CS11-747 Neural Networks for NLP

Attention

Graham Neubig

Carnegie Mellon University

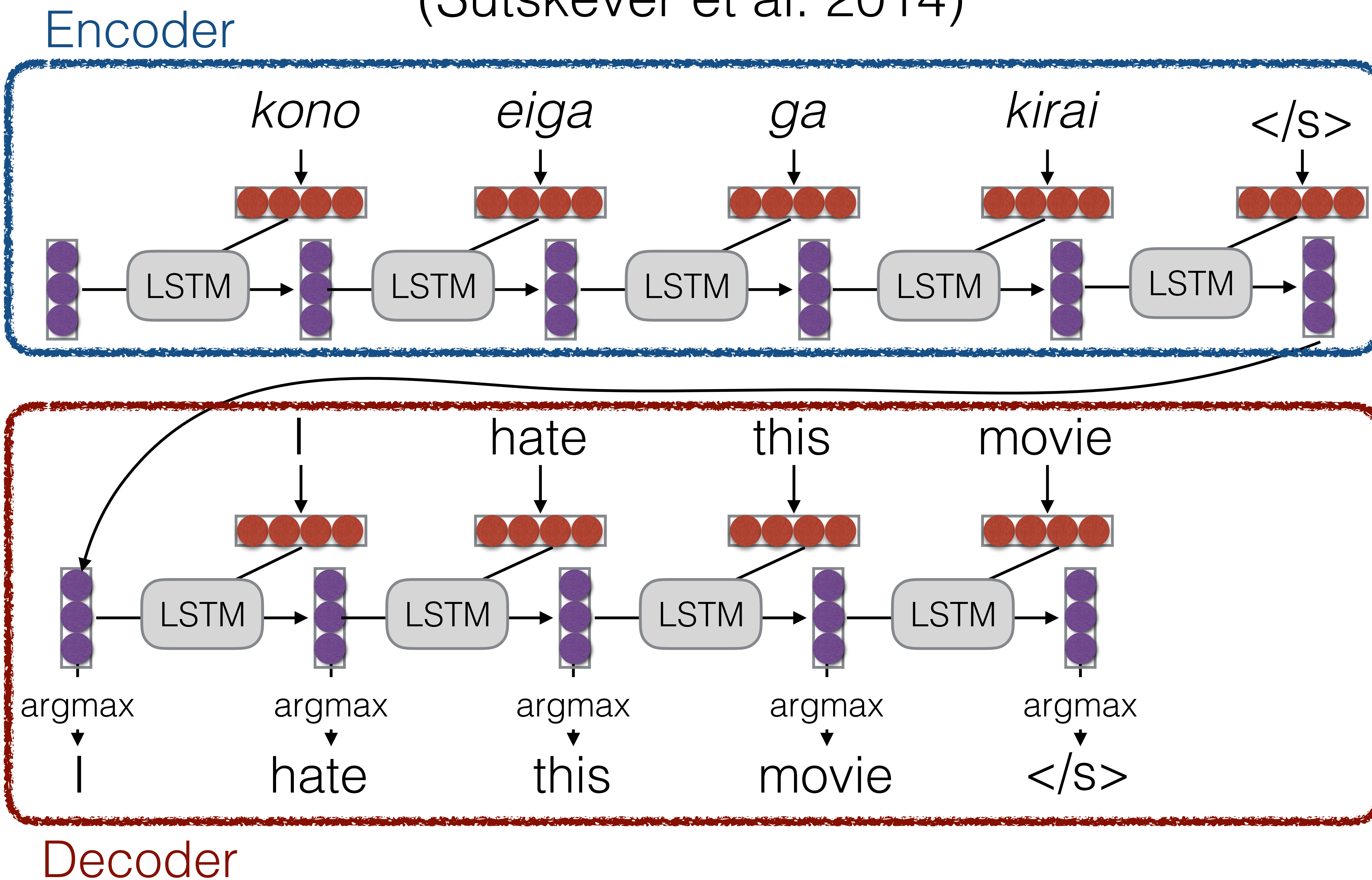
Language Technologies Institute

Site

https://phontron.com/class/nn4nlp2019/

Encoder-decoder Models

(Sutskever et al. 2014)



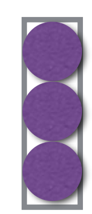
Sentence Representations

<u>Problem!</u>

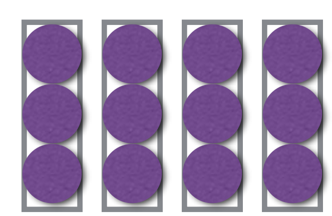
> "You can't cram the meaning of a whole %&!$ing sentence into a single $&!*ing vector!"
> — Ray Mooney

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

this is an example →



this is an example →



Attention

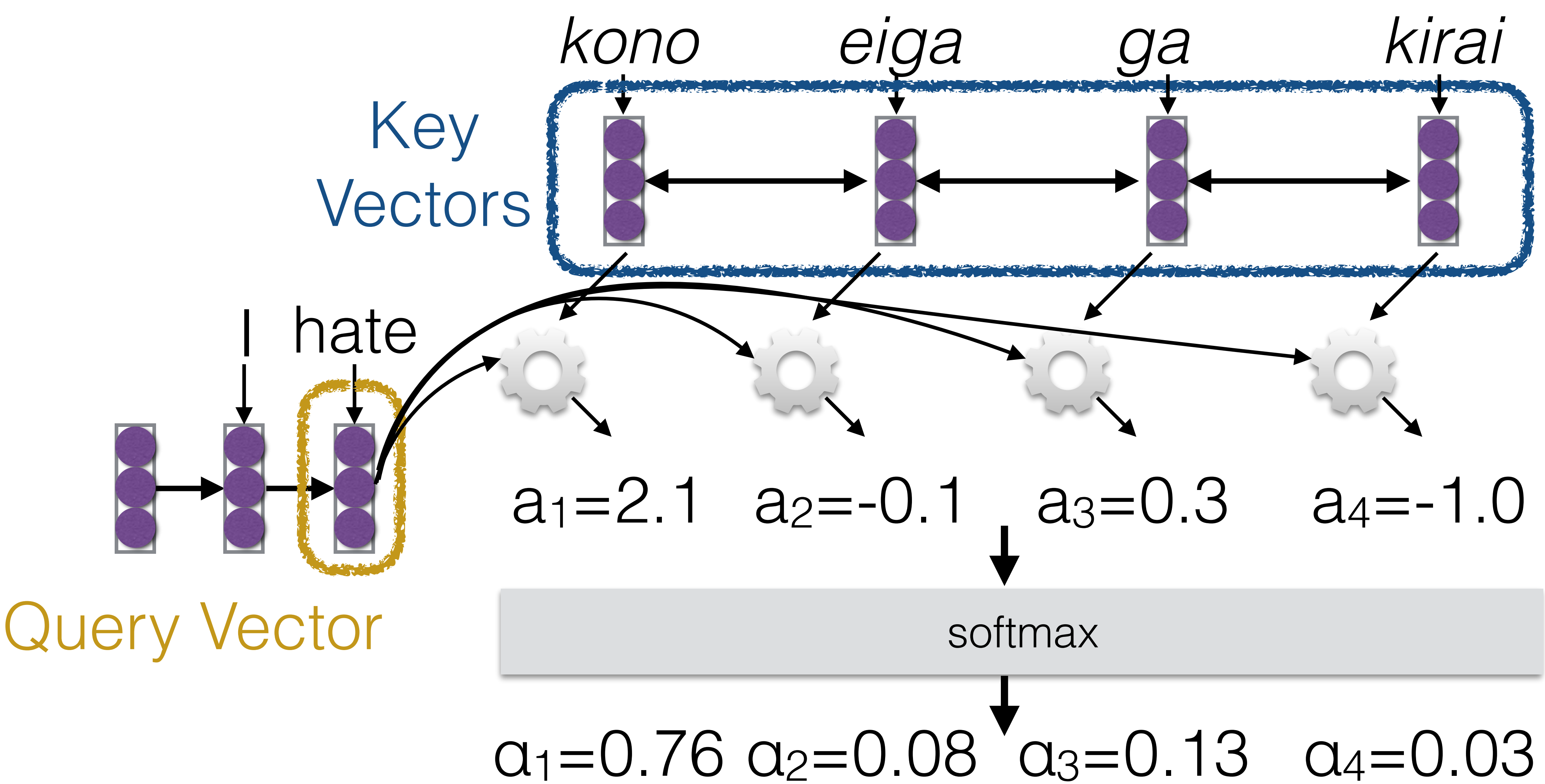
Basic Idea

(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

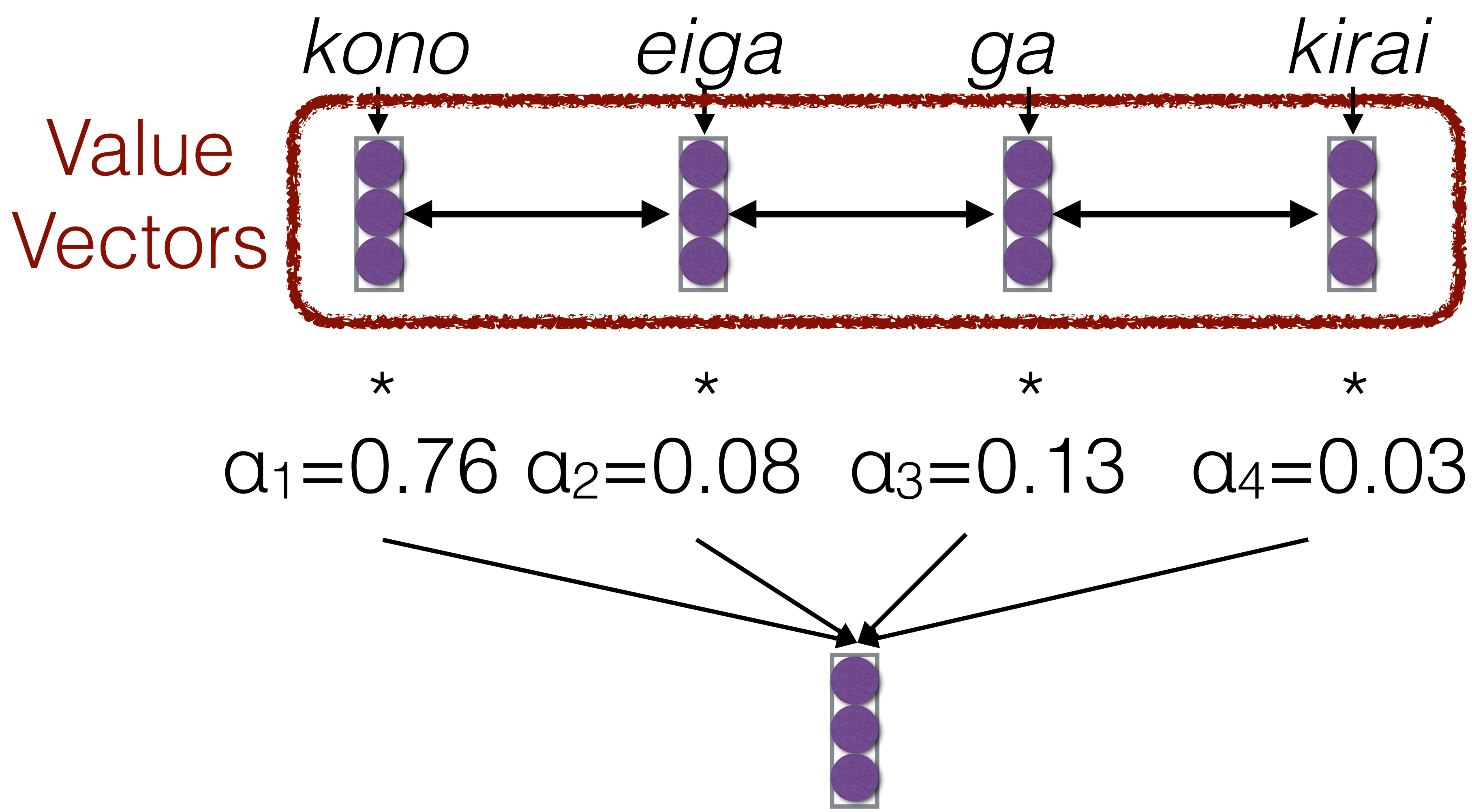
Calculating Attention (1)

- 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



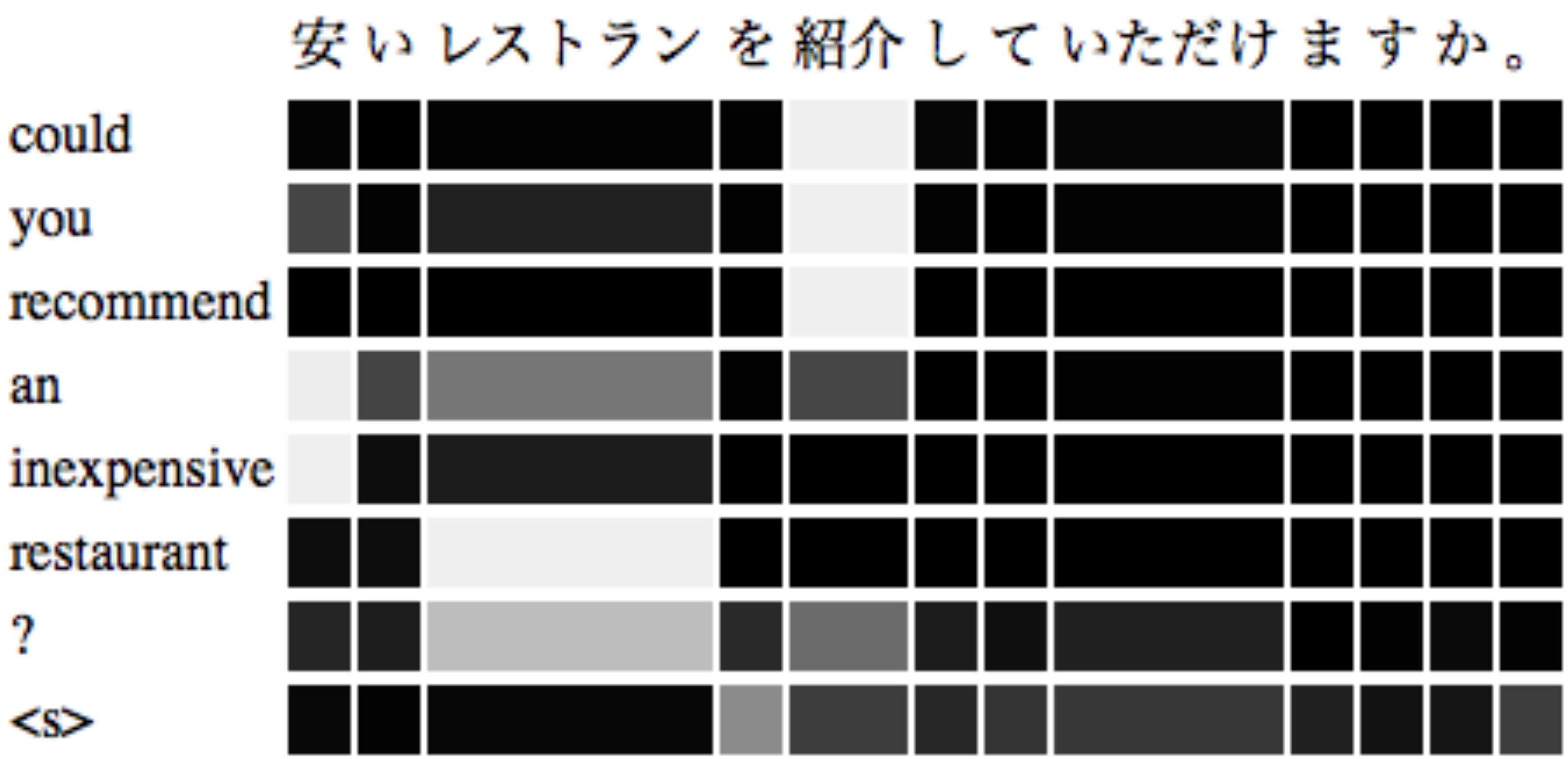
Calculating Attention (2)

- 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



Attention Score Functions (1)

- ***q*** is the query and ***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

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

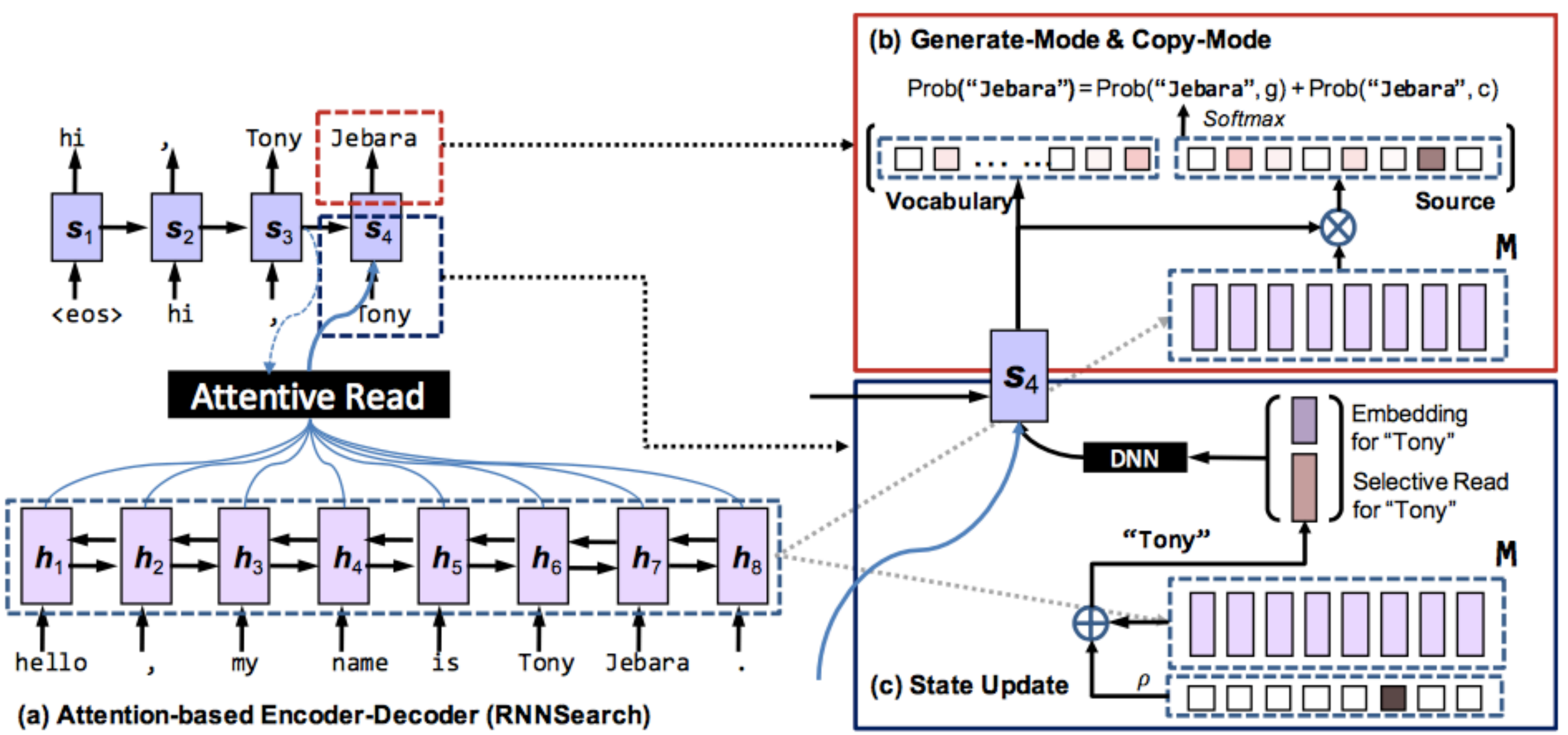
Let's Try it Out!

`attention.py`

What do we Attend To?

Input Sentence

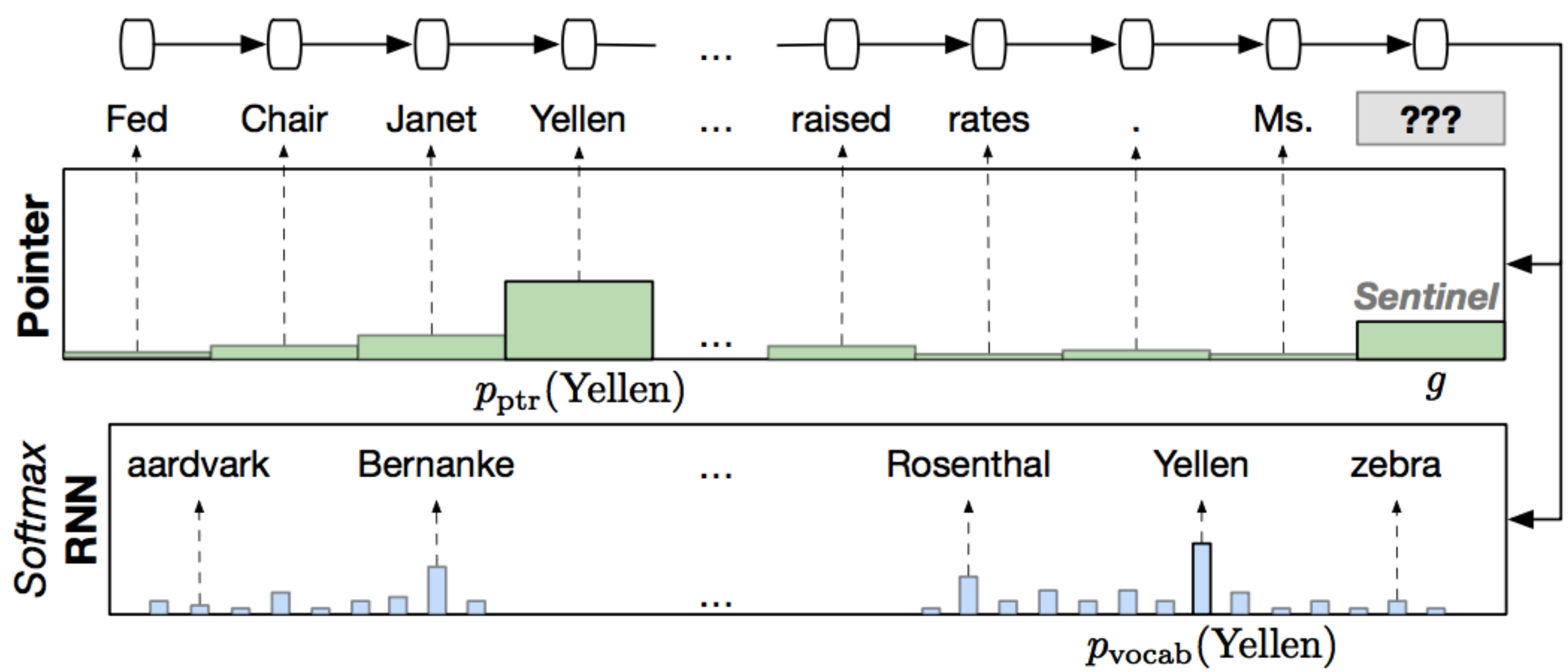
- Like the previous explanation
- But also, more directly
  - **Copying mechanism** (Gu et al. 2016)



  - **Lexicon bias** (Arthur et al. 2016)

Previously Generated Things

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

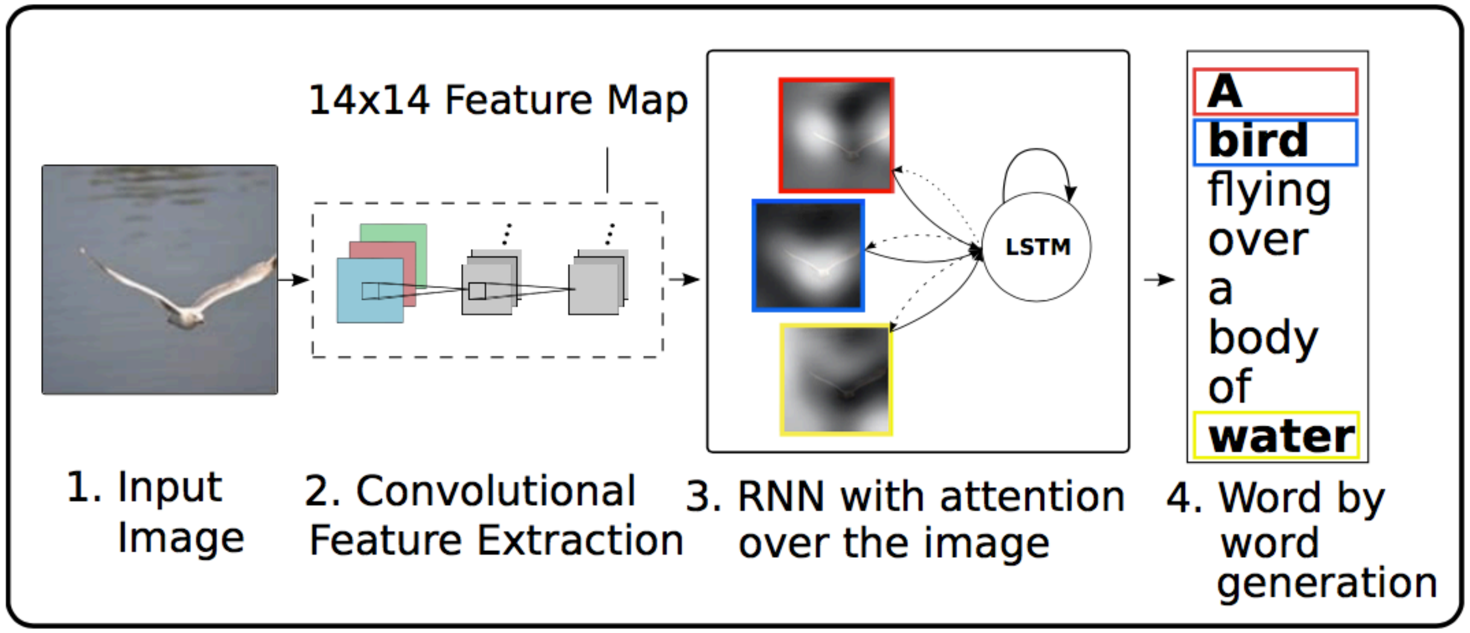


$$p(\mathrm{Yellen}) = g\ p_{\mathrm{vocab}}(\mathrm{Yellen}) + (1 - g)\ p_{\mathrm{ptr}}(\mathrm{Yellen})$$

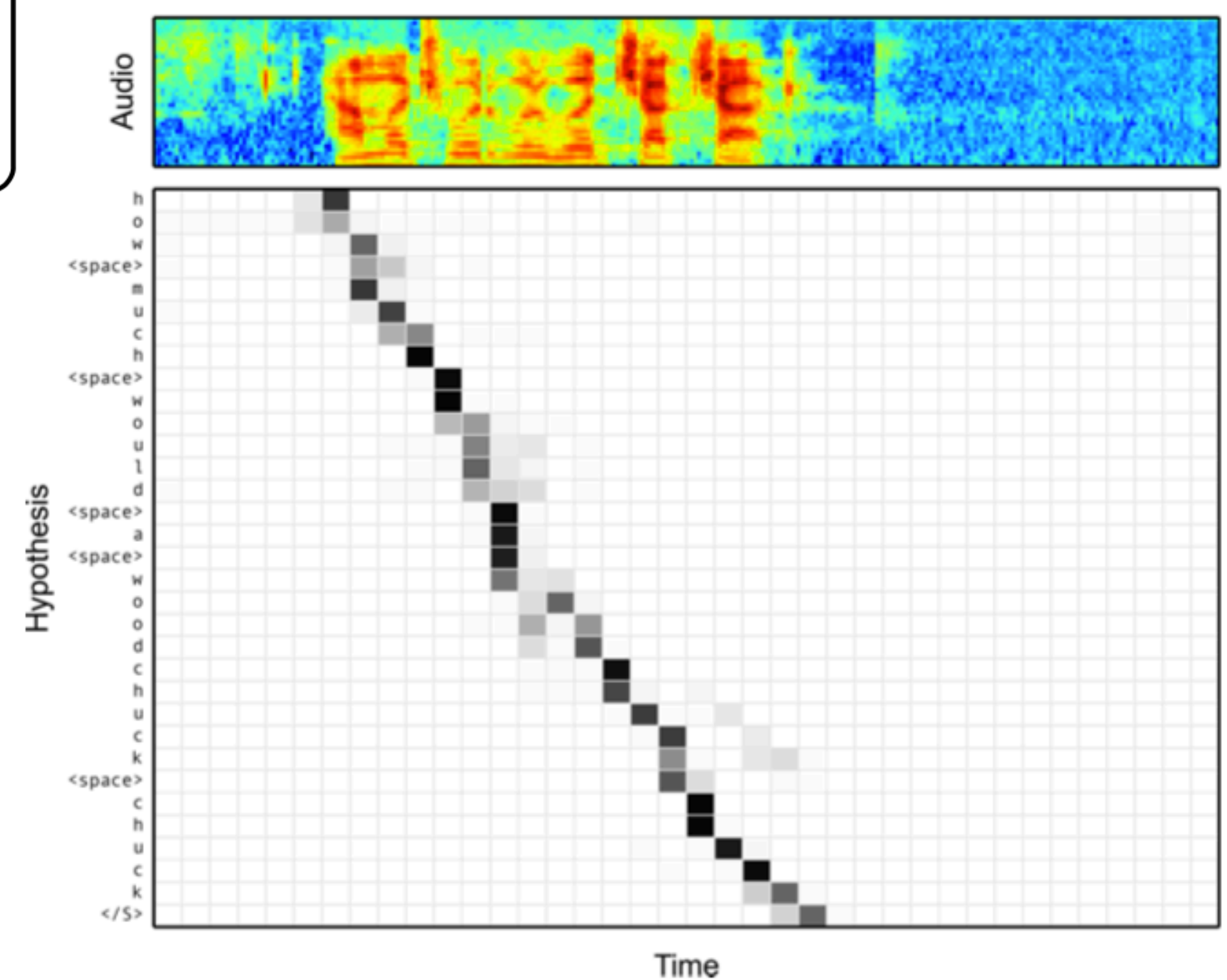
- In translation, attend to either input or previous output (Vaswani et al. 2017)

Various Modalities

- Images (Xu et al. 2015)

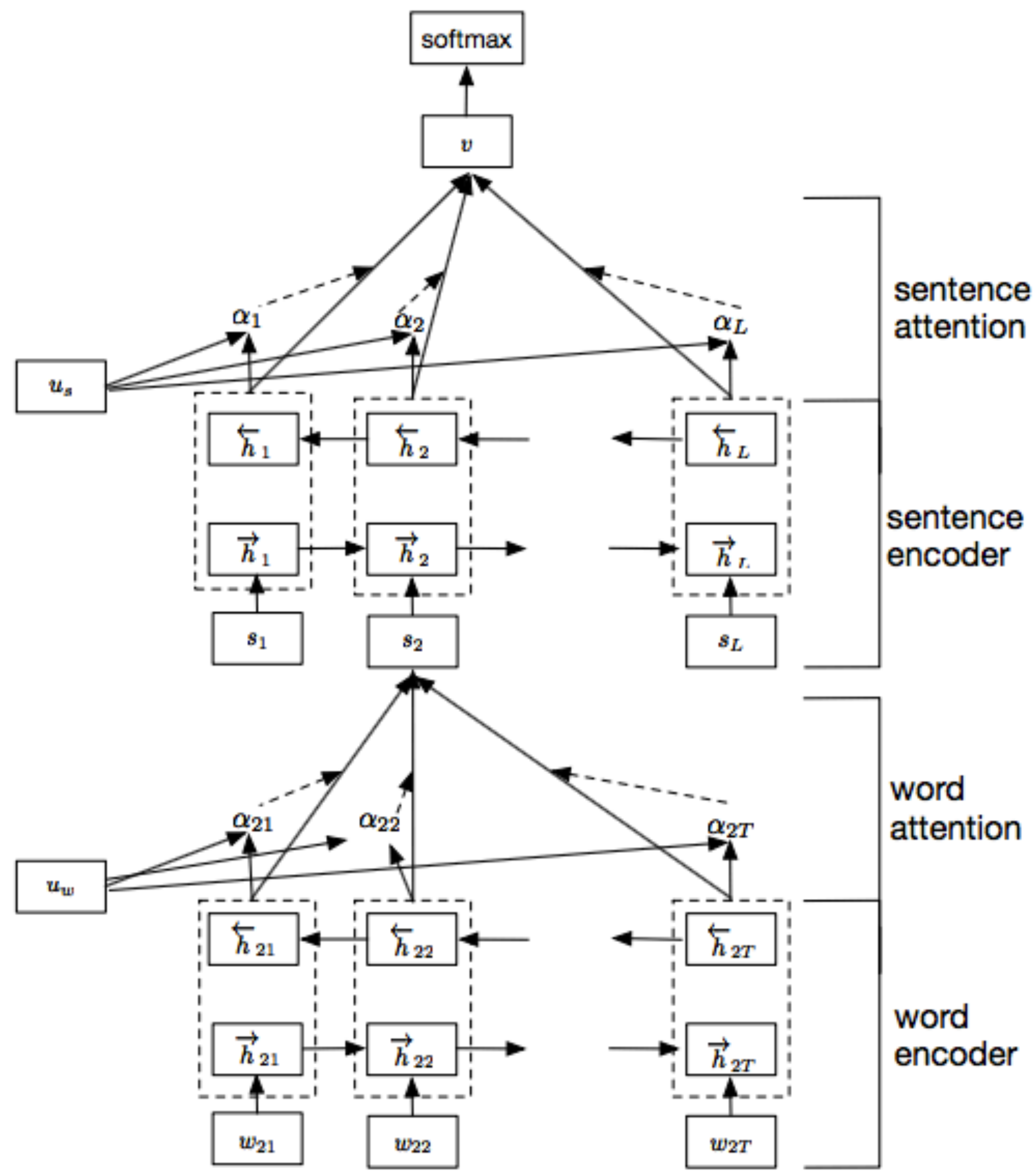


- Speech (Chan et al. 2015)



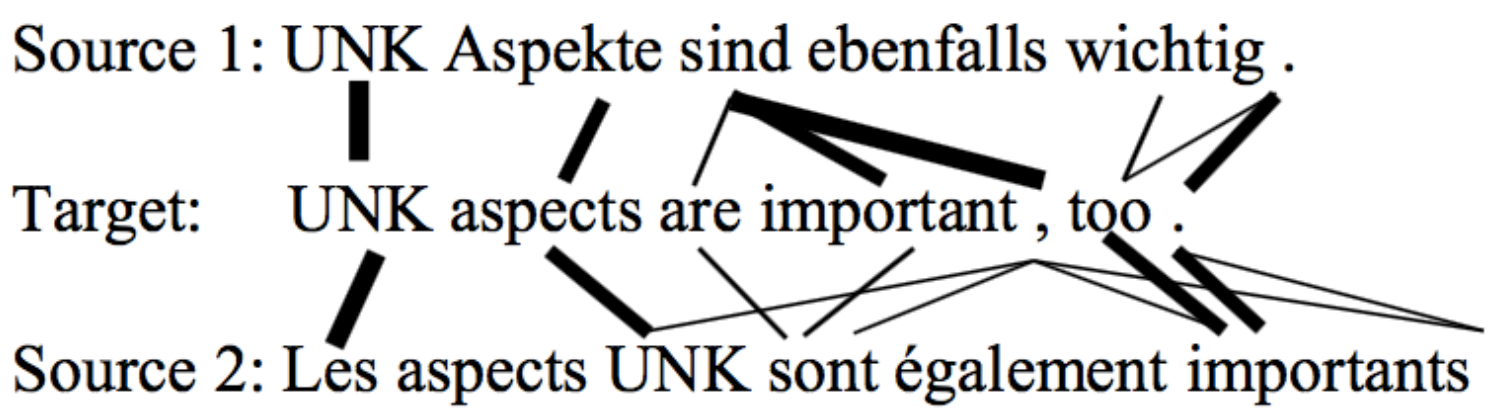
Hierarchical Structures
(Yang et al. 2016)

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



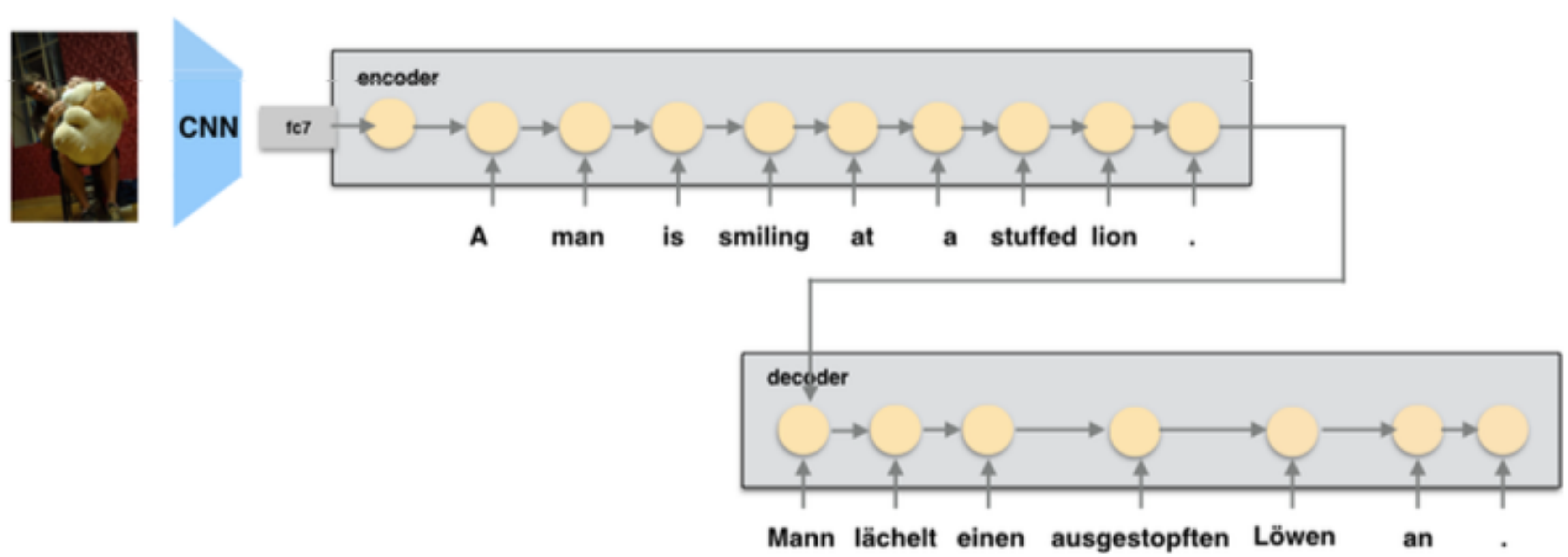
Multiple Sources

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



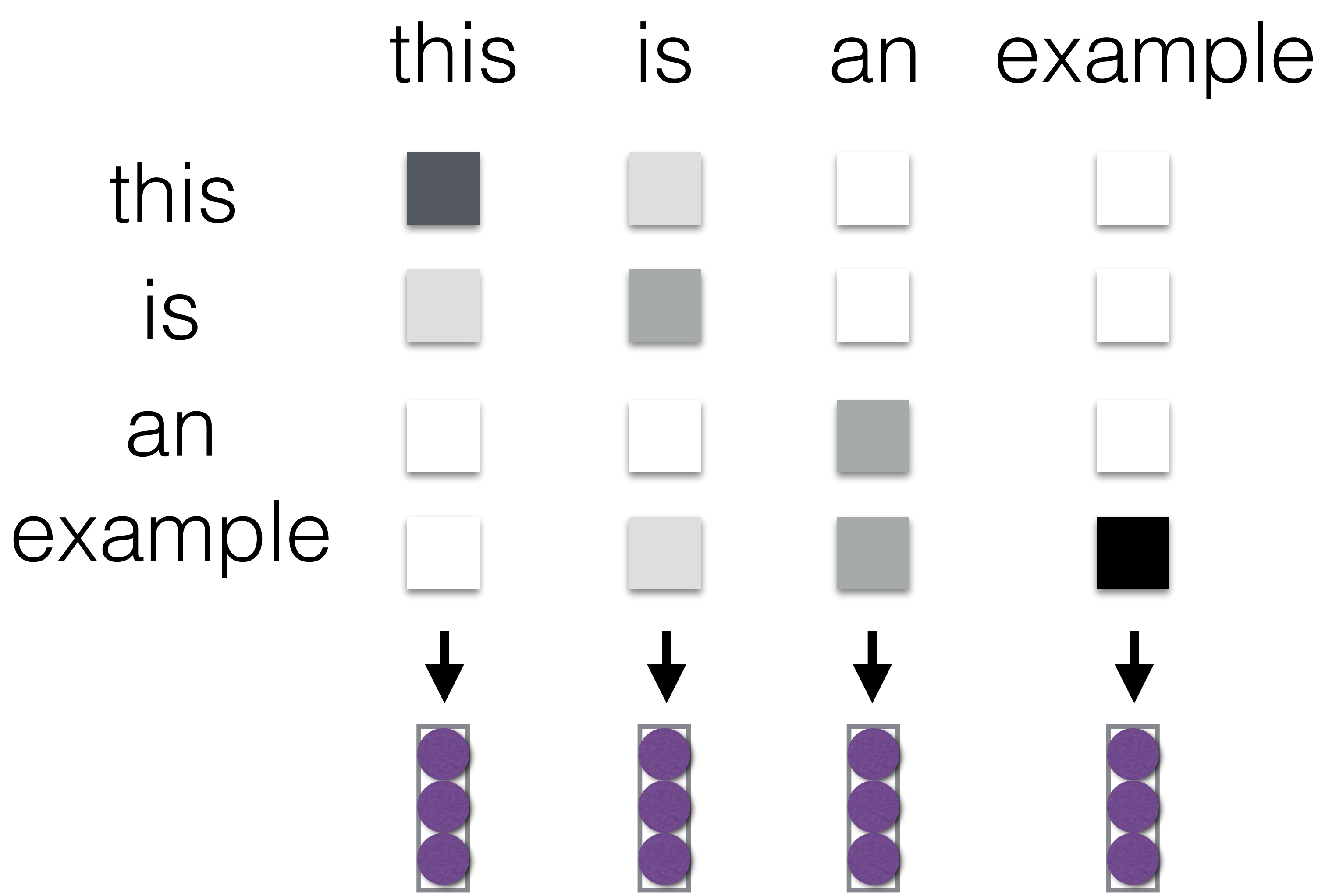
  - Libovicky and Helcl (2017) compare multiple strategies

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



Intra-Attention / Self Attention
(Cheng et al. 2016)

- Each element in the sentence attends to other elements → context sensitive encodings!



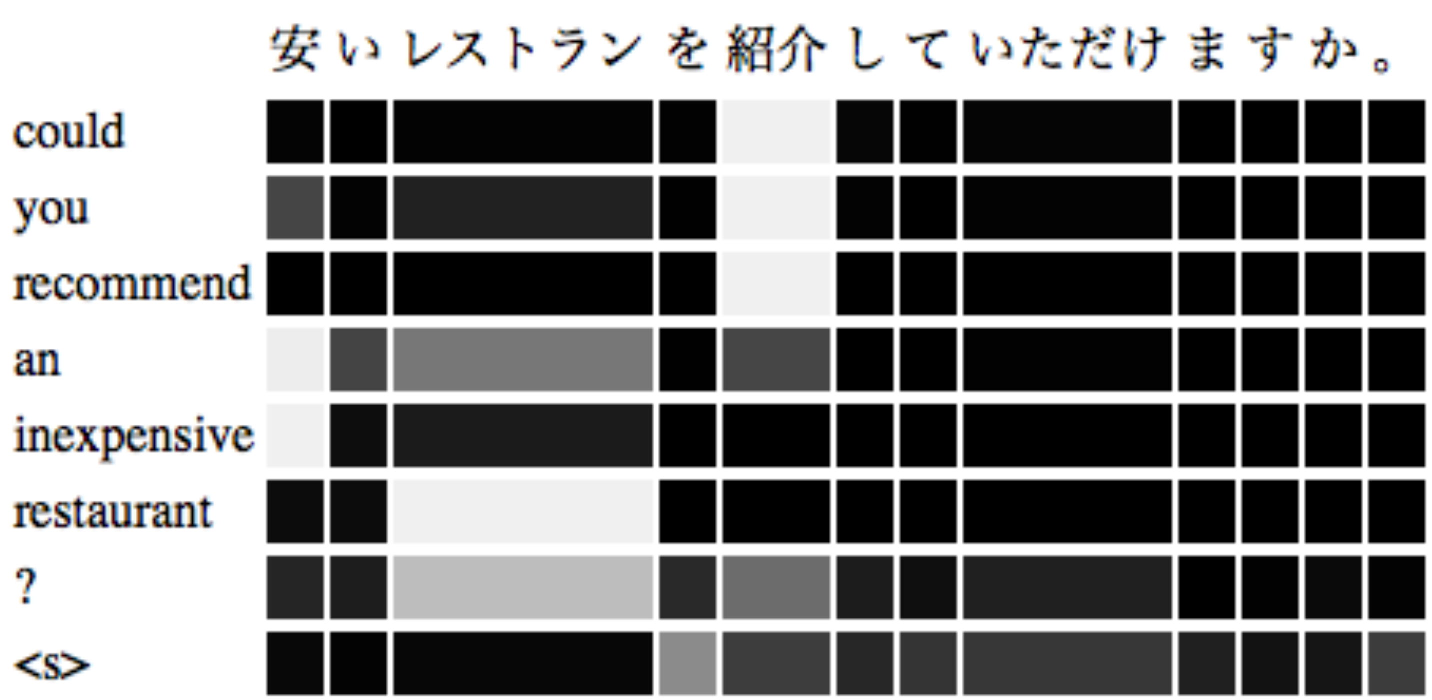
Improvements to Attention

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 (Cohn et al. 2015)
    - Add embeddings indicating coverage (Mi et al. 2016)

Incorporating Markov Properties
(Cohn et al. 2015)

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

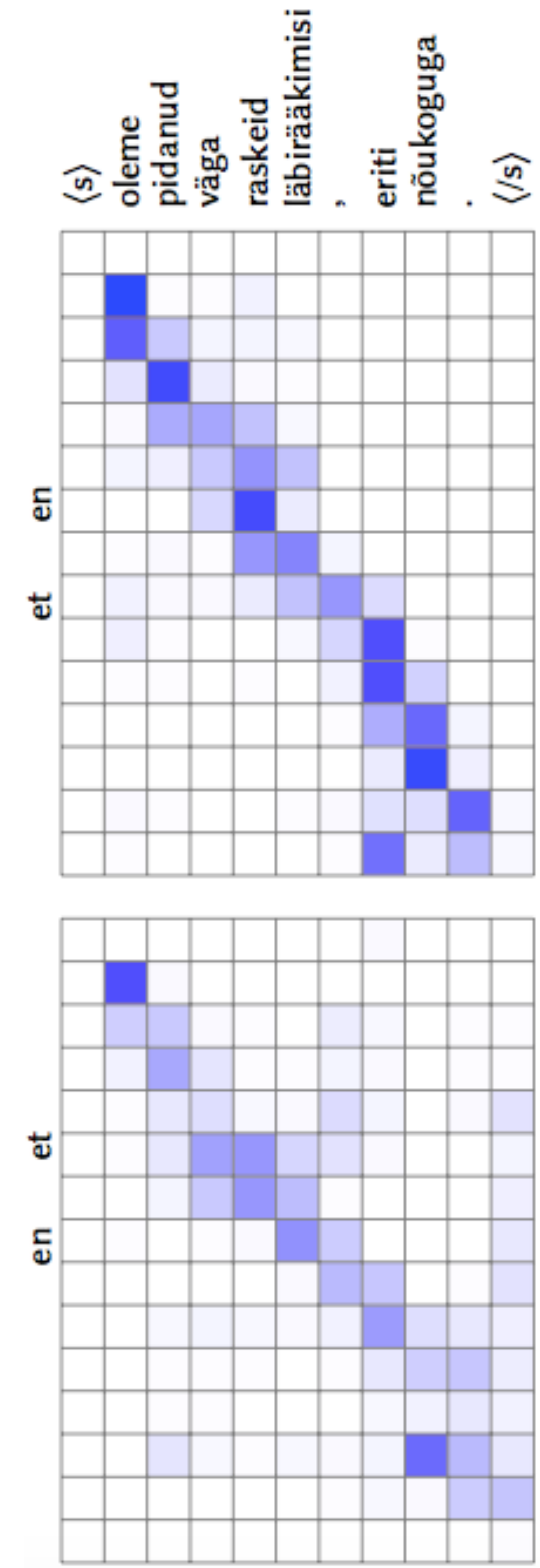


- Add information about the last attention when making the next decision

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

$$\mathrm{tr}(A_{X \to Y} A^{\intercal}_{Y \to X})$$



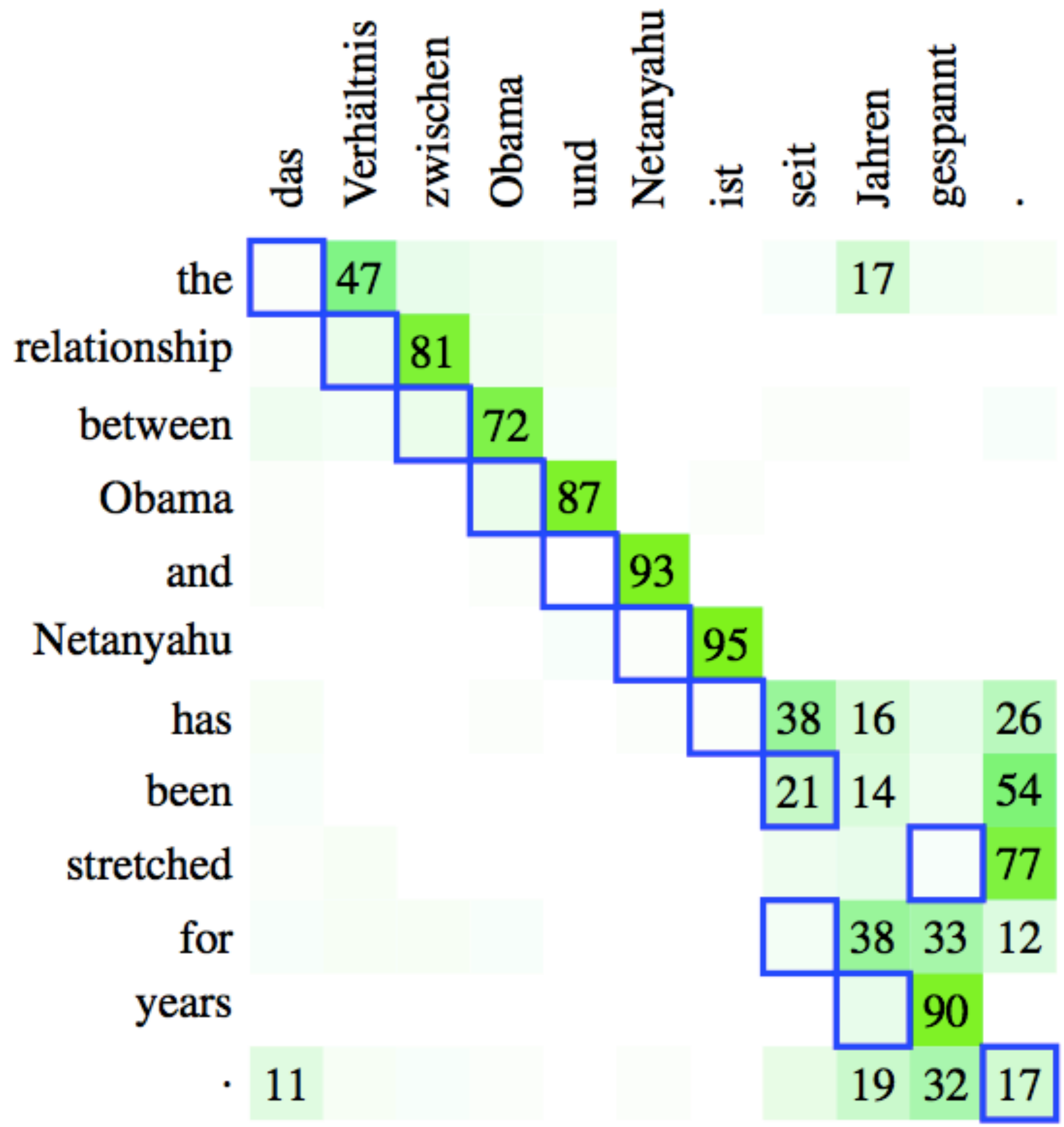
Supervised Training
(Mi et al. 2016)

- 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

Attention is not Alignment!

(Koehn and Knowles 2017)

- Attention is often blurred
- Attention is often off by one



Specialized Attention Varieties

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

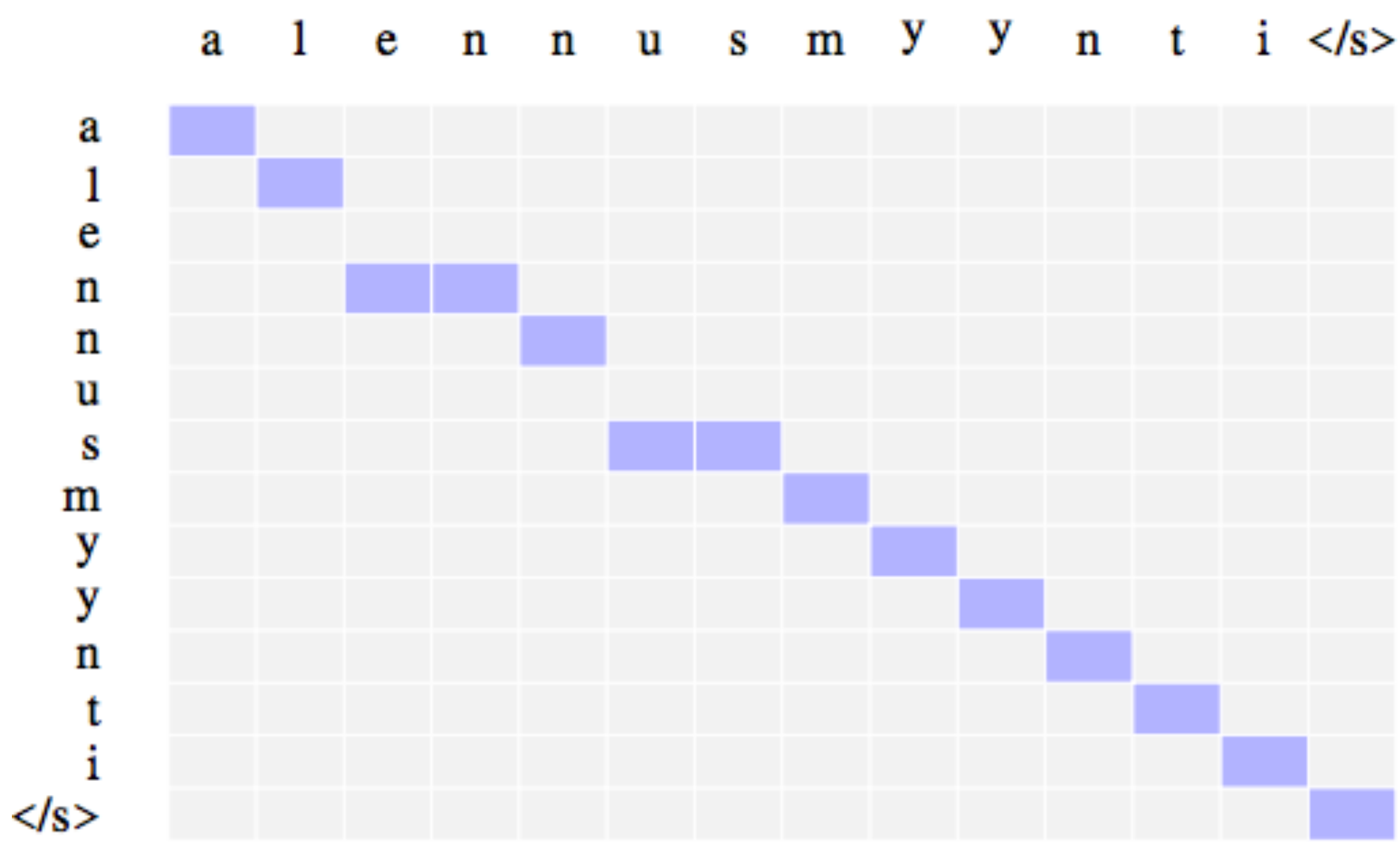
the beer was n't what i expected, and i'm not sure it's "true to style", but i thought it was delicious. **a very pleasant ruby red-amber color** with a relatively brilliant finish, but a limited amount of carbonation, from the look of it. aroma is what i think an amber ale should be - a nice blend of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

Ratings

Look: 5 stars   *Smell:* 4 stars

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

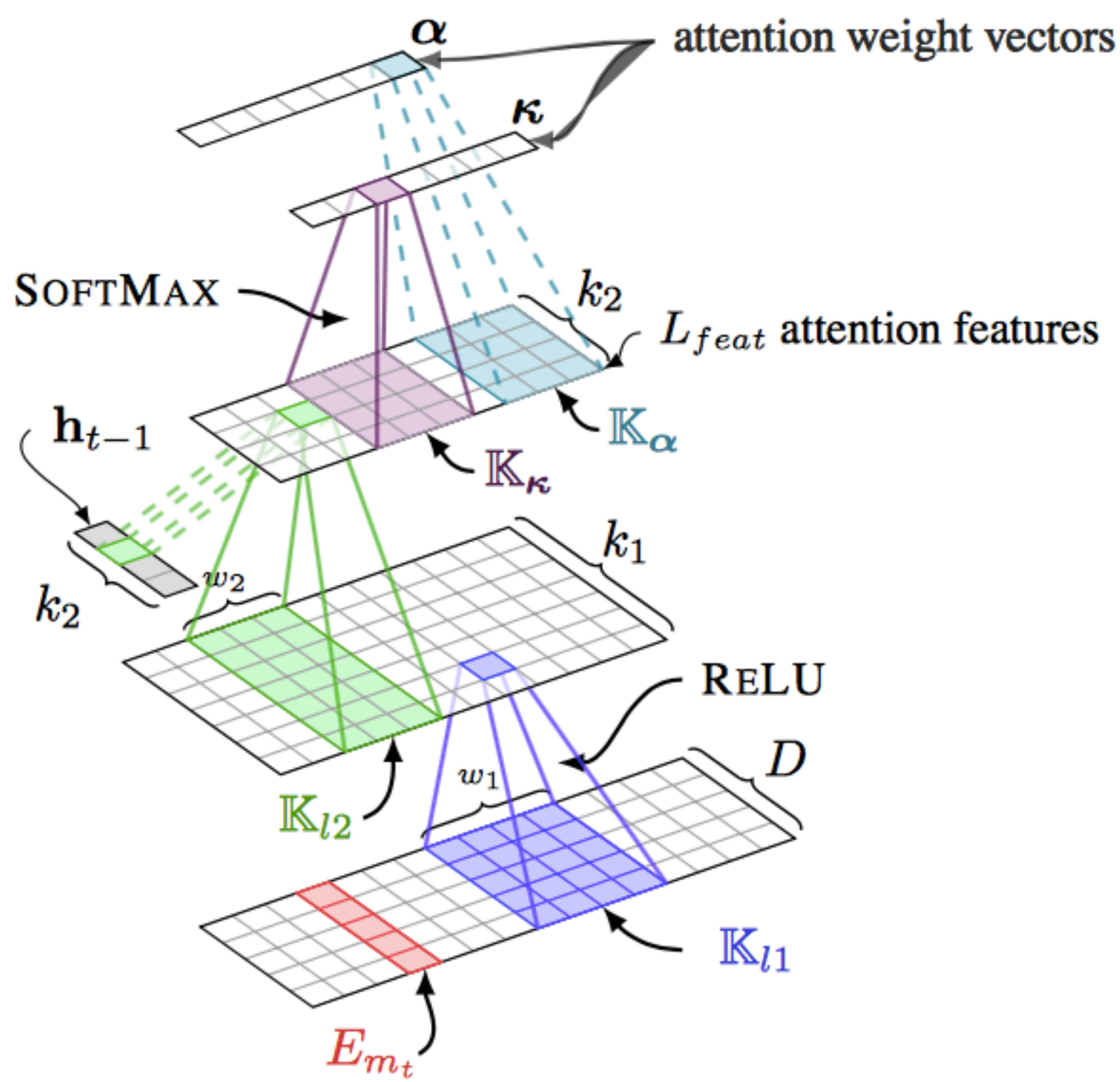
- In some cases, we might know the output will be the same order as the input
  - Speech recognition, incremental translation, morphological inflection (?), summarization (?)



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

Convolutional Attention
(Allamanis et al. 2016)

- **Intuition:** we might want to be able to attend to “the word after ‘Mr.’”, etc.



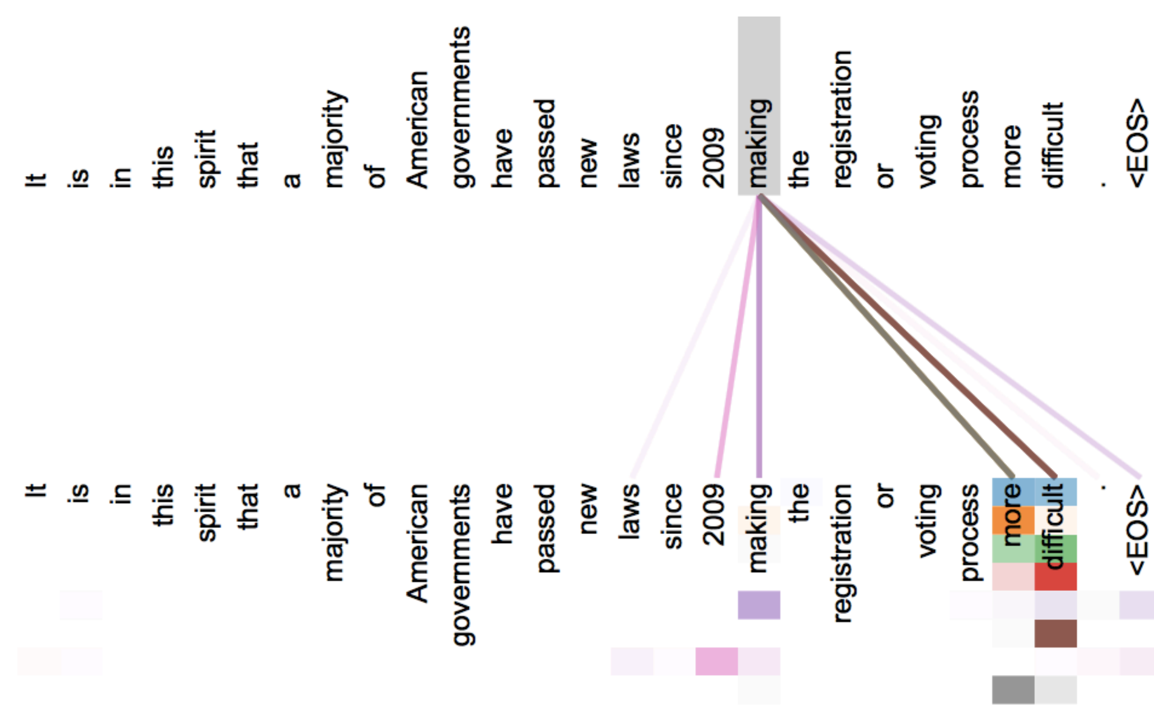
Multi-headed Attention

- **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)



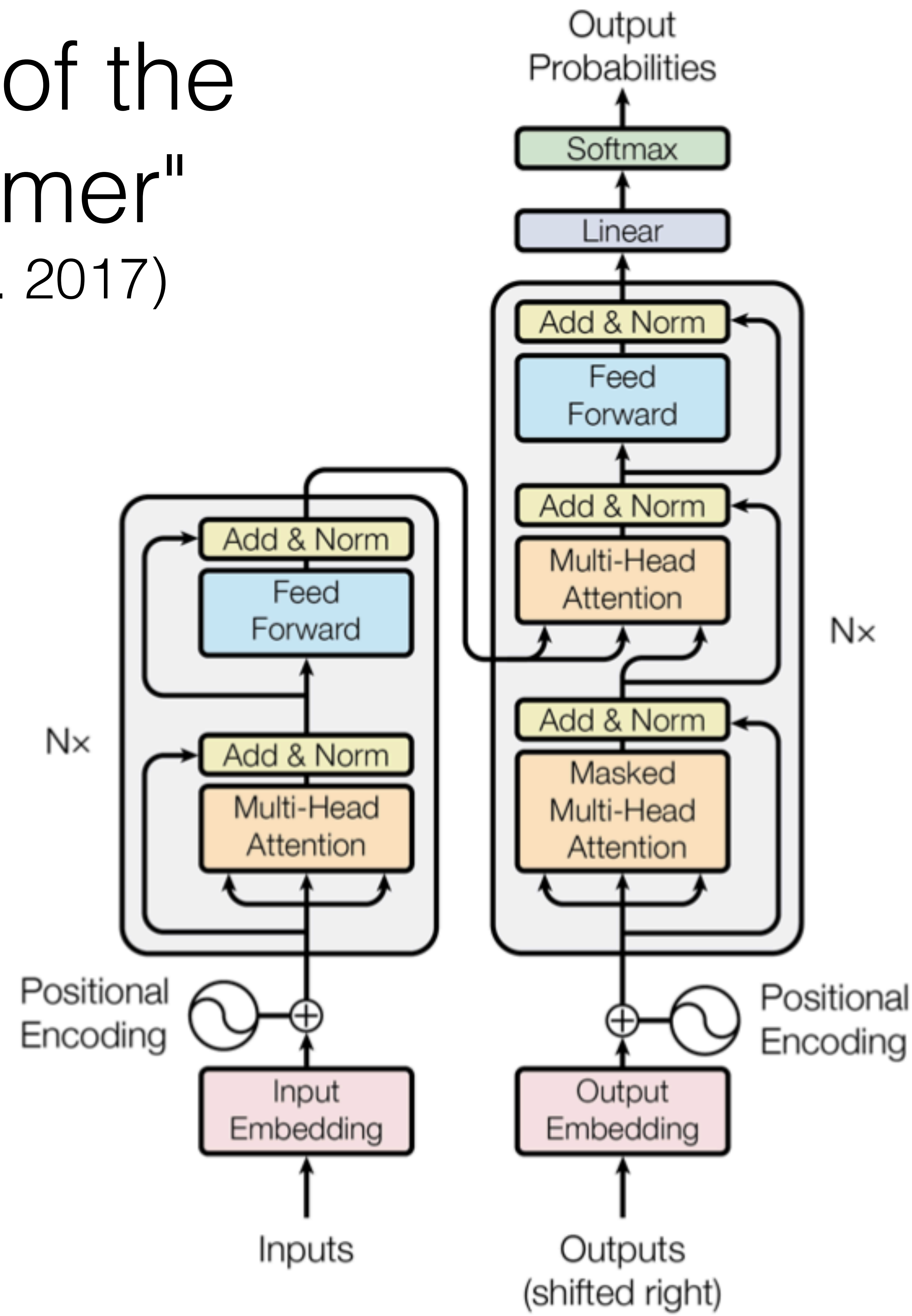
- Or one head for every hidden node! (Choi et al. 2018)

An Interesting Case Study: “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 standard WMT datasets
- Fast: only matrix multiplications



Attention Tricks

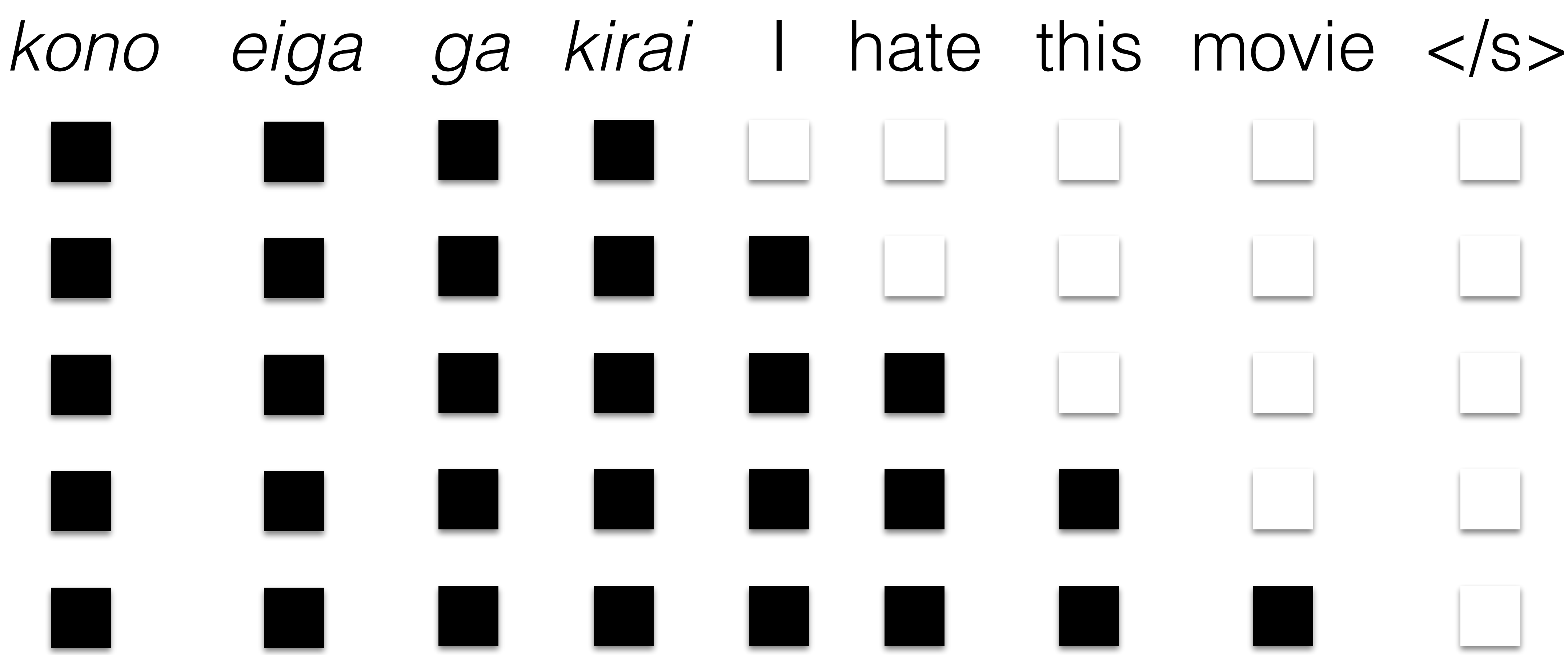
- **Self Attention:** Each layer combines words with others
- **Multi-headed Attention:** 8 attention heads learned independently
- **Normalized Dot-product Attention:** Remove bias in dot product when using large networks
- **Positional Encodings:** Make sure that even if we don't have RNN, can still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

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



Questions?