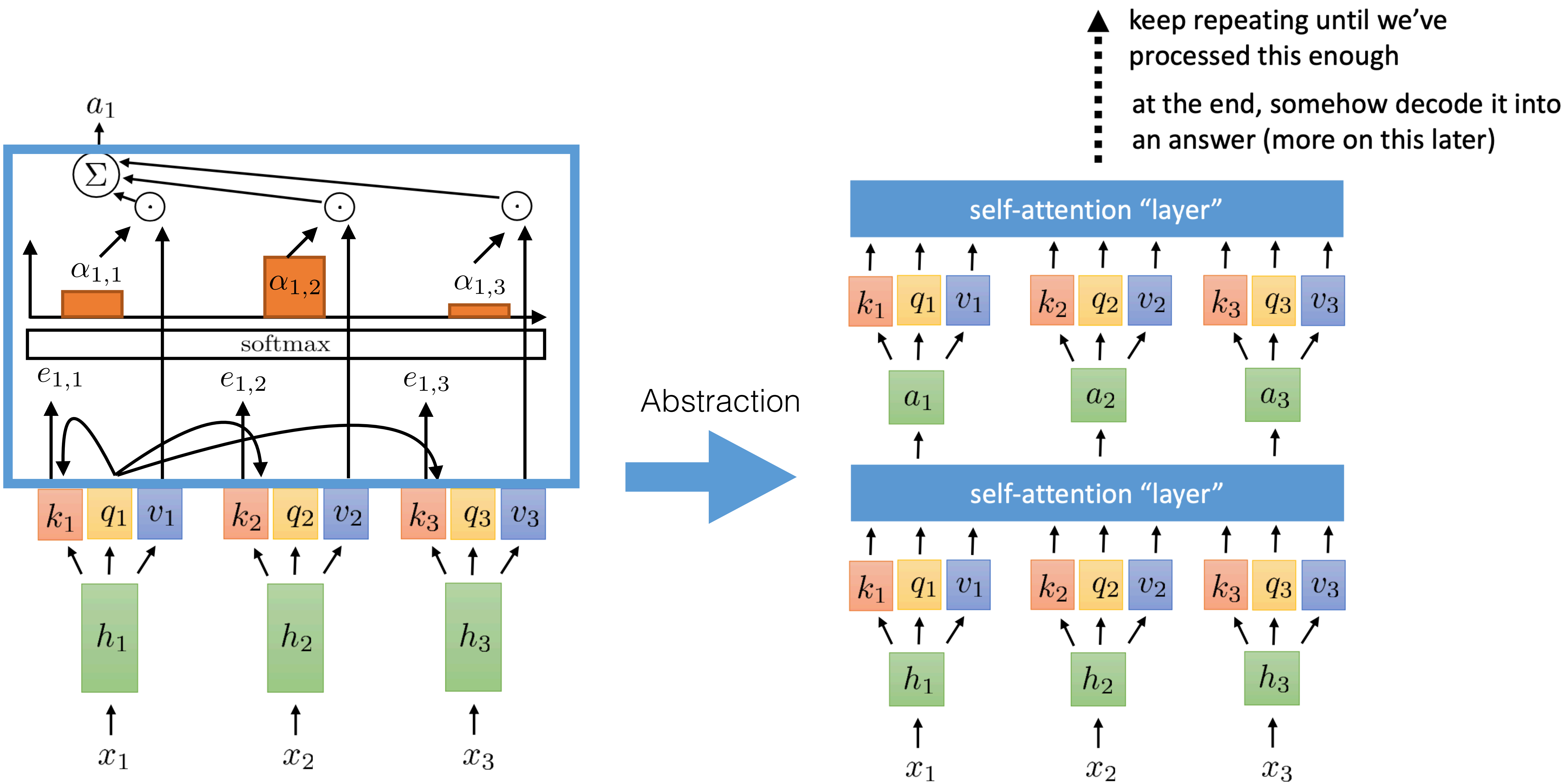
shared weights at all time steps

(or any other nonlinear function)

Self-Attention



Self-Attention



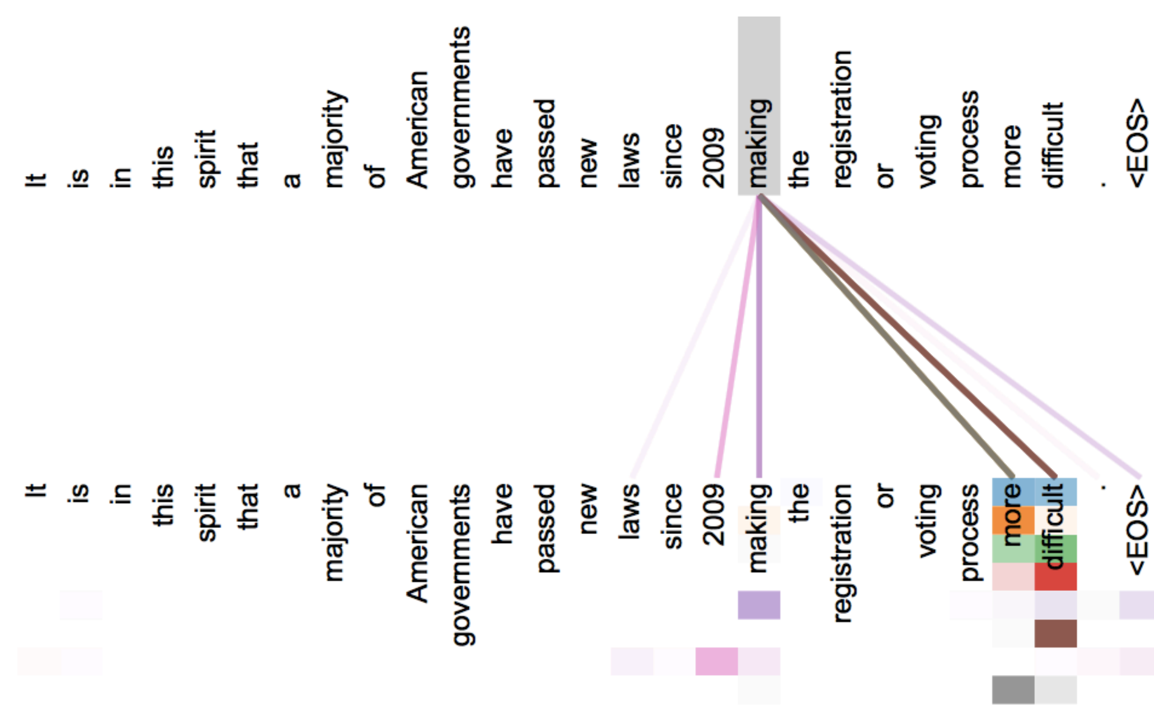
Multi-headed Attention

- **Idea:** multiple attention “heads” focus on different parts of the sentence

- e.g. Different heads for “copy” vs regular (Allamanis et al. 2016)

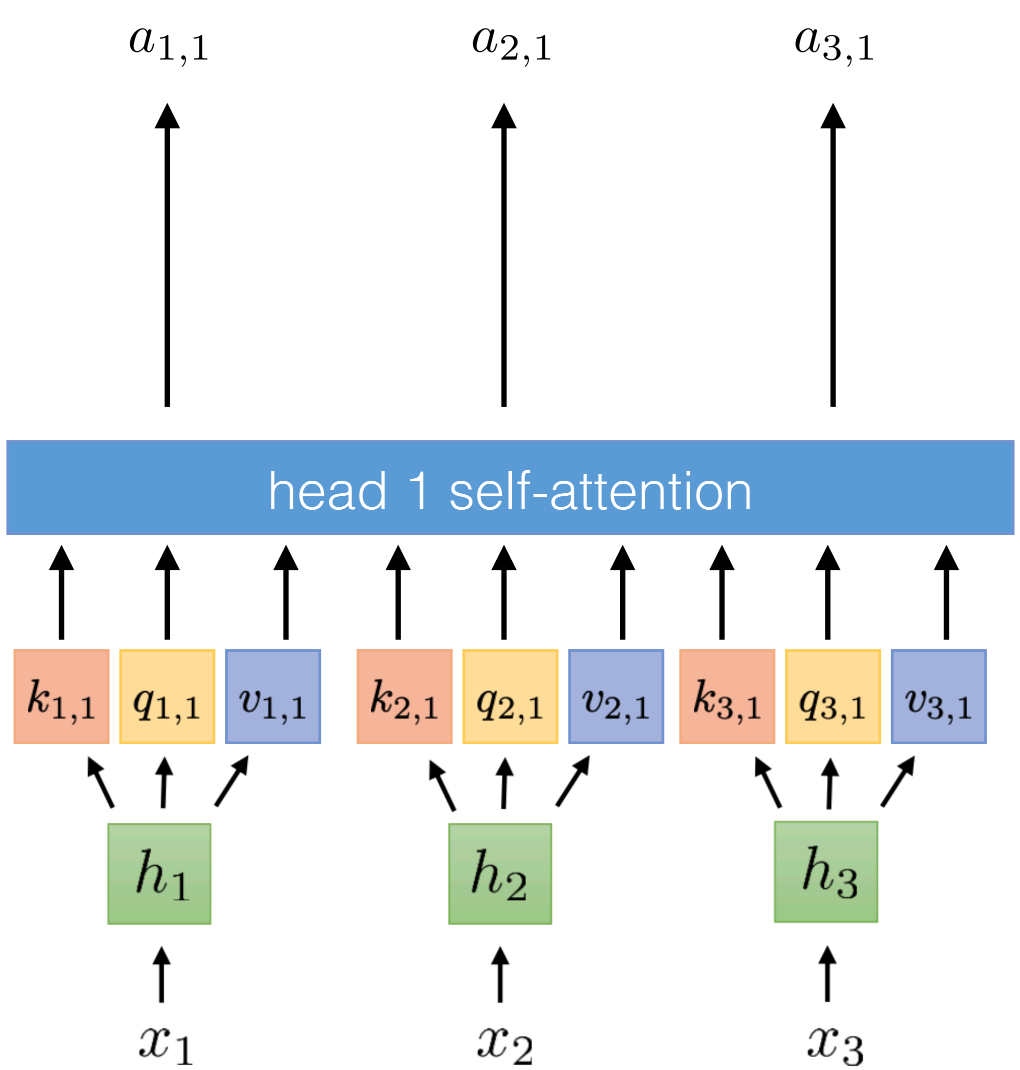
Target			Attention Vectors	λ
m_1	set	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.012
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_2	use	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.974
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_3	browser	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.969
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_4	cache	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.583
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	
m_5	END	$\boldsymbol{\alpha} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	0.066
		$\boldsymbol{\kappa} =$	<s> { this . use Browser Cache = use Browser Cache ; } </s>	

- Or multiple independently learned heads (Vaswani et al. 2017)



- Or one head for every hidden node! (Choi et al. 2018)

Multi-head attention



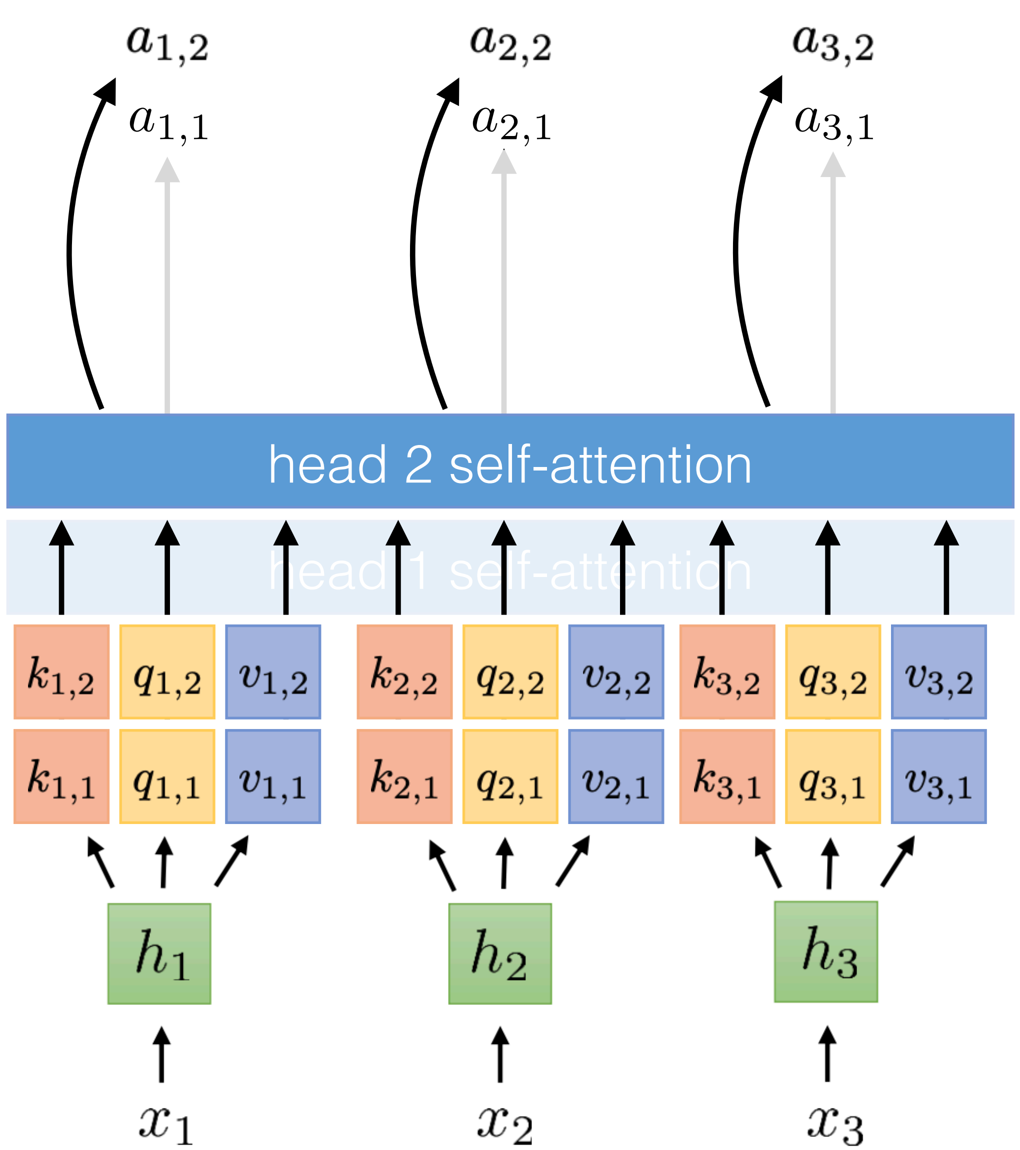
Compute weights independently for each head

$$e_{l,t,i} = q_{l,i} \cdot k_{l,i}$$

$$\alpha_{l,t,i} = \exp(e_{l,t,i}) / \sum_{t'} \exp(e_{l,t',i})$$

$$a_{l,i} = \sum_{t} \alpha_{l,t,i} v_{t,i}$$

Multi-head attention



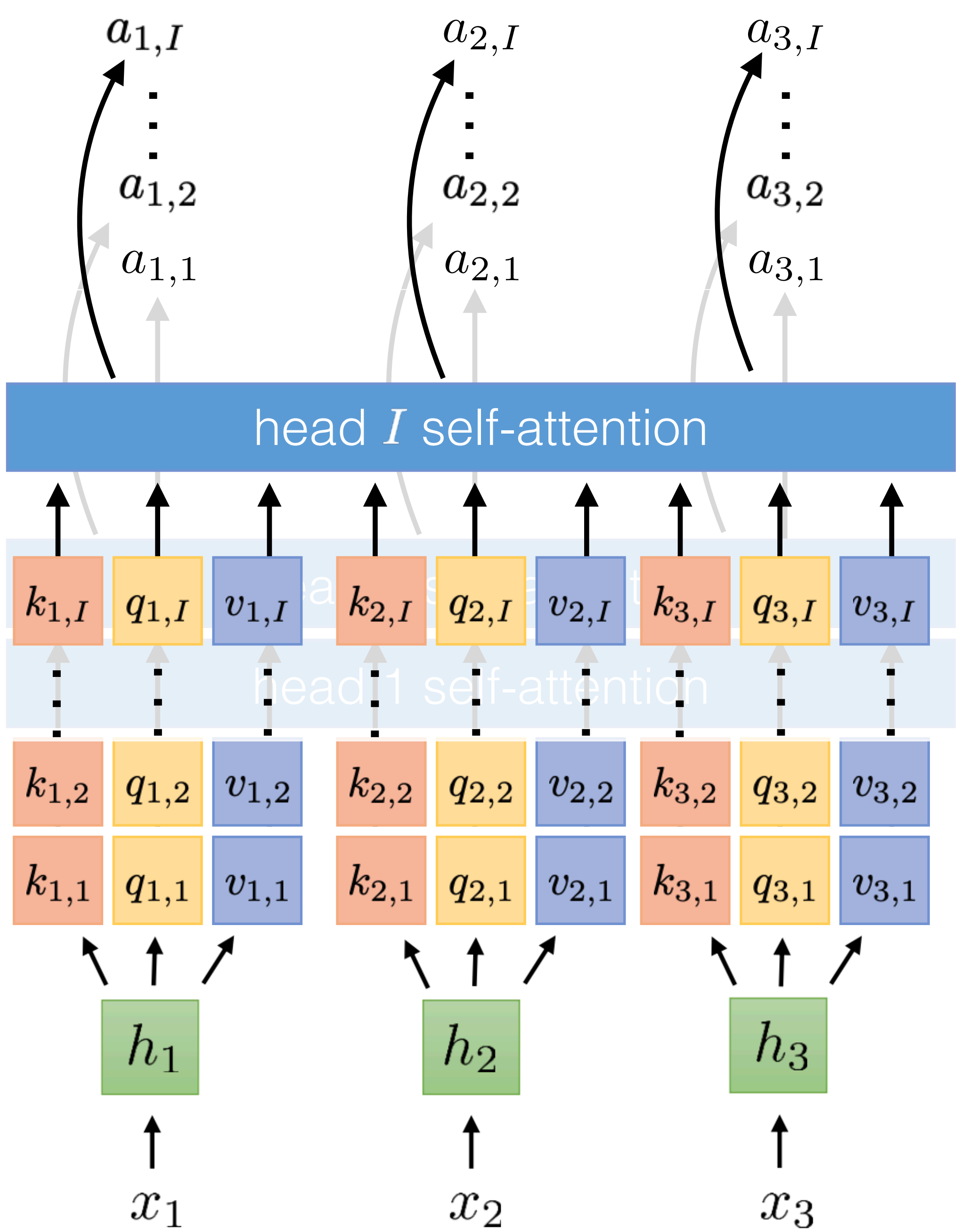
Compute weights independently for each head

$$e_{l,t,i} = q_{l,i} \cdot k_{l,i}$$

$$\alpha_{l,t,i} = \exp(e_{l,t,i}) / \sum_{t'} \exp(e_{l,t',i})$$

$$a_{l,i} = \sum_t \alpha_{l,t,i} v_{t,i}$$

Multi-head attention



$$a_l = \begin{bmatrix} a_{l,I} \\ \cdot \\ \cdot \\ \cdot \\ a_{l,2} \\ a_{l,1} \end{bmatrix} \in \mathbb{R}^d, \quad a_{l,i} \in \mathbb{R}^{\frac{d}{I}}$$

where I is the number of heads. Around 8 heads seems to work pretty well for big models

Compute weights independently for each head

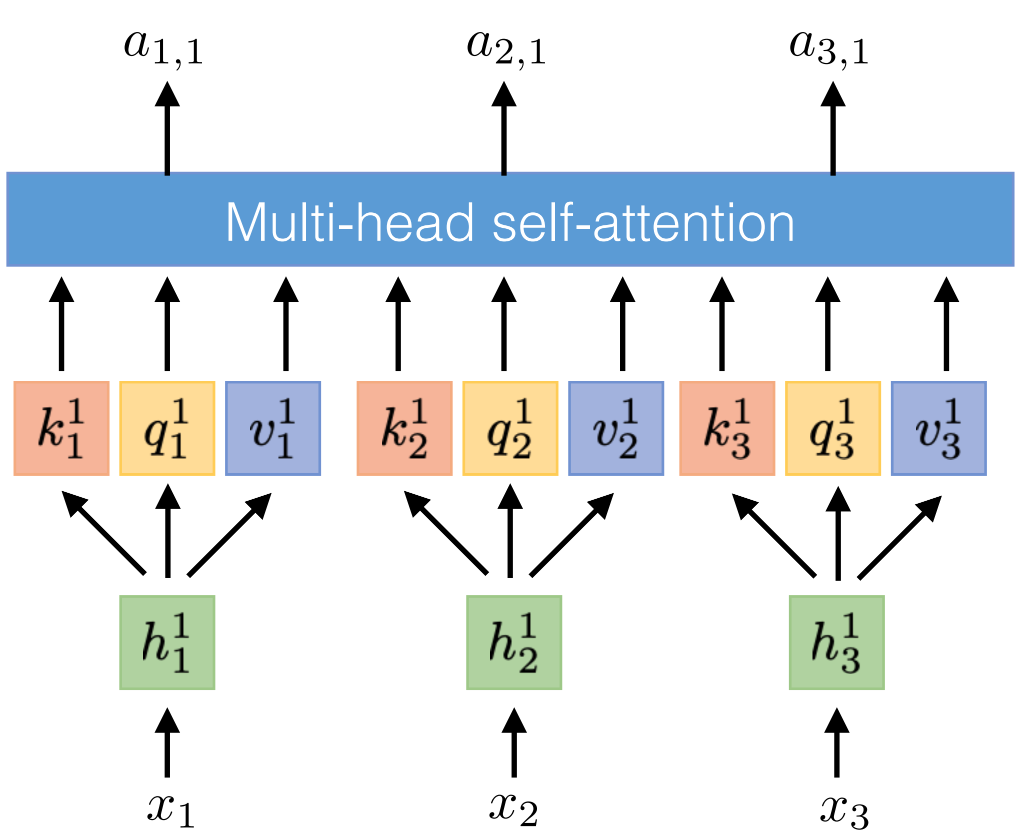
$$e_{l,t,i} = q_{l,i} \cdot k_{l,i}$$

$$\alpha_{l,t,i} = \exp(e_{l,t,i}) / \sum_{t'} \exp(e_{l,t',i})$$

$$a_{l,i} = \sum_{t} \alpha_{l,t,i} v_{t,i}$$

Self-attention is still linear

- Every self-attention “layer” is a linear transformation of the previous layer (with non-linear attention weights)
- This is not very expressive to learn from the complex data



$$k_t = W_k h_t \qquad q_t = W_q h_t \qquad v_t = W_v h_t$$

$$\alpha_{l,t} = \exp(e_{l,t}) / \sum_{t'} \exp(e_{l,t'})$$

$$e_{l,t} = q_l \cdot k_t$$

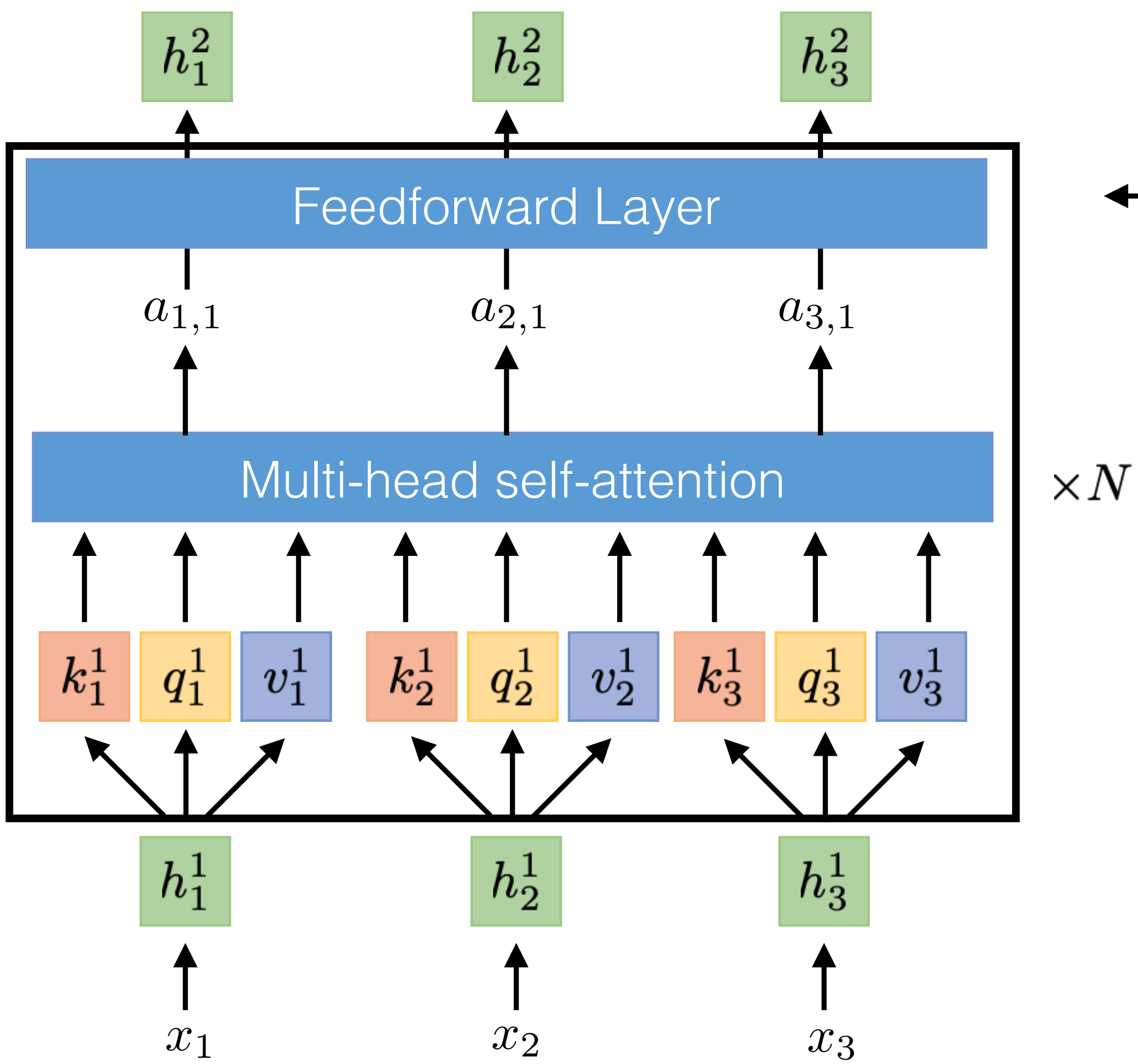
$$a_l = \sum_t \alpha_{l,t} v_t = \sum_t \alpha_{l,t} W_v h_t = W_v \sum_t \alpha_{l,t} h_t$$

linear transformation

non-linear weights

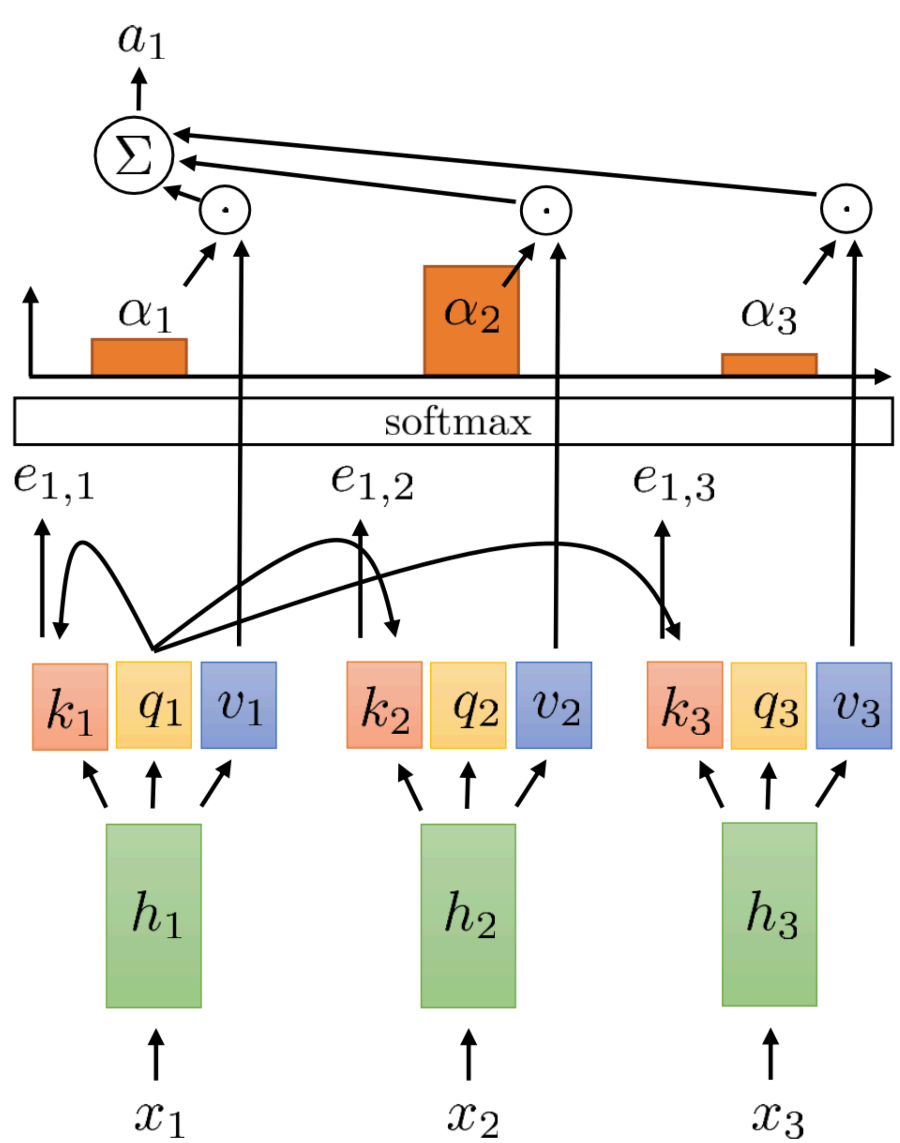
Alternating self-attention & nonlinearity

- Each transformer layer contains a multi-head self-attention layer and a feedforward layer.
- We alternate self-attention and non-linear layer *N* times, namely stack *N* transformer layers.



some non-linear (learned) function
e.g., $h_t^\ell = \sigma(W^\ell a_t^\ell + b^\ell)$

Positional encoding



what we see:

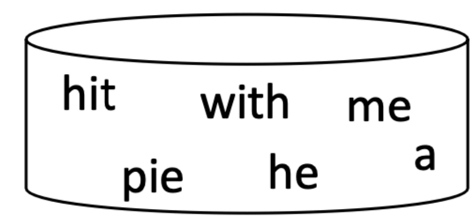
he hit me with a pie

what naïve self-attention sees:

a pie hit me with he

a hit with me he pie

he pie me with a hit



most alternative orderings are nonsense, but some change the meaning

in general the position of words in a sentence carries information!

Idea: add some information to the representation at the beginning that indicates where it is in the sequence!

$$h_t = f(x_t, t)$$

some function

Positional encoding: sin/cos

Naïve positional encoding: just append t to the input $\qquad \bar{x}_t = \begin{bmatrix} x_t \\ t \end{bmatrix}$

This is not a great idea, because **absolute** position is less important than **relative** position

I walk my dog every day

every single day I walk my dog

The fact that "my dog" is right after "I walk" is the important part, not its absolute position

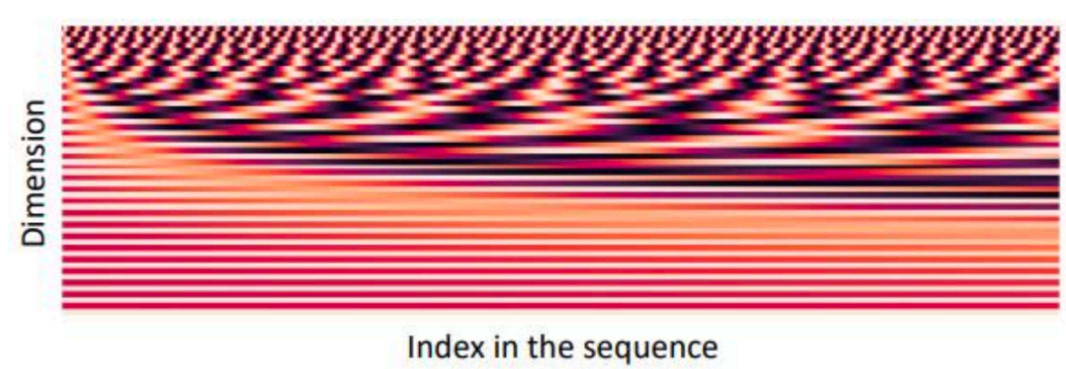
we want to represent **position** in a way that tokens with similar **relative** position have similar **positional encoding**

Idea: what if we use **frequency-based** representations?

$$p_t = \begin{bmatrix} \sin(t/10000^{2*1/d}) \\ \cos(t/10000^{2*1/d}) \\ \sin(t/10000^{2*2/d}) \\ \cos(t/10000^{2*2/d}) \\ \dots \\ \sin(t/10000^{2*\frac{d}{2}/d}) \\ \cos(t/10000^{2*\frac{d}{2}/d}) \end{bmatrix}$$

dimensionality of positional encoding

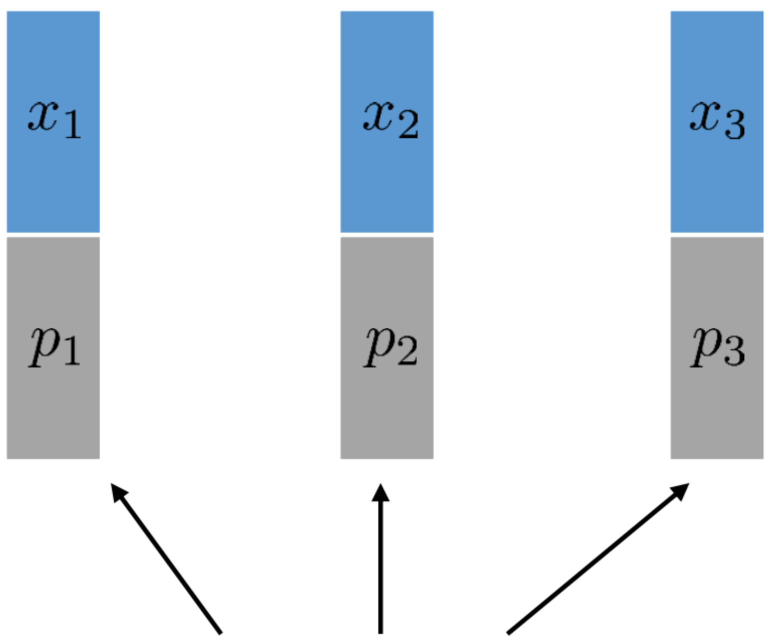
"even-odd" indicator



"first-half vs. second-half" indicator

Positional encoding: learned

Another idea: just learn a positional encoding



Different for every input sequence

The same learned values for every sequence

but different for different time steps

dimensionality

max sequence length

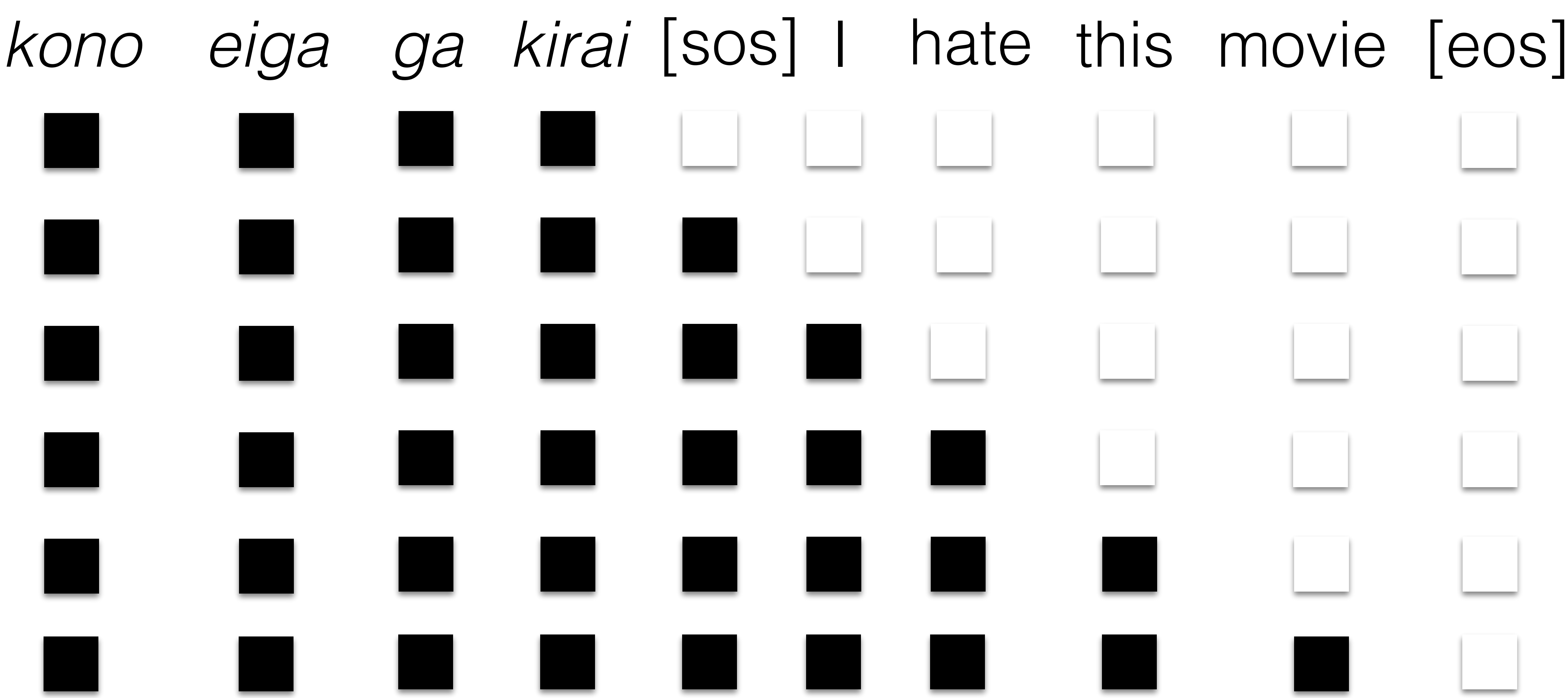
How many values do we need to learn? $P = [p_1, p_2, \ldots, p_T] \in R^{d \times T}$

+ more flexible (and perhaps more optimal) than sin/cos encoding

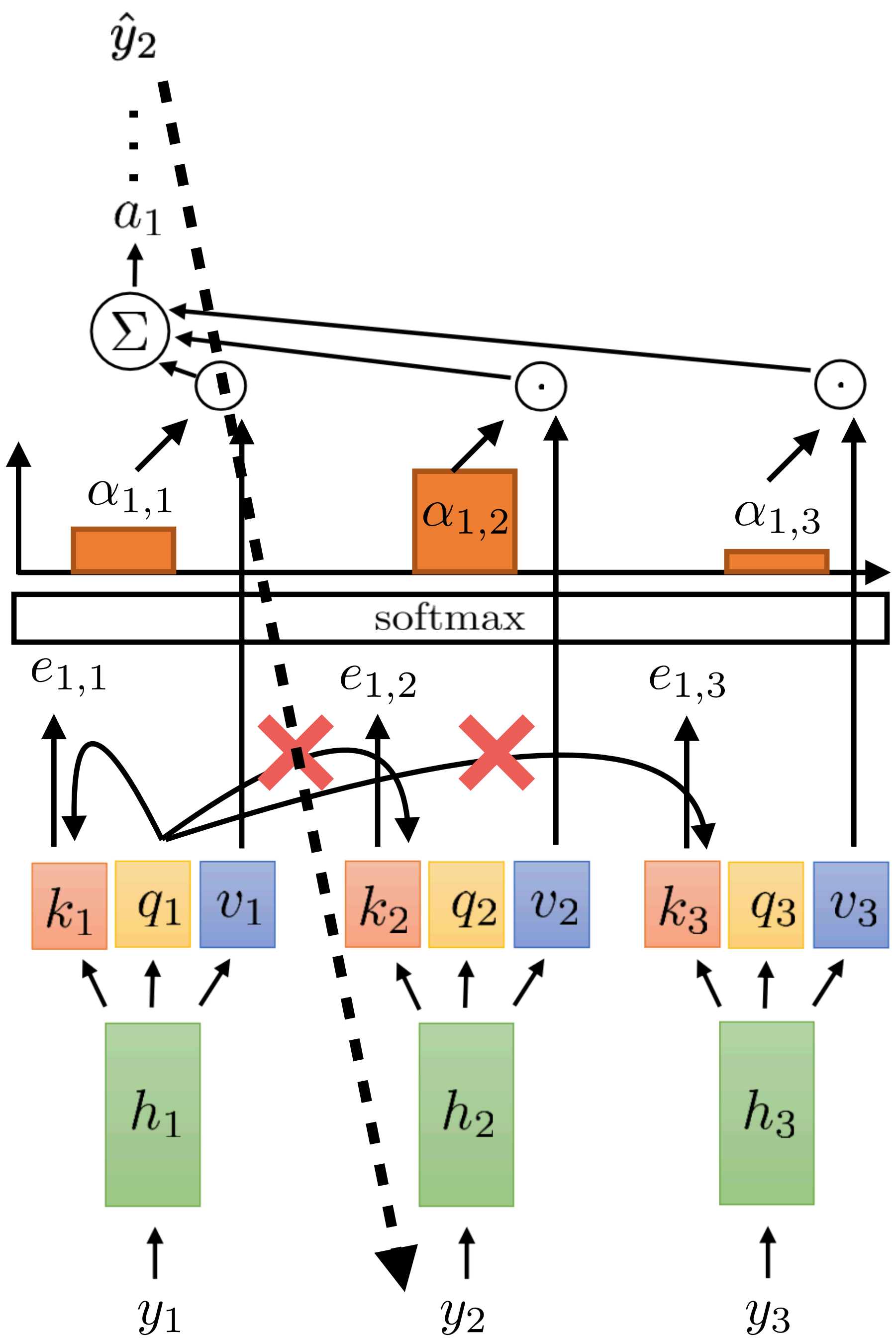
+ a bit more complex, need to pick a max sequence length (and can't generalize beyond it)

Masked attention for Target sentence

- For the conditioned prediction, we aim to predict the current target word based on its past words and the source input, i.e., $P(y_i|X, y_{<i})$
- We can do so by "masking" the results for the output



Masked attention for Target sentence



- **At test time**, the predicted token will be feed as input to the next time step
- We must design a masking to allow self-attention on the **past tokens**, but not on the **future tokens**.

Easy solution:

$$\cancel{e_{l,t} = q_l \cdot k_t}$$

$$e_{l,t} = \begin{cases} q_l \cdot k_t & \text{if } l \geq t \\ -\infty & \text{otherwise} \end{cases}$$

in practice:

just replace $\exp(e_{l,t})$ with 0 if $l < t$

inside the softmax

Multiply the attention matrix by 0-1 masking matrix

Attention Tricks

- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Remove bias in dot product when using large networks
- **Positional Encodings:** Make sure that even if we don't have RNN, can still distinguish positions

Training Tricks

- **Layer Normalization:** Help ensure that layers remain in reasonable range
- **Specialized Training Schedule:** Adjust default learning rate of the Adam optimizer
- **Label Smoothing:** Insert some uncertainty in the training process
- **Masking for Efficient Training**

Code Walk:
The Annotated Transformer

https://nlp.seas.harvard.edu/2018/04/03/attention.html

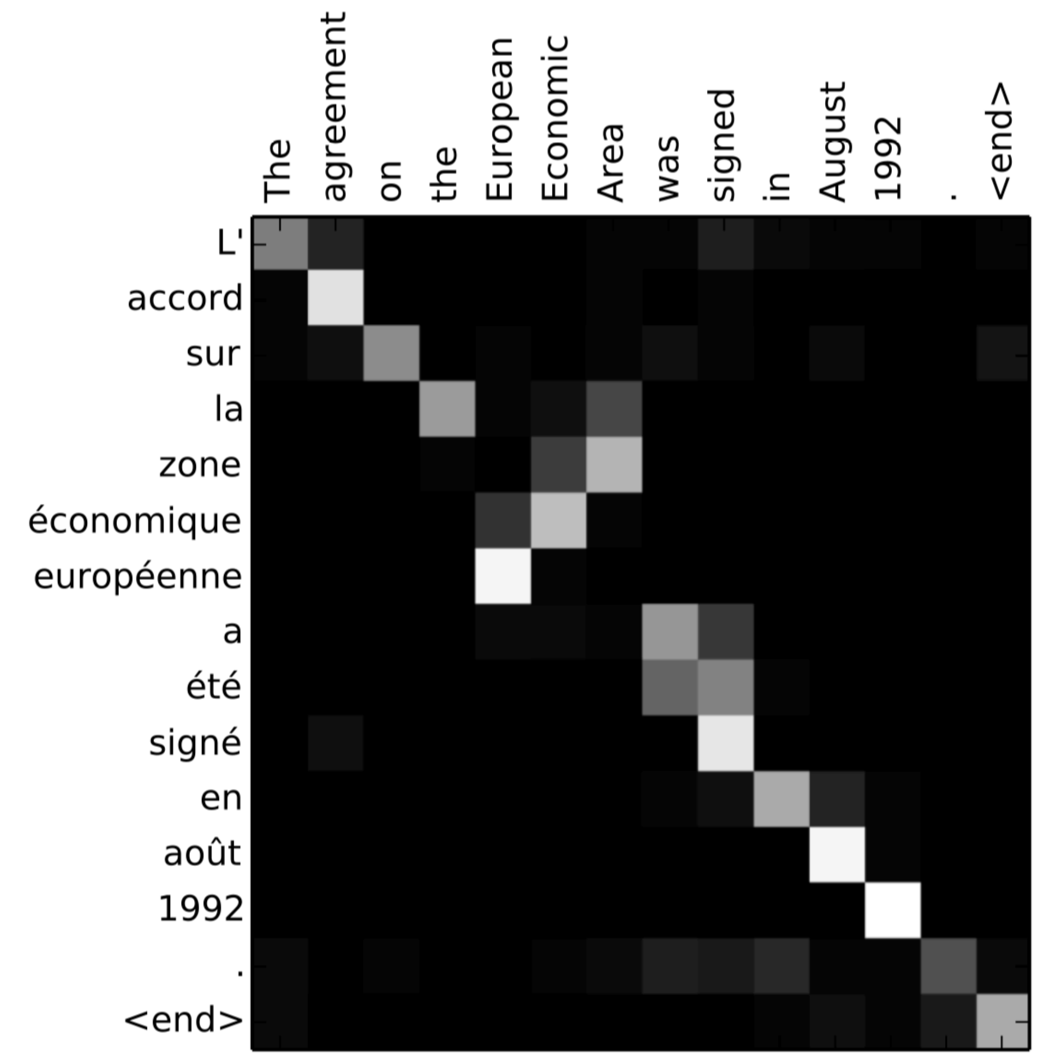
A Caveat: Attention Is Not All You Need?

- Transformers are very popular, for good reason, but
- They can be **slow to decode** at test time (Zhang et al. 2018)
- They **don't necessarily outperform RNNs** on the decoder side of seq2seq tasks (Chen et al. 2018)
- They can be **hard to train on small data** (Nguyen and Salazar 2019)
- Use them, but also be aware of limitations!

Better Modeling for Attention

Incorporating Markov Properties
(Cohn et al. 2015)

- **Intuition:** attention from last time tends to be correlated with attention this time



- Add information about the last attention when making the next decision

Hard Attention

- Instead of a soft interpolation, make a **zero-one decision** about where to attend (Xu et al. 2015)
    - Harder to train, requires methods such as reinforcement learning (see later classes)
- Perhaps this helps interpretability? (Lei et al. 2016)

Review

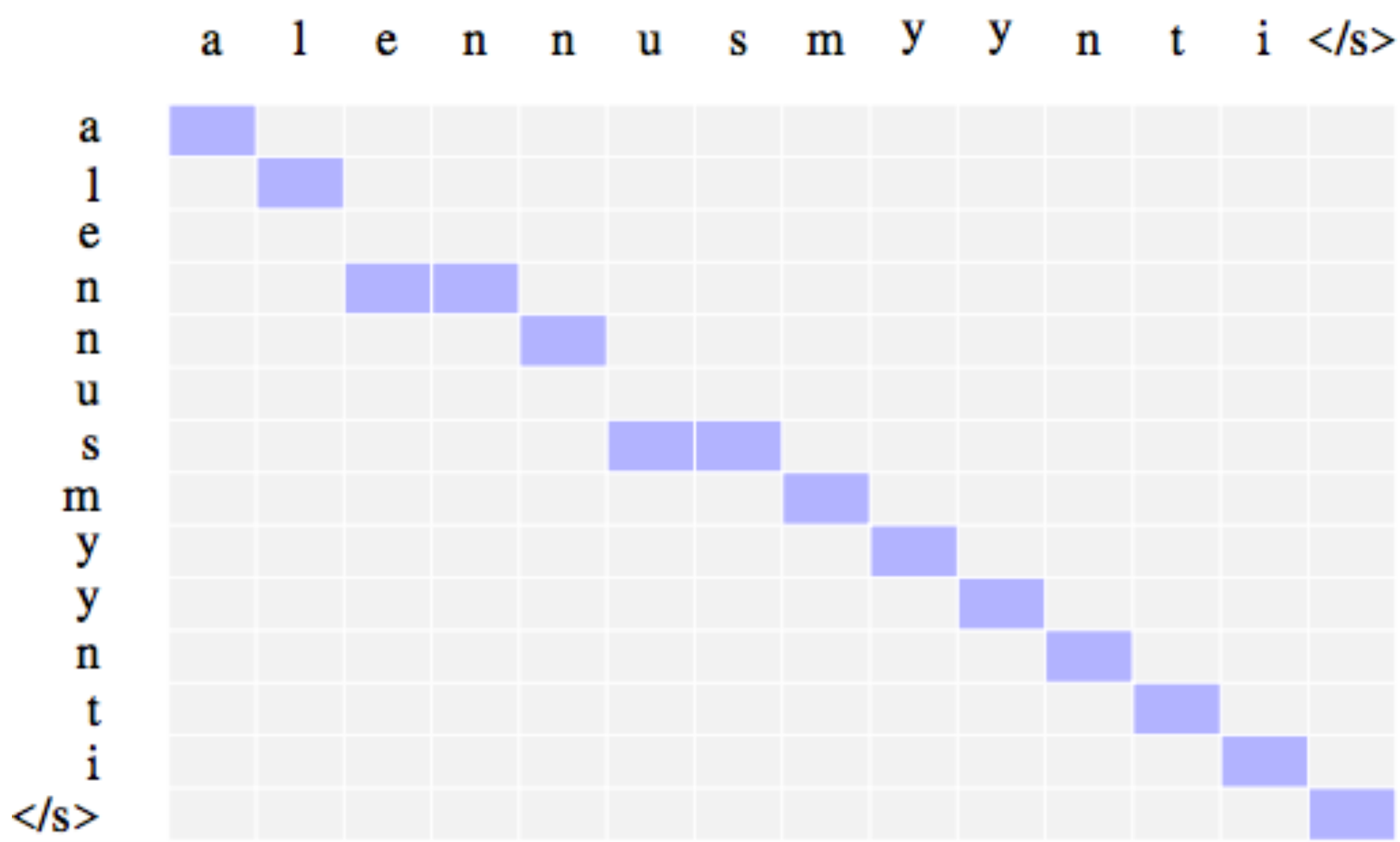
the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of caramel and happiness bound together.

Ratings

Look: 5 stars | *Smell:* 4 stars

Monotonic Attention
(e.g. Yu et al. 2016)

- In some cases, we might know the output will be the same order as the input
  - Speech recognition, incremental translation, morphological inflection (?), summarization (?)



- **Basic idea:** hard decisions about whether to read more

Better Training for Attention

Coverage

- **Problem:** Neural models tends to drop or repeat content
- **Solution:** Model how many times words have been covered
    - Impose a penalty if attention not approx.1 over each word (Cohn et al. 2015)
    - Add embeddings indicating coverage (Mi et al. 2016)

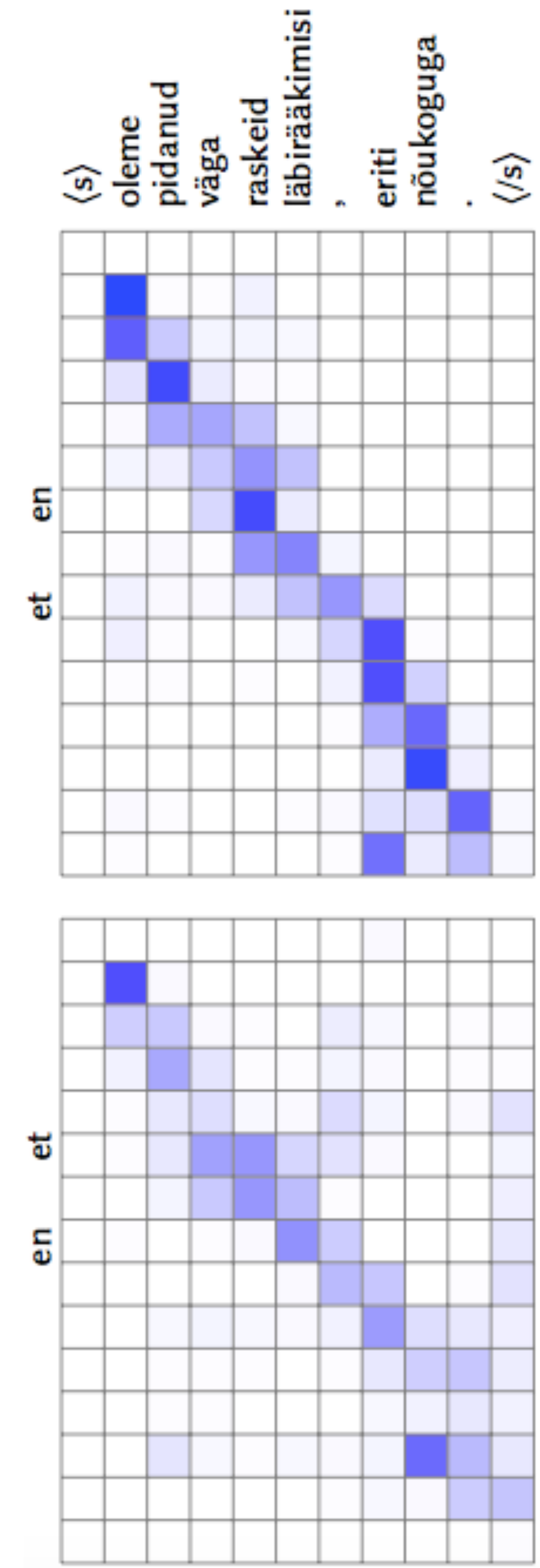
Bidirectional Training

(Cohn et al. 2015)

- **Intuition:** Our attention should be roughly similar in forward and backward directions

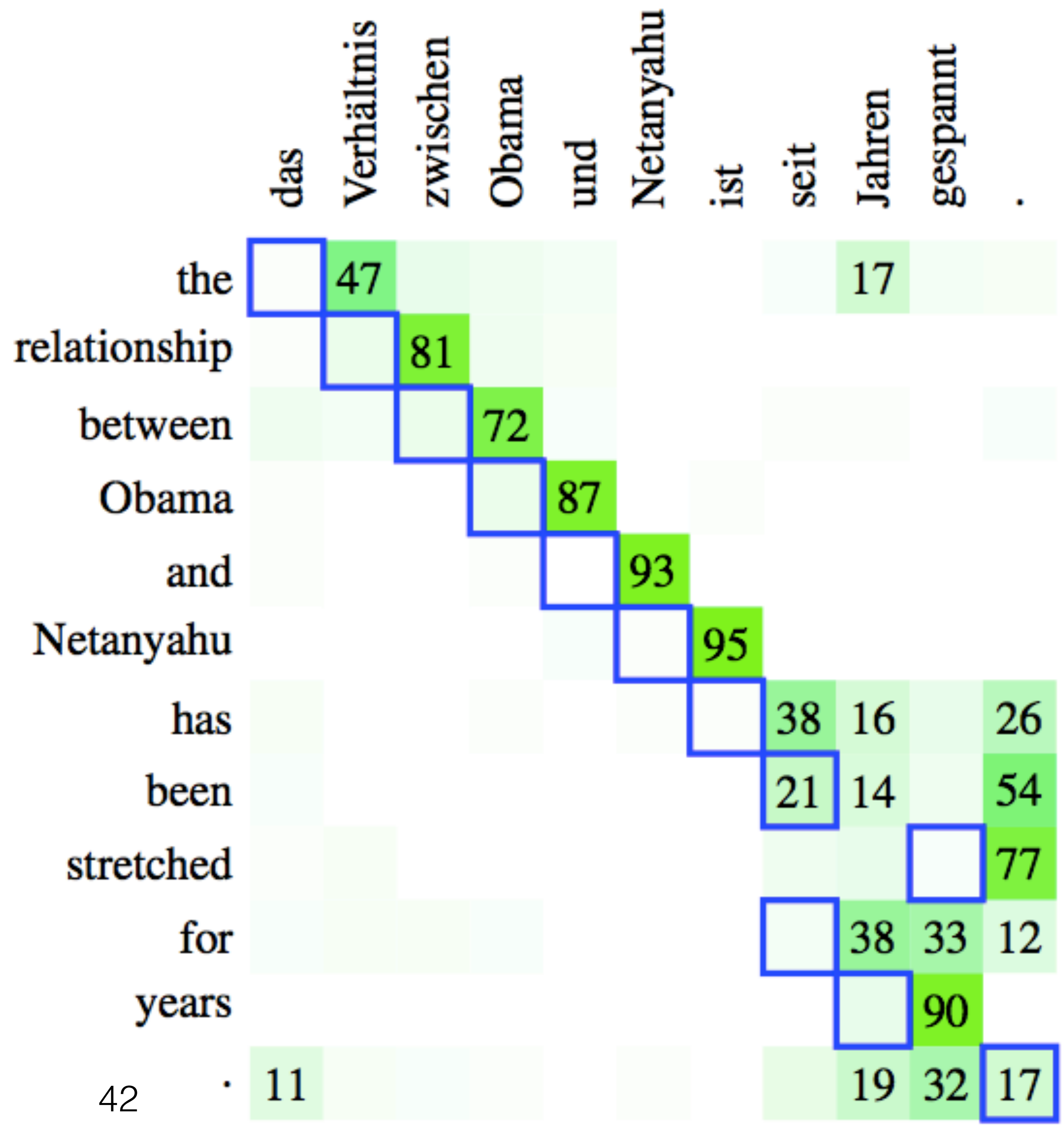
- **Method:** Train so that we get a bonus based on the trace of the matrix product for training in both directions

$$\mathrm{tr}(A_{X \rightarrow Y} A_{Y \rightarrow X}^{\intercal})$$



Attention is not Alignment!
(Koehn and Knowles 2017)

- Attention is often blurred
- Attention is often off by one
- It can even be manipulated to be non-intuitive! (Jain and Wallace 2019, Pruthi et al. 2020)



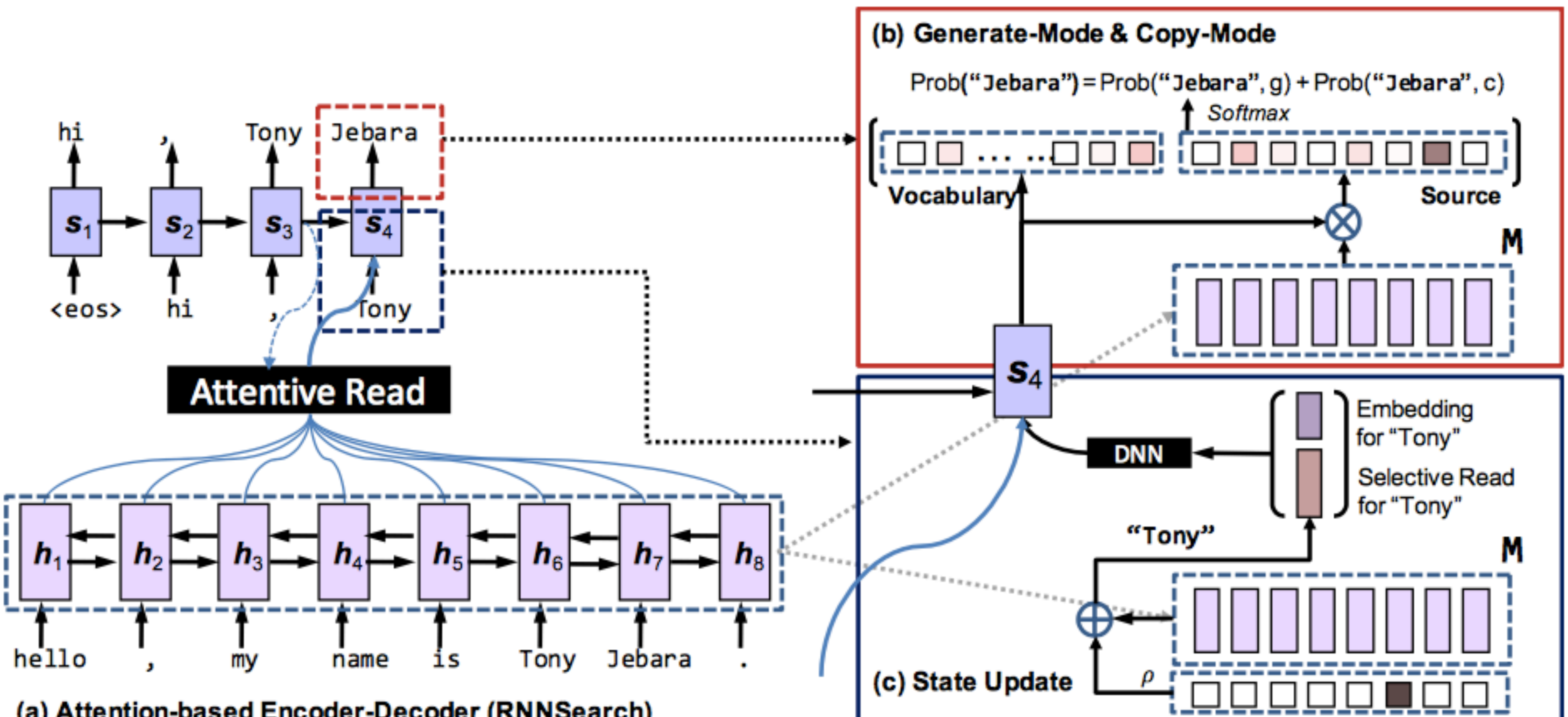
Supervised Training
(Mi et al. 2016)

- Sometimes we can get “gold standard” alignments *a-priori*
  - Manual alignments
  - Pre-trained with strong alignment model
- **Train the model to match** these strong alignments

What Else Can We Attend To?

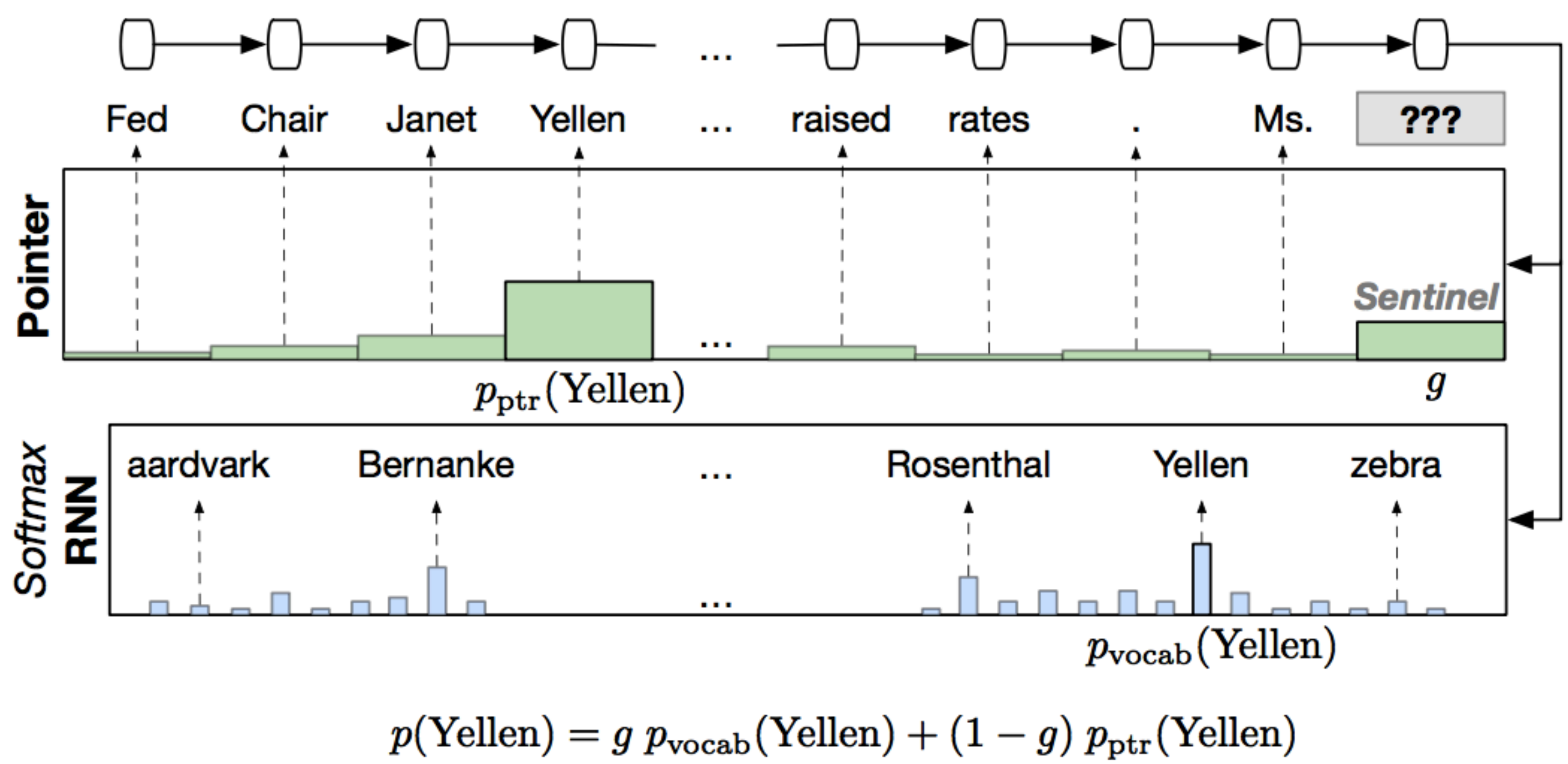
Copy Mechanisms

- Like the previous explanation
- But also, more directly through a *copy mechanism* (Gu et al. 2016)



Copying from History

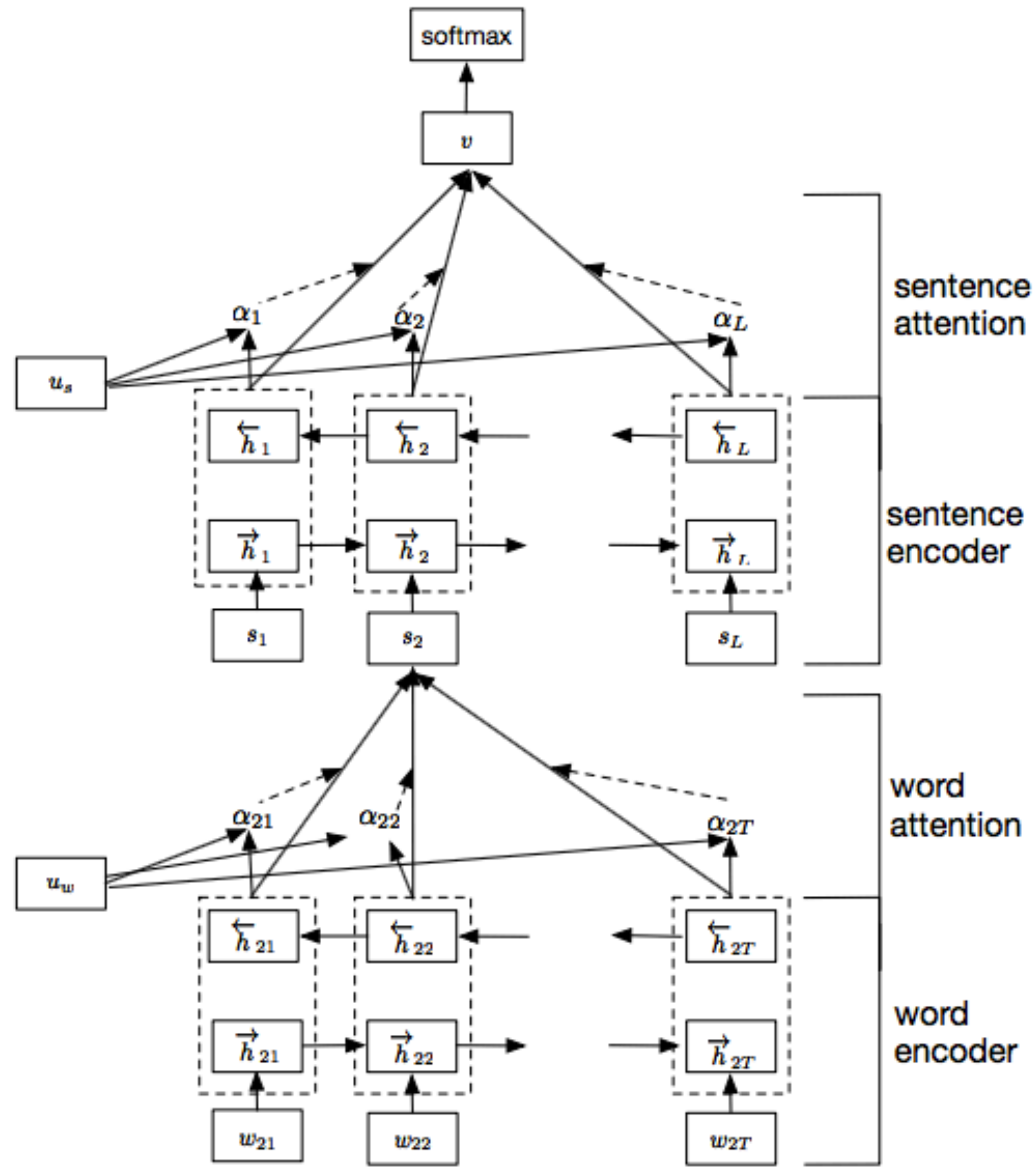
- In language modeling, attend to the previous words (Merity et al. 2016)



- In translation, attend to either input or previous output (Vaswani et al. 2017)

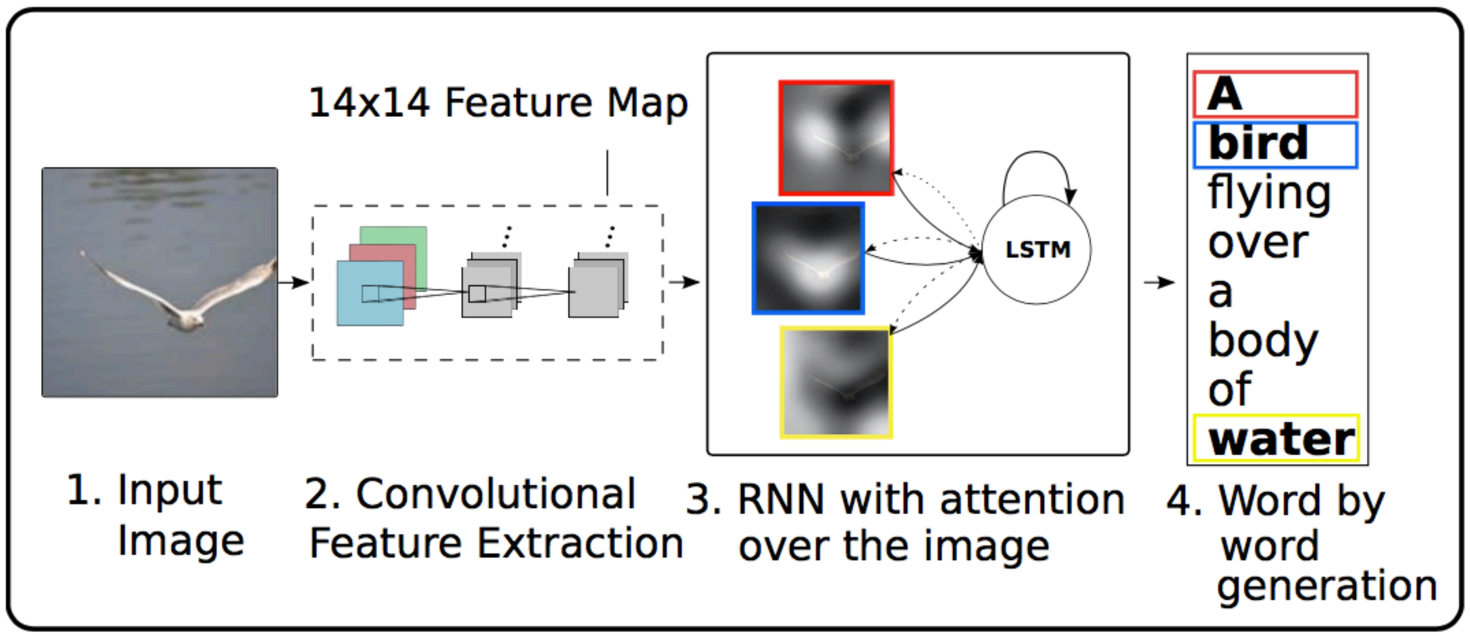
Hierarchical Structures
(Yang et al. 2016)

- Encode with attention over each sentence, then attention over each sentence in the document

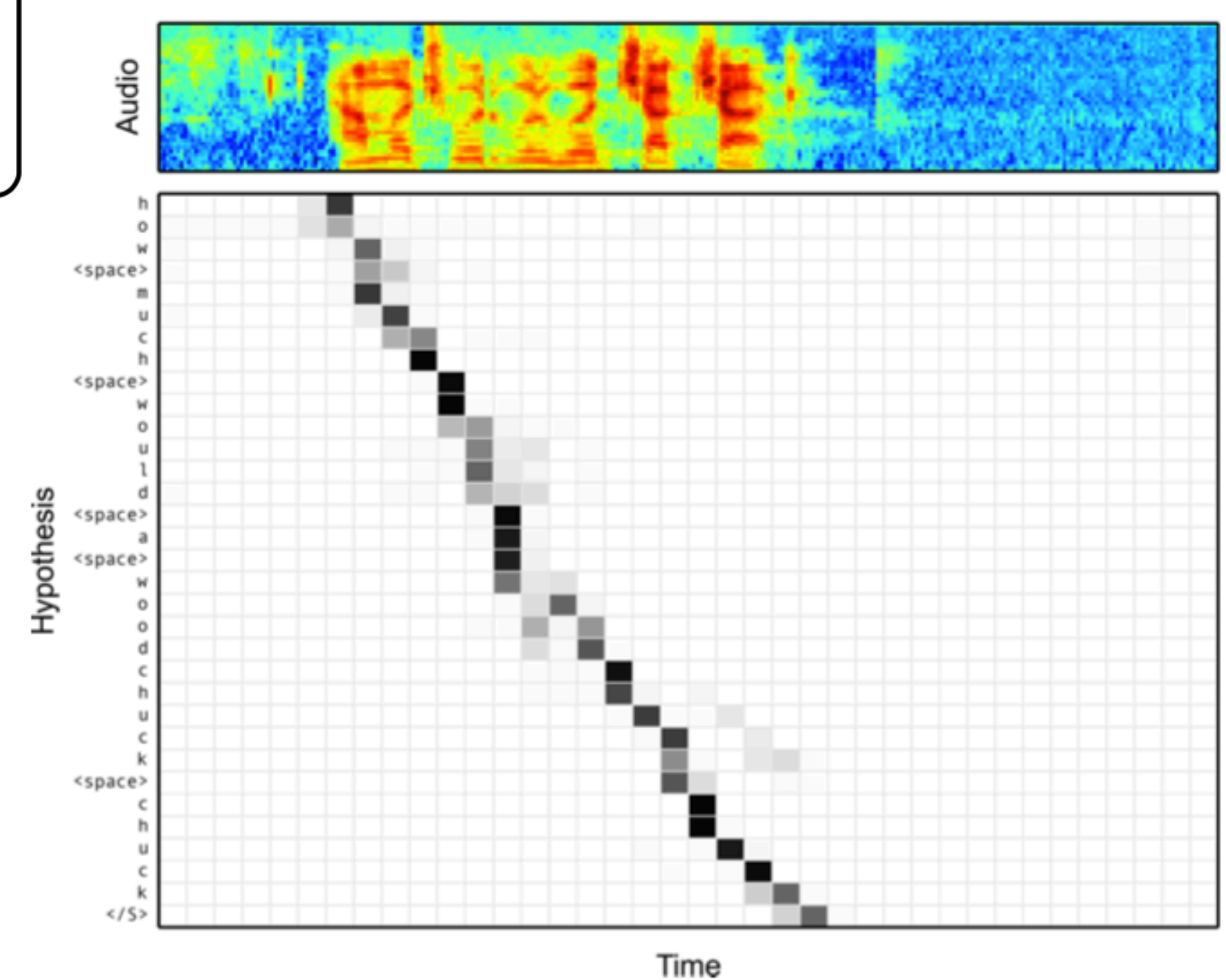


Various Modalities

- Images (Xu et al. 2015)

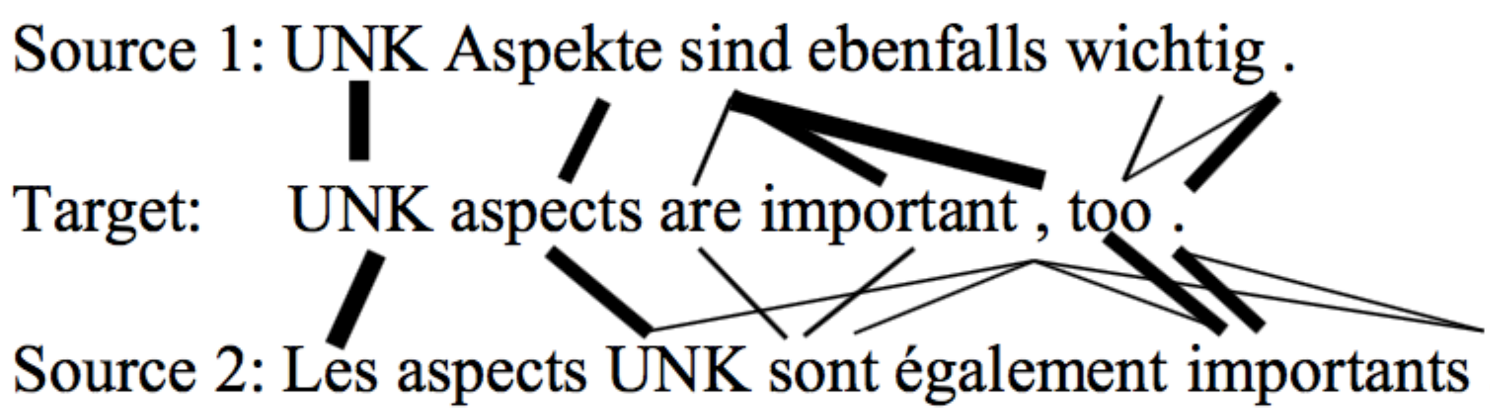


- Speech (Chan et al. 2015)



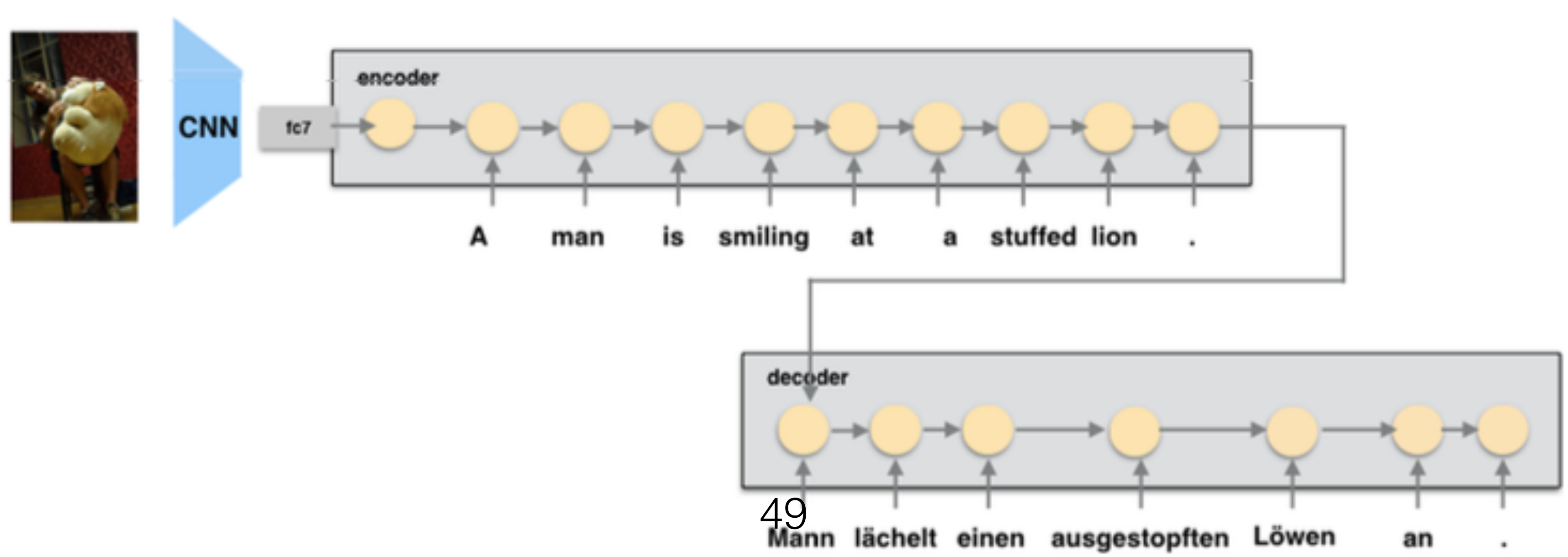
Multiple Sources

- Attend to multiple sentences (Zoph et al. 2015)



  - Libovicky and Helcl (2017) compare multiple strategies

- Attend to a sentence and an image (Huang et al. 2016)



Questions?