CS11-711 Advanced NLP

Attention and Transformers

Daniel Fried and Robert Frederking
with slides from Graham Neubig

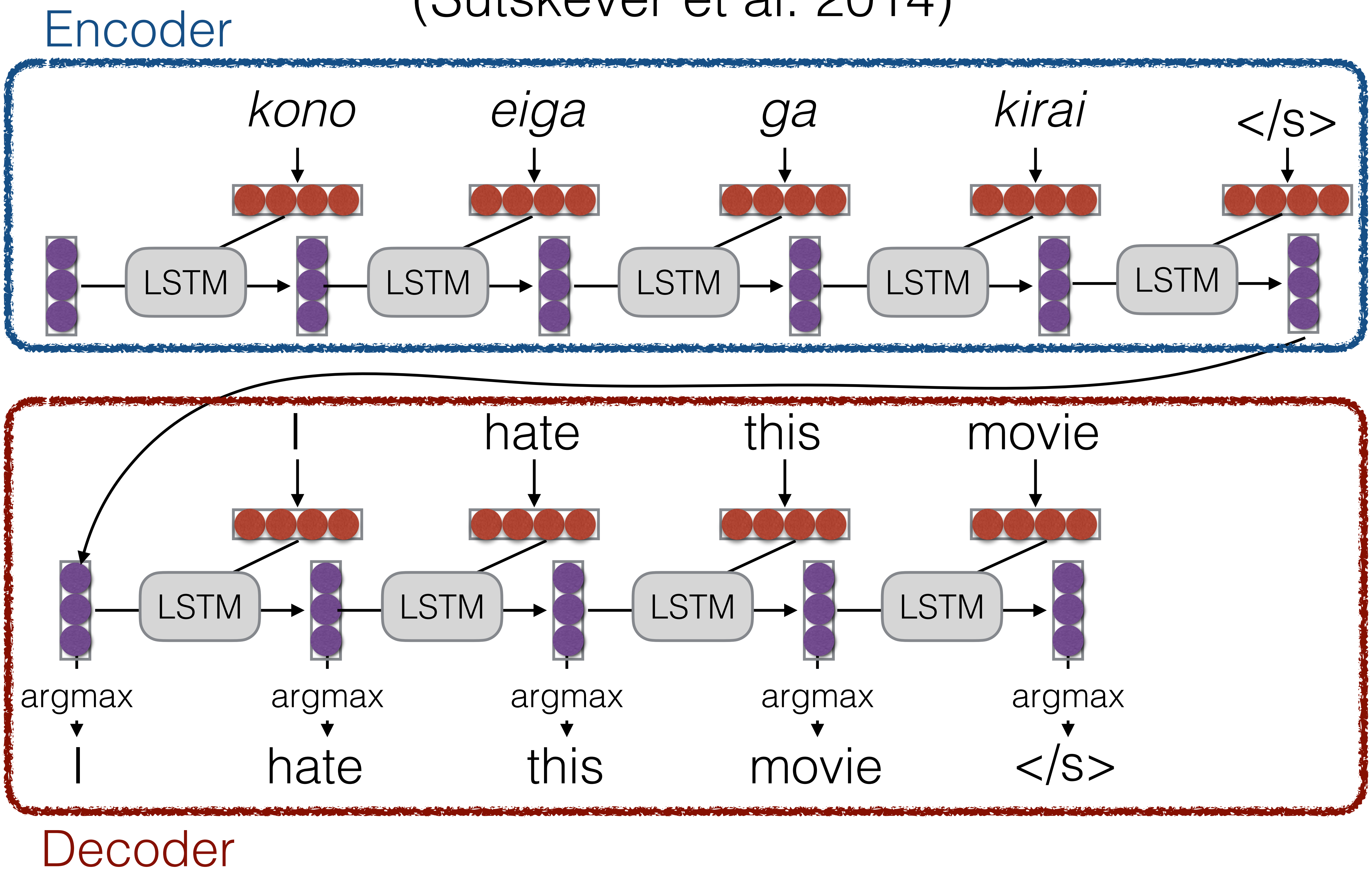
Carnegie Mellon University
Language Technologies Institute

Site

https://cmu-anlp.github.io/

Encoder-decoder Models

(Sutskever et al. 2014)

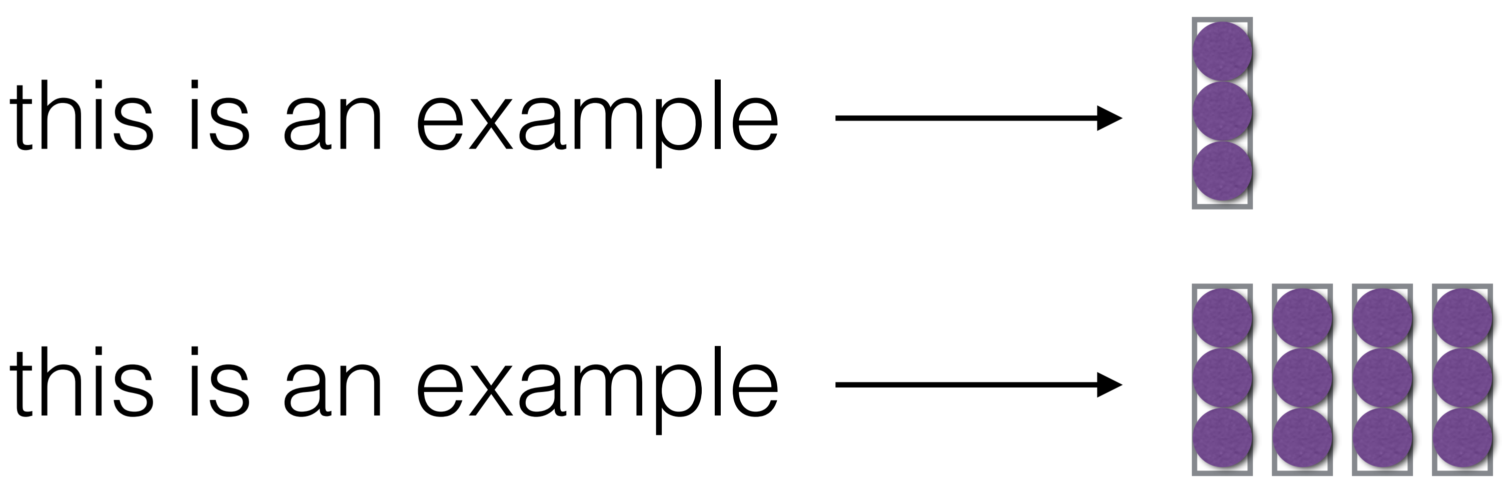


Sentence Representations

<u>Problem!</u>

> "You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"
> — Ray Mooney

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



Attention

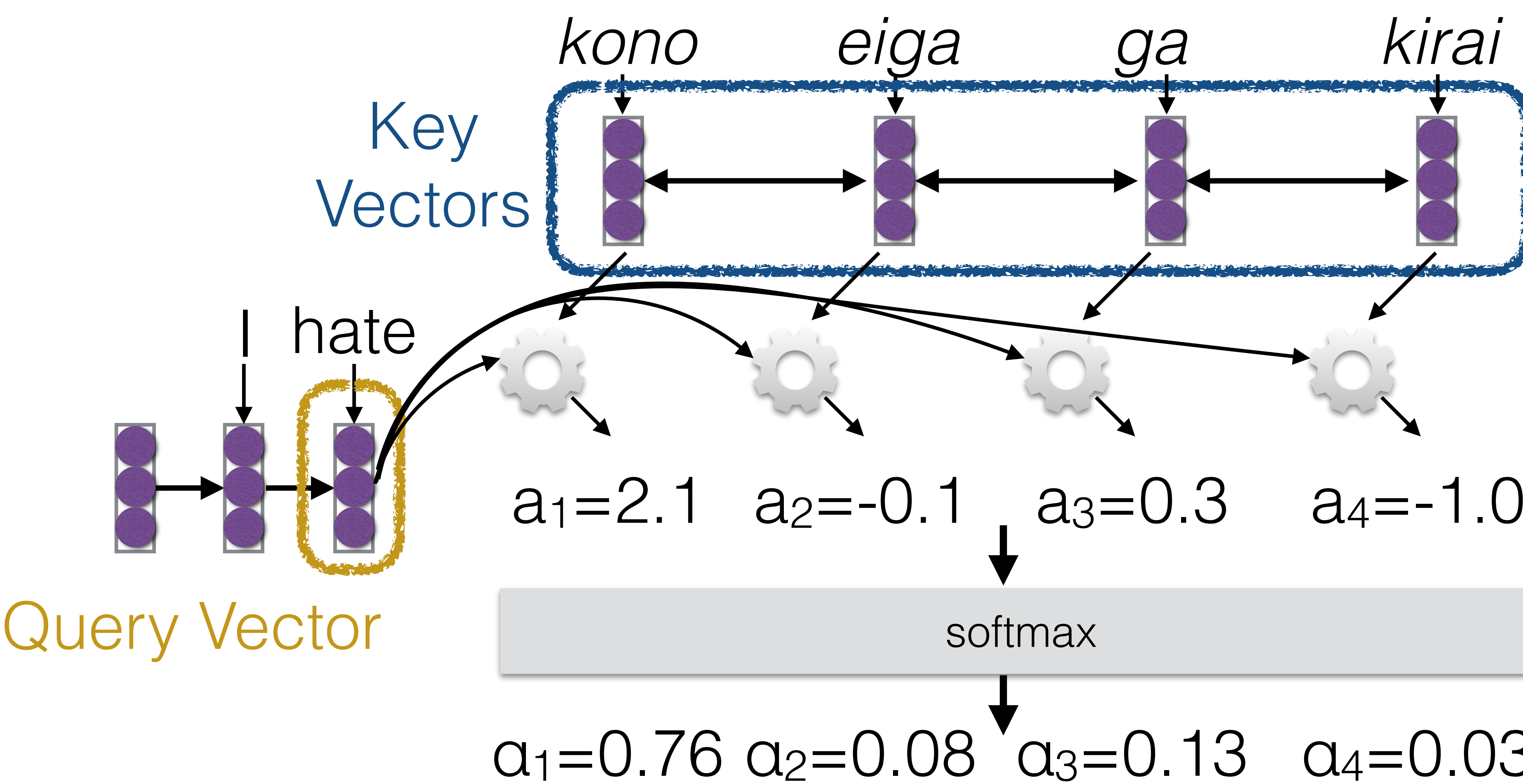
Basic Idea

(Bahdanau et al. 2015)

- 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

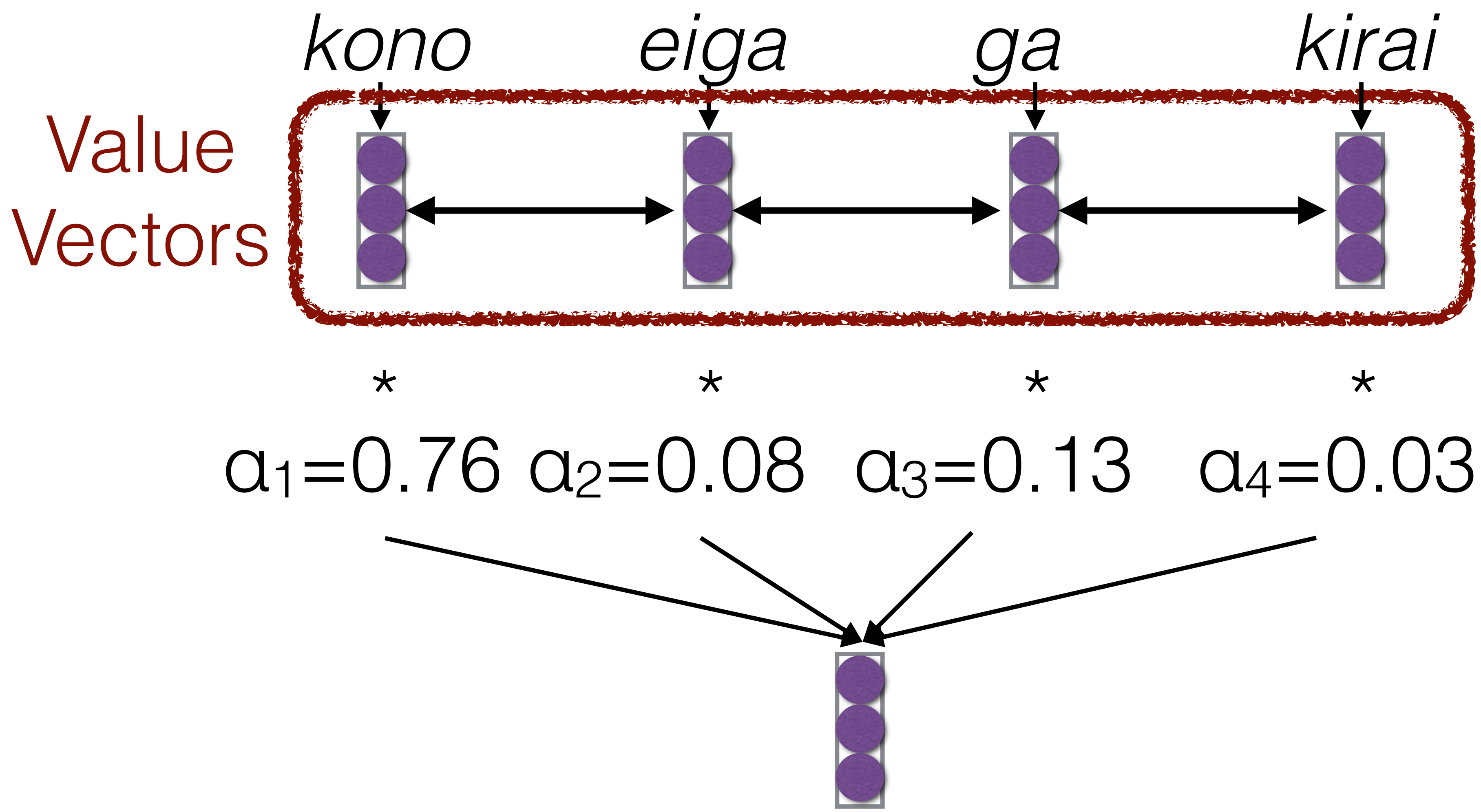
Calculating Attention (1)

- Use "query" vector (decoder state) and "key" vectors (all encoder states)
- For each query-key pair, calculate weight
- Normalize to add to one using softmax



Calculating Attention (2)

- Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



- Use this in any part of the model you like

A Graphical Example

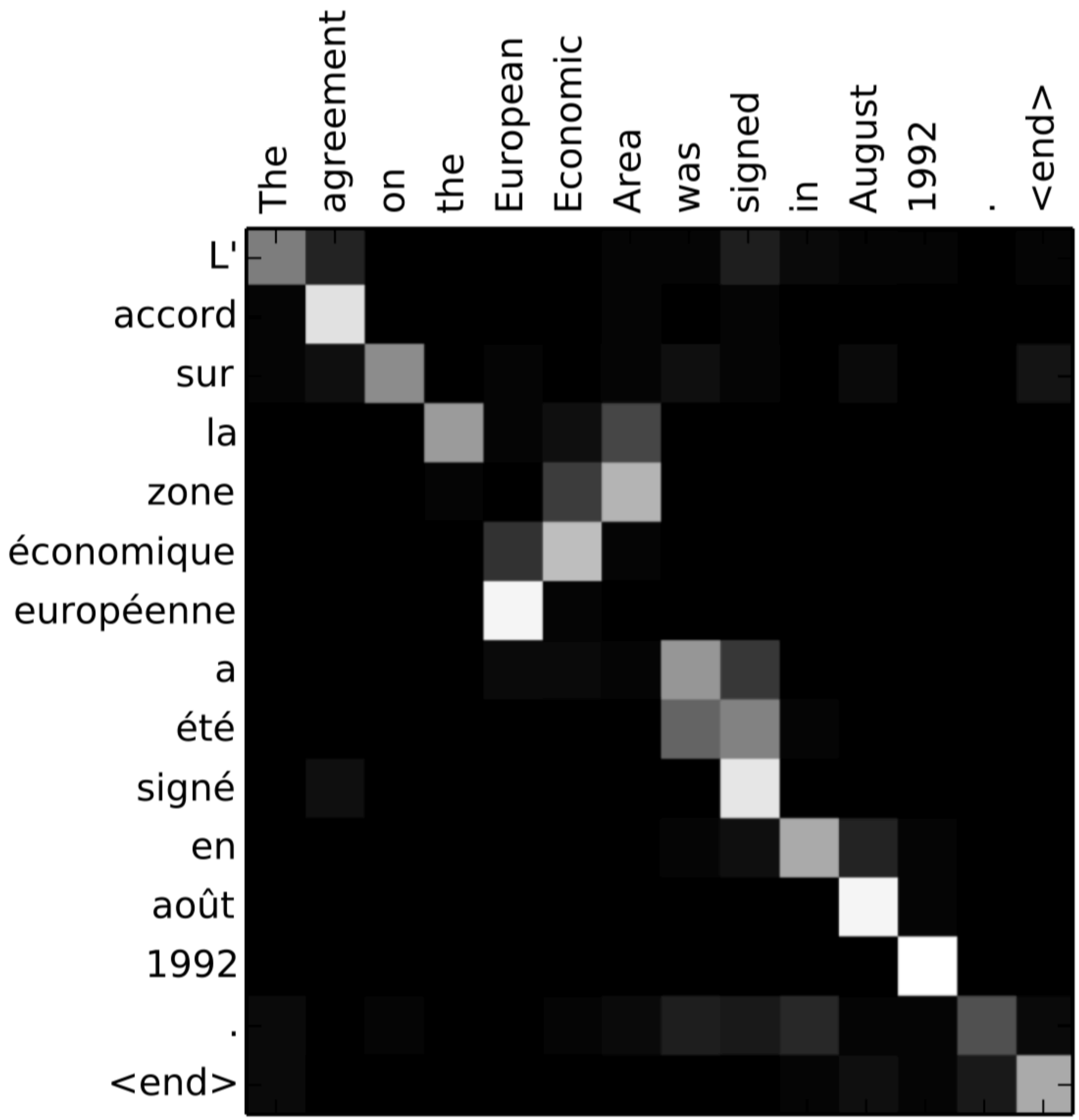


Image from Bahdanau et al. (2015)

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)

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

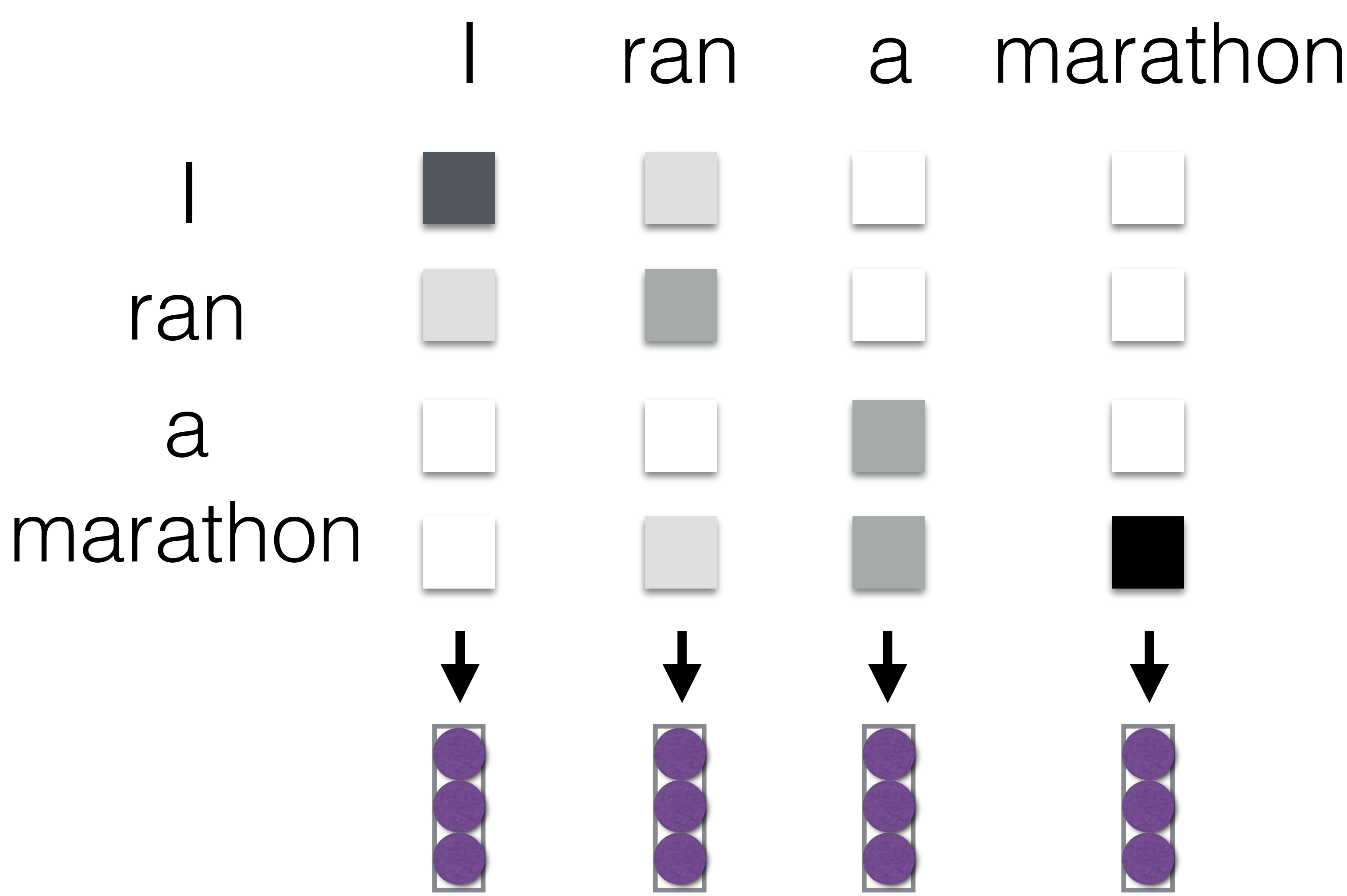
  - No parameters! But requires sizes to be the same.

- **Scaled Dot Product** (Vaswani et al. 2017)
  - *Problem:* scale of dot product increases as dimensions get larger
  - *Fix:* scale by size of the vector

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

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

- Each element in the sentence attends to other elements → context sensitive encodings!
  ran should have a different representation than in “I ran a company”



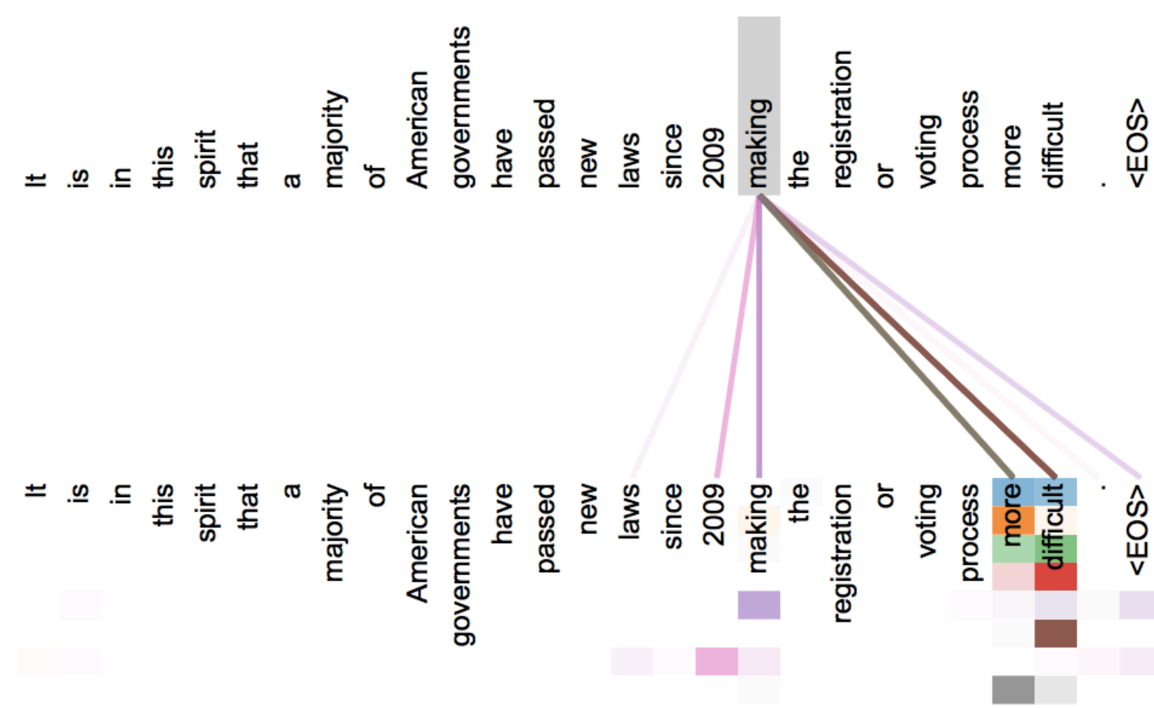
Multi-headed Attention

- **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)



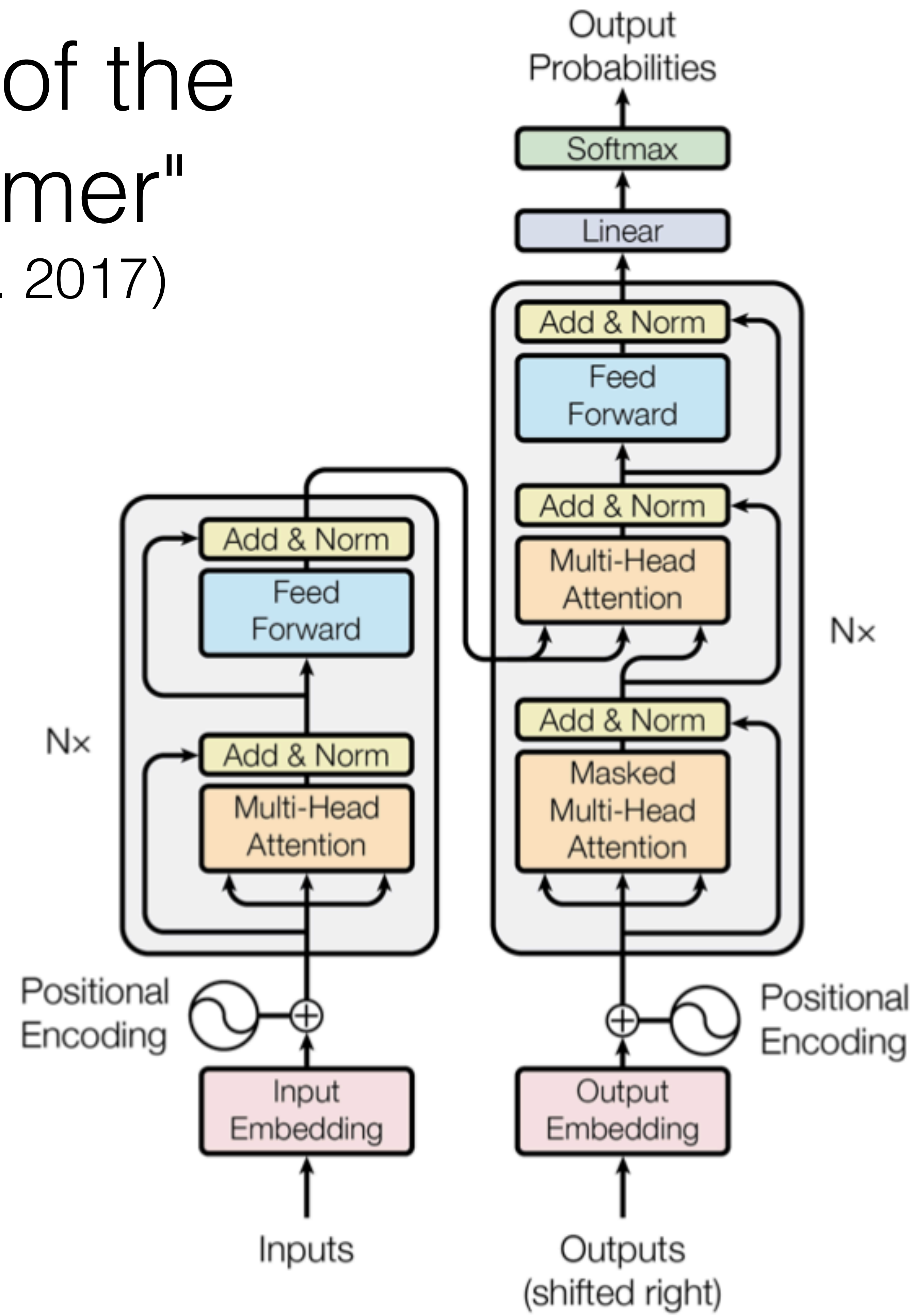
- Or one head for every hidden node! (Choi et al. 2018)

Transformers

See supplemental slides

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

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



Model Tricks

- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Multi-layer perceptron**: Transform attended vectors
- **Residual connections**: Train deep models; each layer computes a modification to the “residual stream”
- **Positional Encodings:** Make sure that even if we don’t have RNN, can still distinguish positions

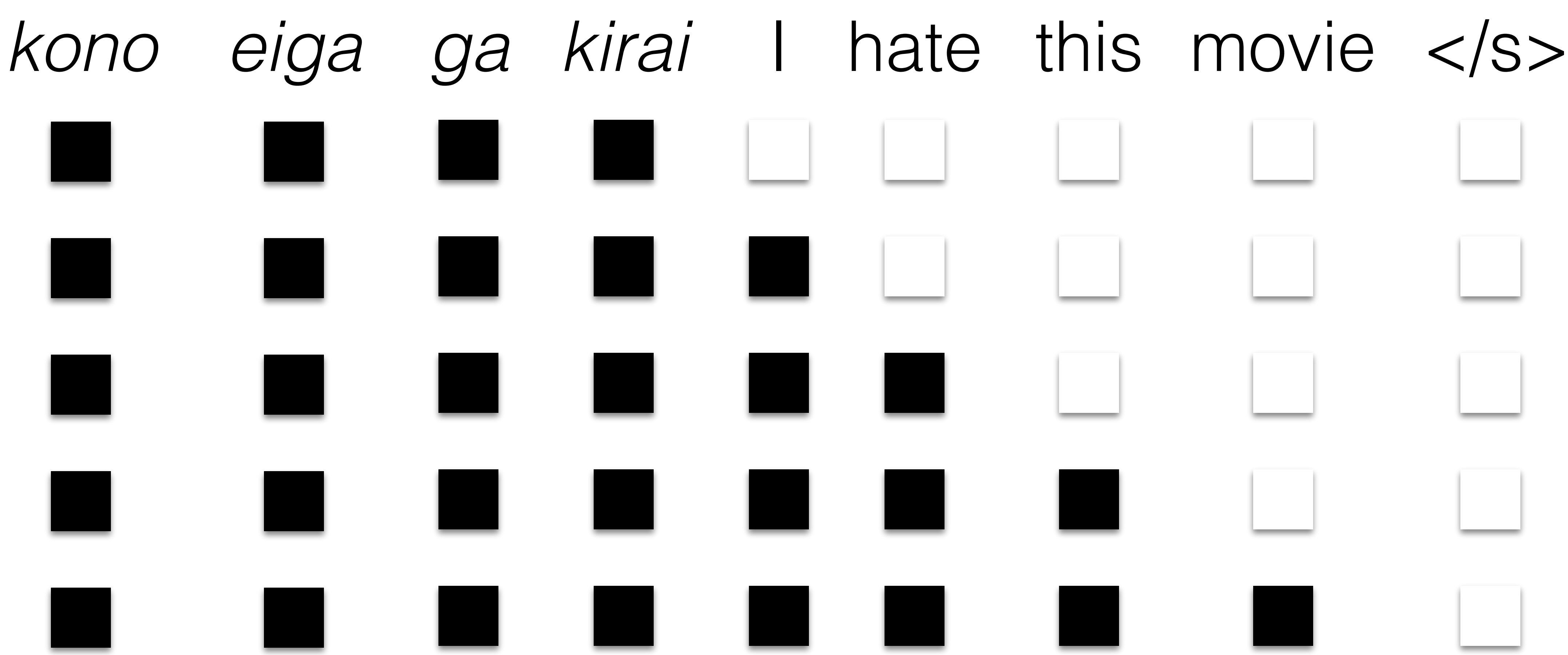
Training Tricks

- **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**

Masking for Training

- We want to perform training in as few operations as possible using big matrix multiplies

- We can do so by “masking” the results for the output



More Resources

- **The Annotated Transformer**: PyTorch implementation of Vaswani et al. 2017, interleaved with the original paper text. Helpful for Assignment 1! https://nlp.seas.harvard.edu/2018/04/03/attention.html

- **A Mathematical Framework for Transformer Circuits**: Build intuition for self-attention and simplified Transformer layers. https://transformer-circuits.pub/2021/framework/index.html

Extensions to Attention

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

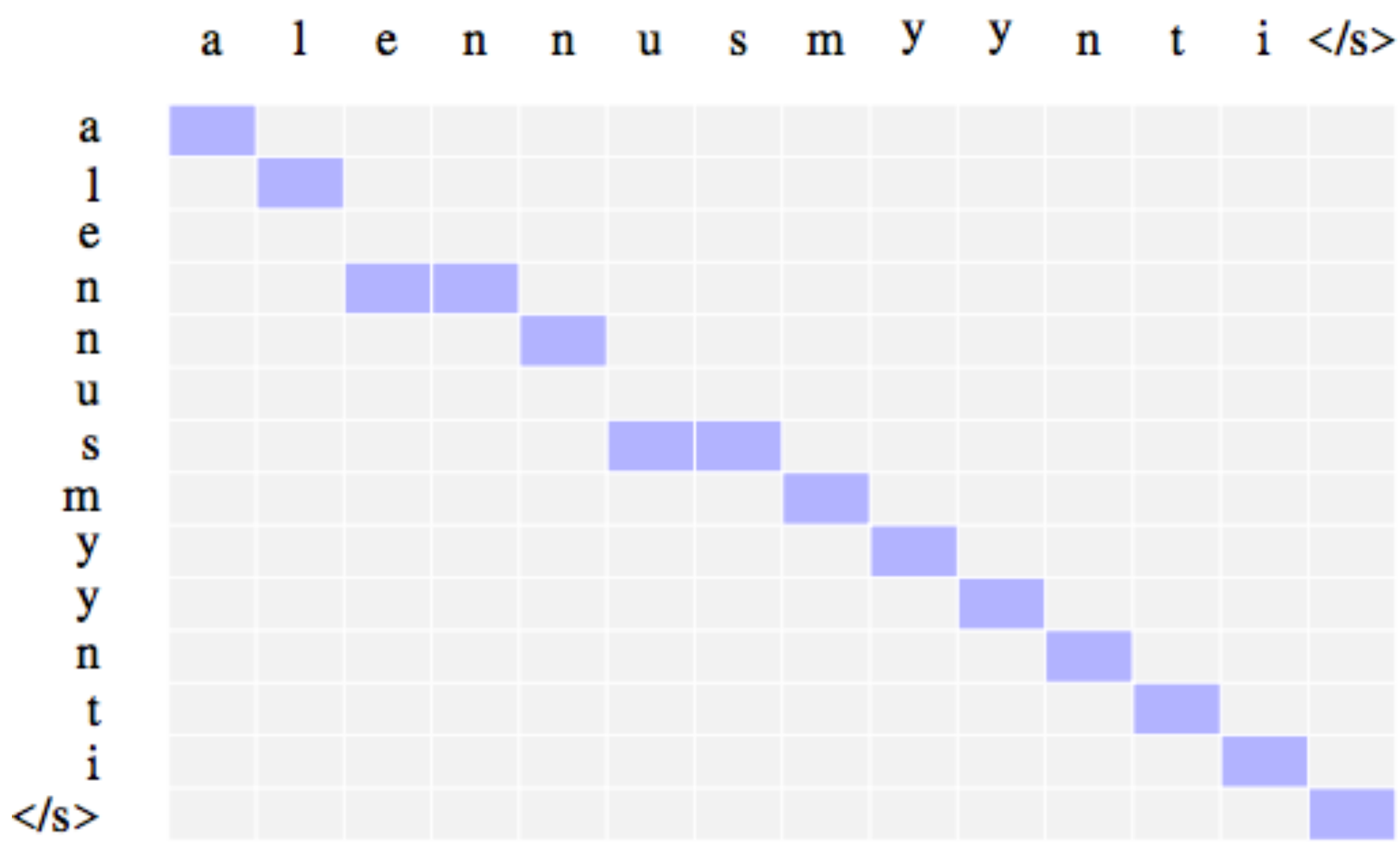
the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of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

Ratings

Look: 5 stars *Smell:* 4 stars

Monotonic Attention
(e.g. Graves et al. 2006, Yu et al. 2016)

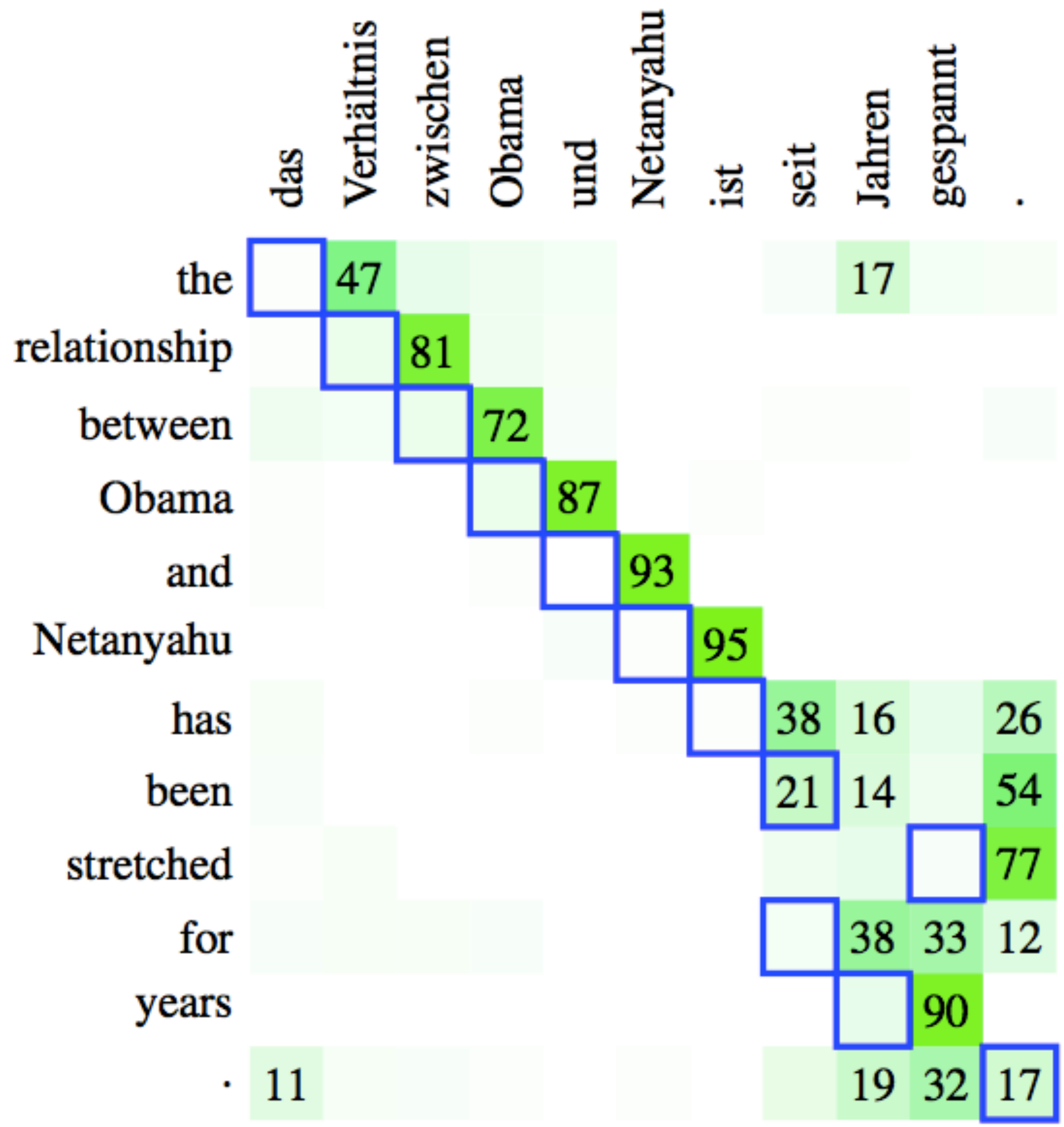
- In some cases, we might know the output will be in the same order as the input
    - Speech recognition, incremental translation, morphological inflection (?), summarization (?)



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

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)



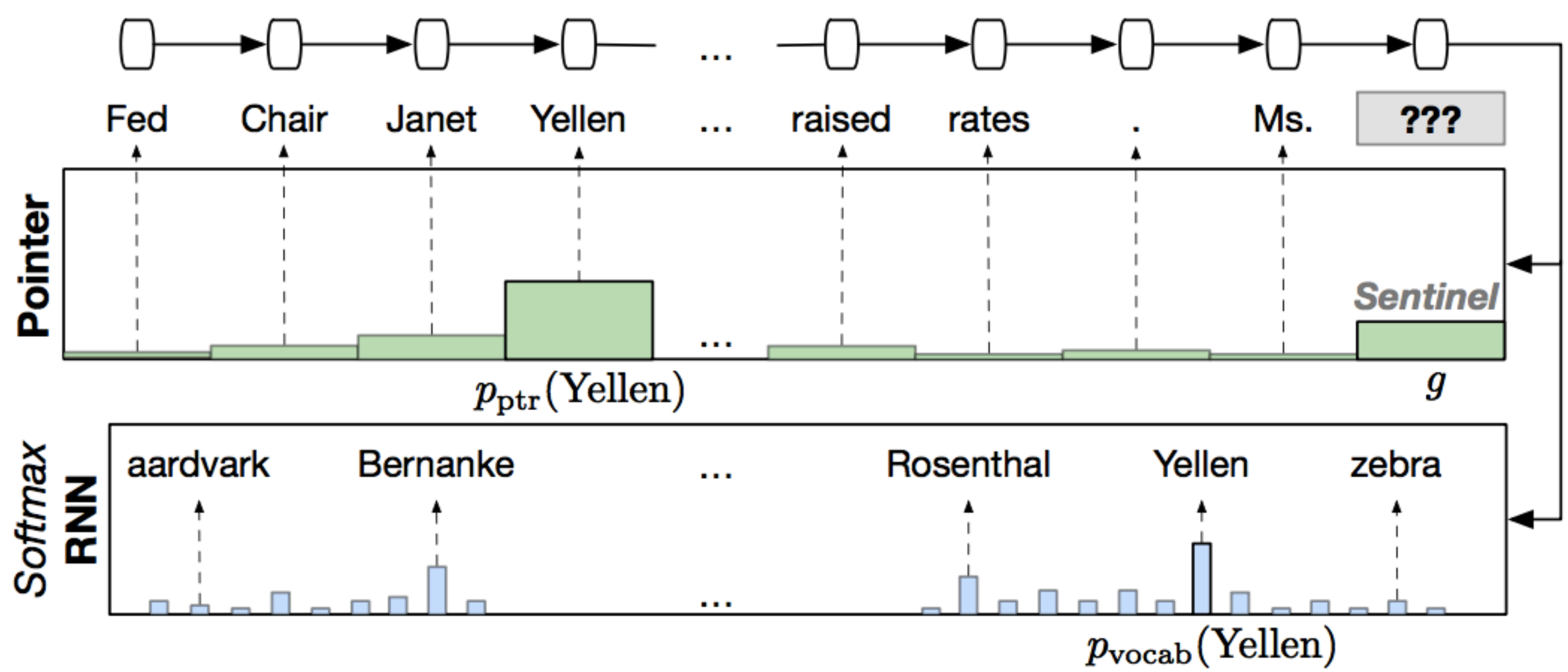
Supervised Training
(Mi et al. 2016)

- 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?

Copying from History

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



$$p(\text{Yellen}) = g\, p_{\text{vocab}}(\text{Yellen}) + (1-g)\, p_{\text{ptr}}(\text{Yellen})$$

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

Dictionary Probabilities

- If you have a translation dictionary, use it to bias outputs (Arthur et al. 2016)

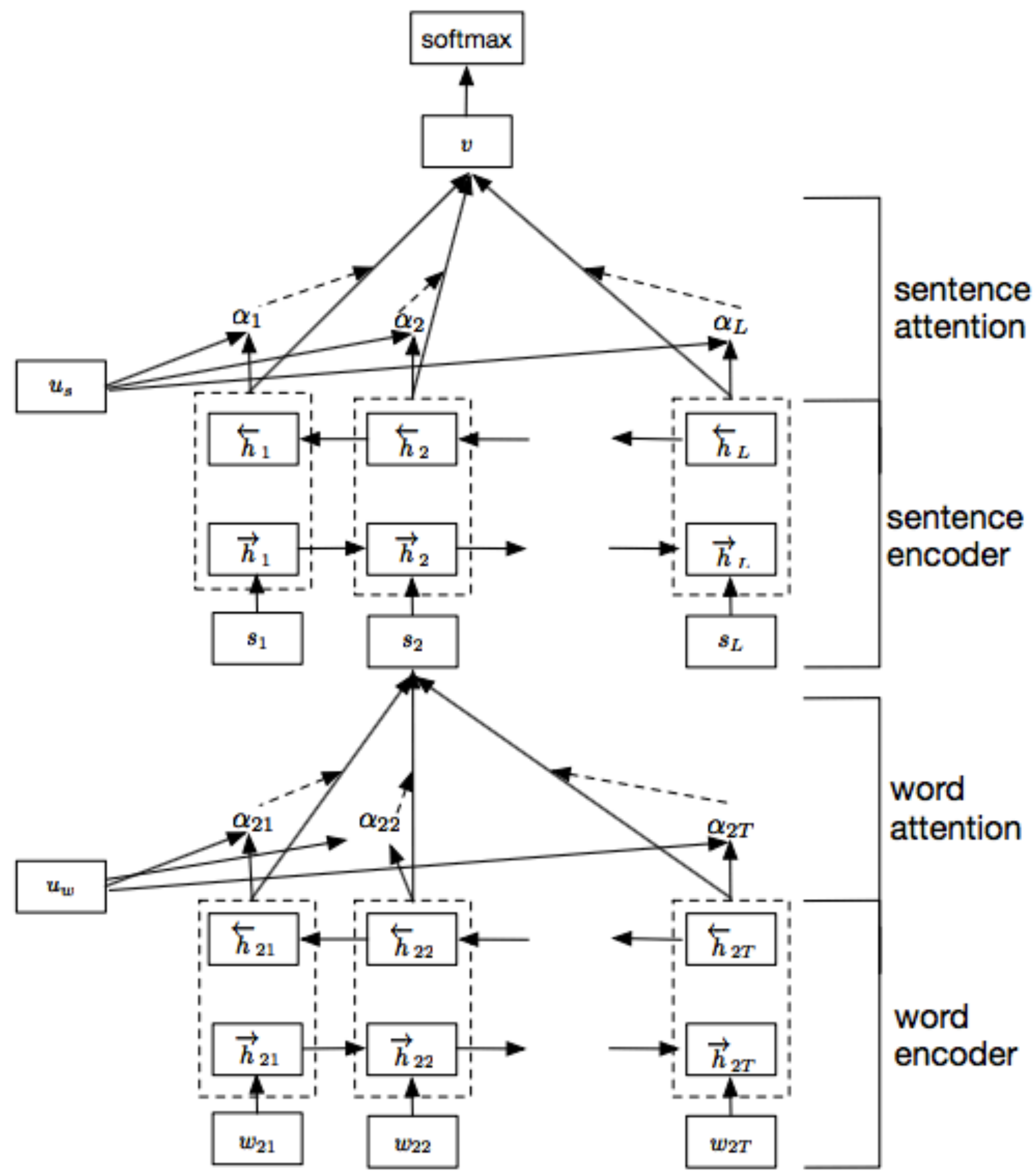
	I	come	from	Tunisia	
Attention	0.05	0.01	0.02	0.93	
watashi	0.6	0.03	0.01	0.0	0.03
ore	0.2	0.01	0.02	0.0	0.01
…	…	…	…	…	…
kuru	0.01	0.3	0.01	0.0	0.00
kara	0.02	0.1	0.5	0.01	0.02
…	…	…	…	…	…
chunijia	0.0	0.0	0.0	0.96	0.89
oranda	0.0	0.0	0.0	0.0	0.00

Sentence-level dictionary probability matrix

Dictionary probability for current word

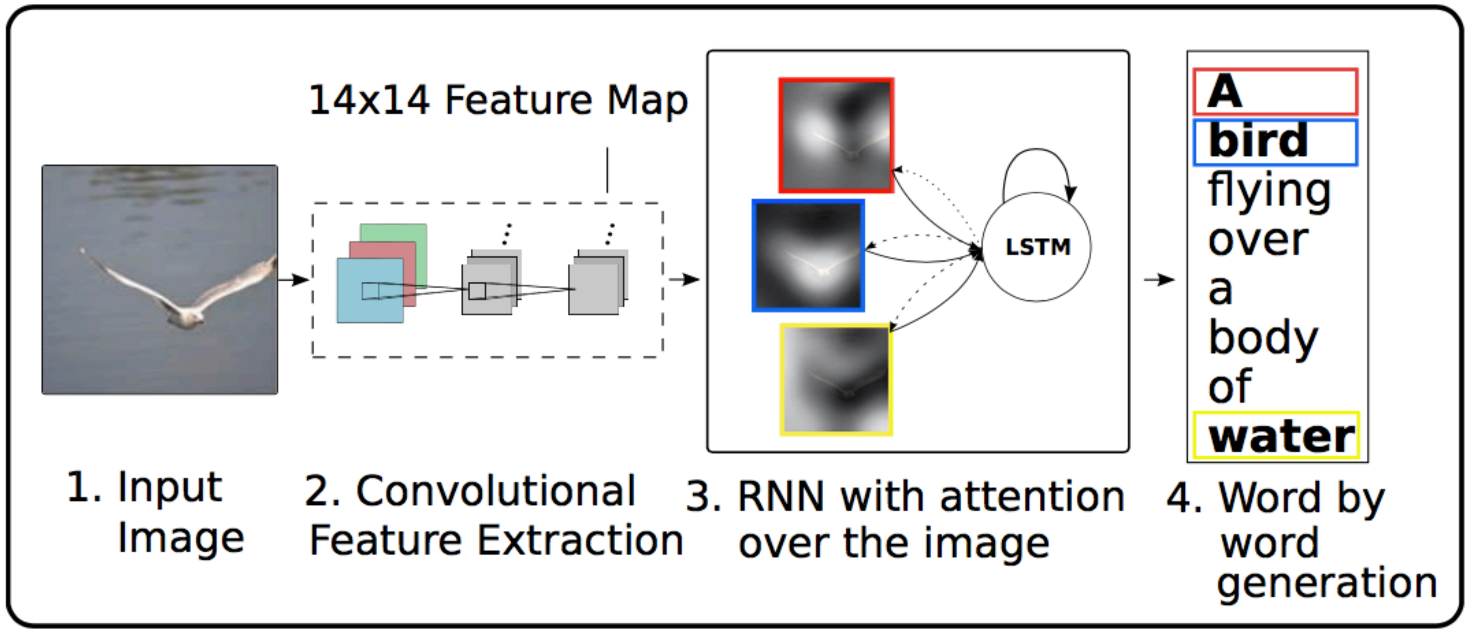
Hierarchical Structures
(Yang et al. 2016)

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

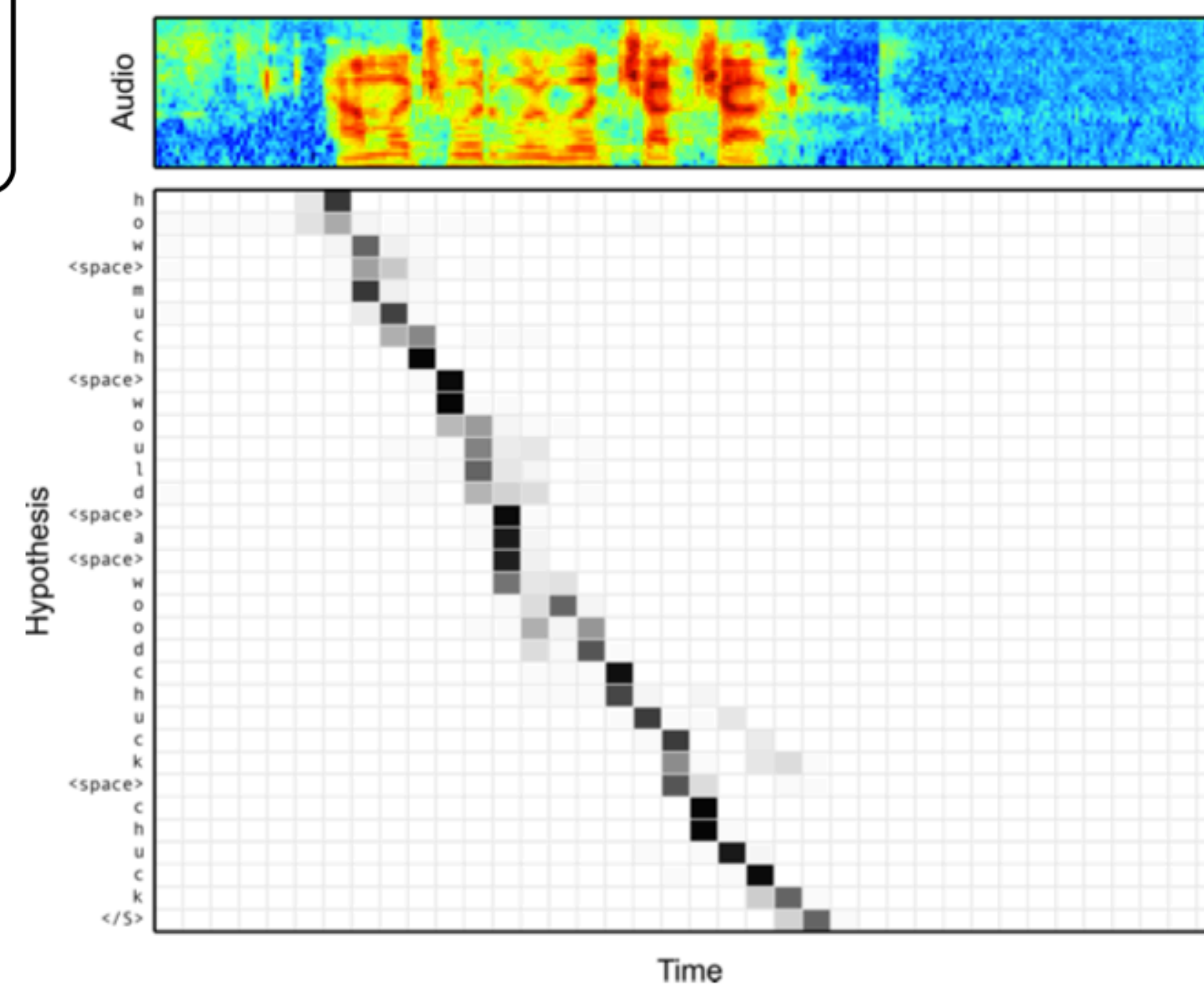


Various Modalities

- Images (Xu et al. 2015)

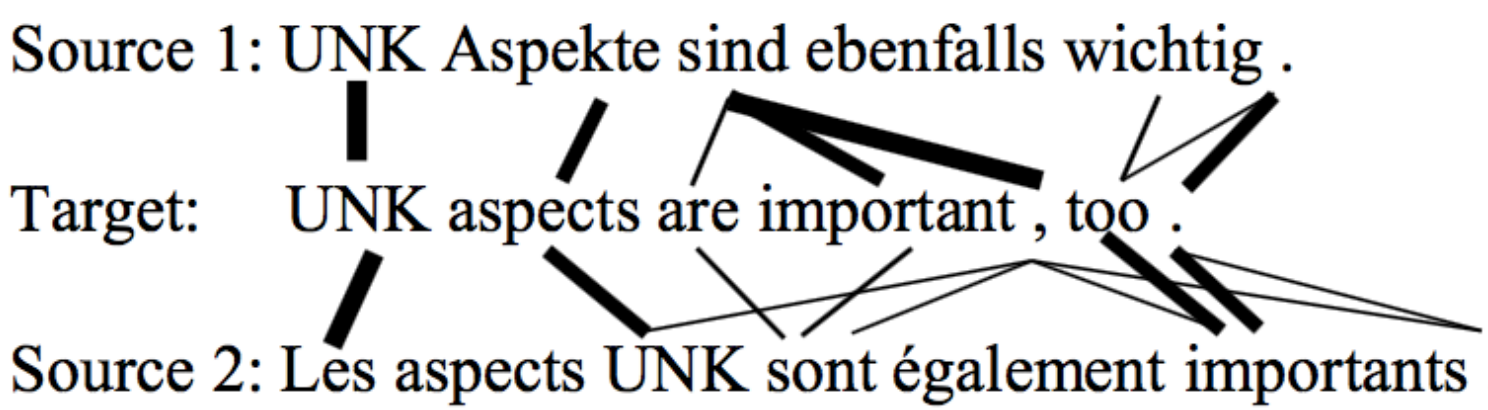


- Speech (Chan et al. 2015)



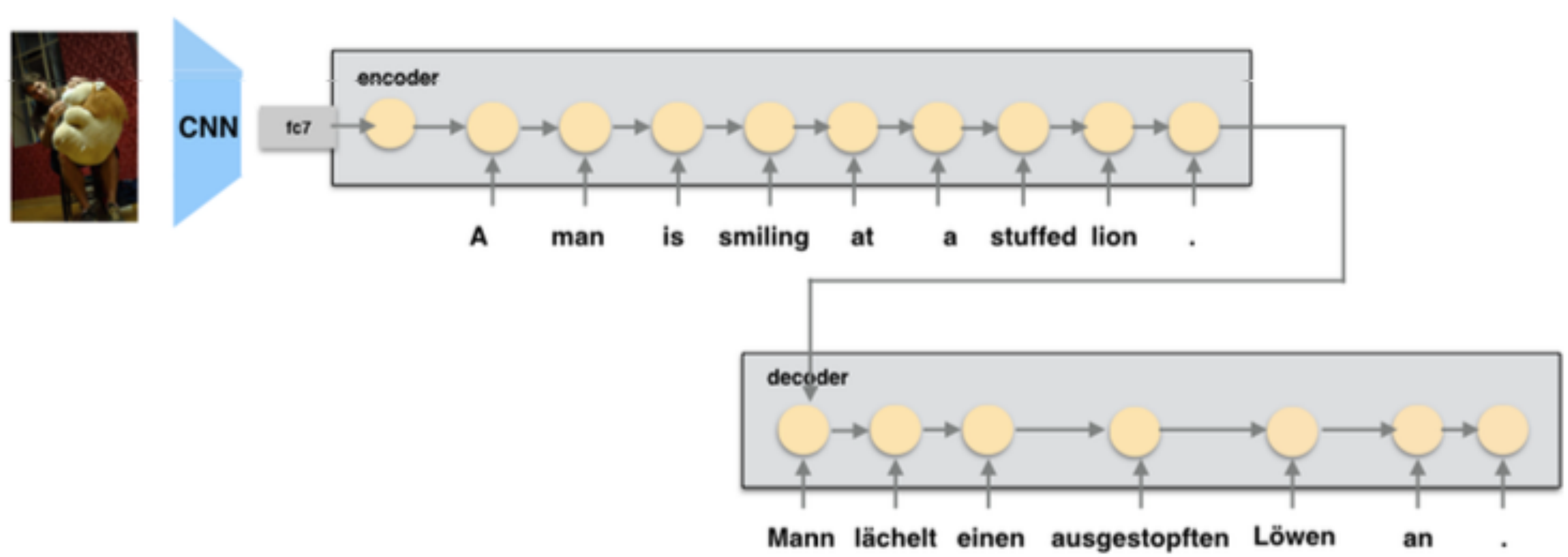
Multiple Sources

- 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?