Sequence-to-sequence Models and Attention

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Preliminaries: Language Models

Language Models

- Language models are generative models of text

$$s \sim P(x)$$

$$\downarrow$$

“The Malfoys!” said Hermione.

Harry was watching him. He looked like Madame Maxime. When she strode up the wrong staircase to visit himself.

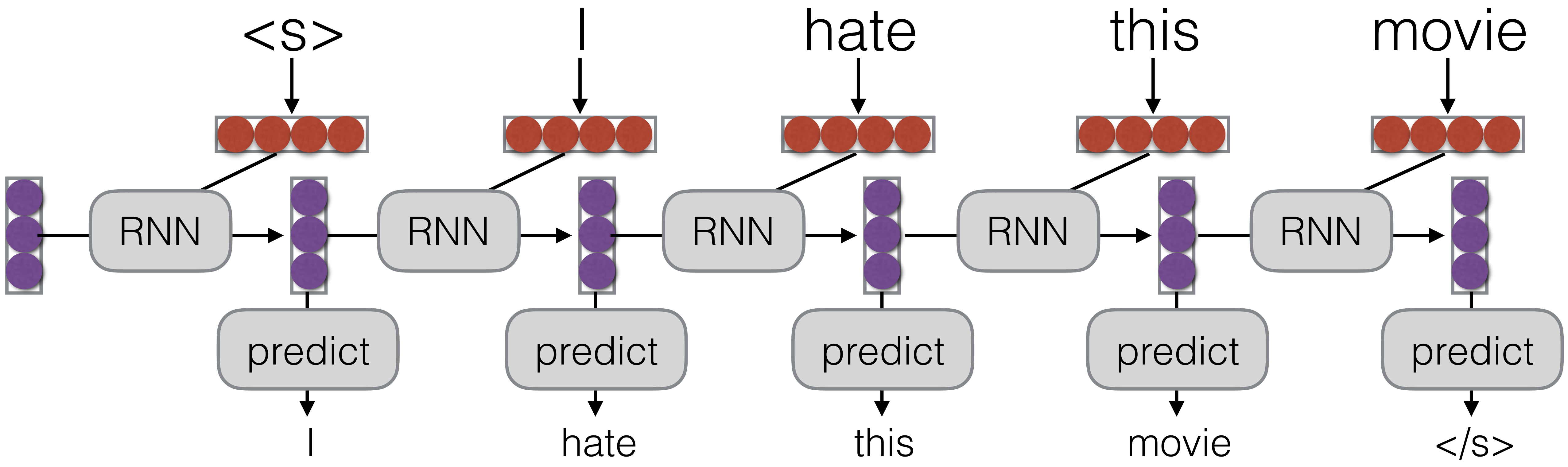
“I’m afraid I’ve definitely been suspended from power, no chance—indeed?” said Snape. He put his head back behind them and read groups as they crossed a corner and fluttered down onto their ink lamp, and picked up his spoon. The doorbell rang. It was a lot cleaner down in London.

Text Credit: Max Deutsch (https://medium.com/deep-writing/

Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

Language Modeling w/ Neural Networks



- At each time step, input the previous word, and predict the probability of the next word

Conditional Language Models

Conditioned Language Models

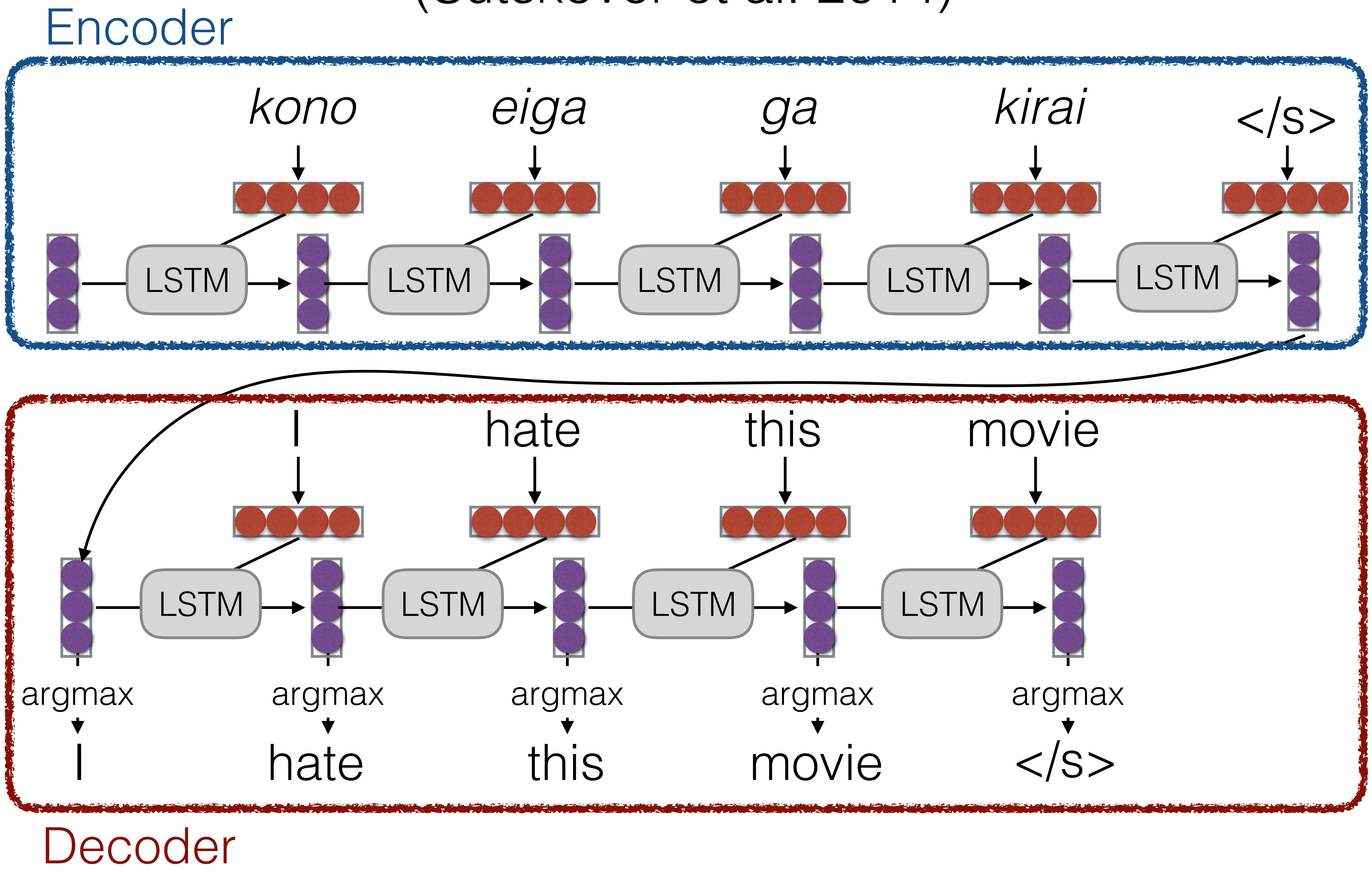
- Not just generate text, generate text according to some specification

Input *X*	Output *Y* (**Text**)	Task
Structured Data	NL Description	NL Generation
English	Japanese	Translation
Document	Short Description	Summarization
Utterance	Response	Response Generation
Image	Text	Image Captioning
Speech	Transcript	Speech Recognition

Conditional Language Models

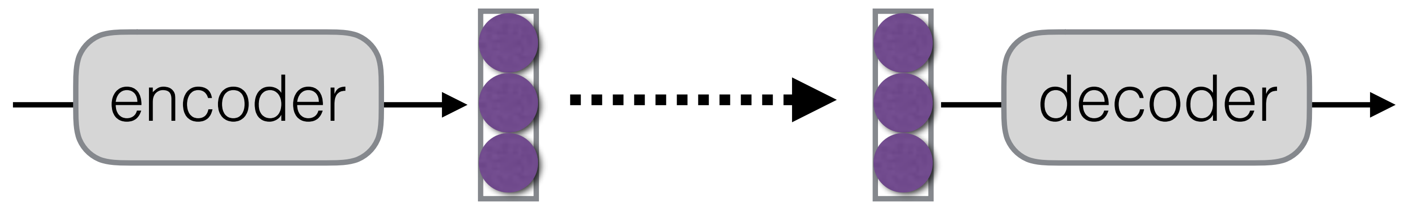
$$P(Y|X) = \prod_{j=1}^{J} P(y_j \mid \underbrace{X}_{\text{Added Context!}}, y_1, \ldots, y_{j-1})$$

(One Type of) Conditional Language Model (Sutskever et al. 2014)

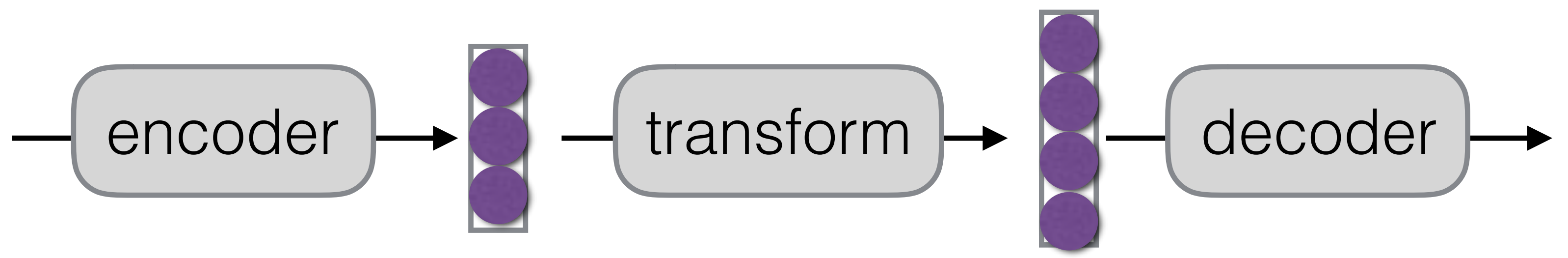


How to Pass Hidden State?

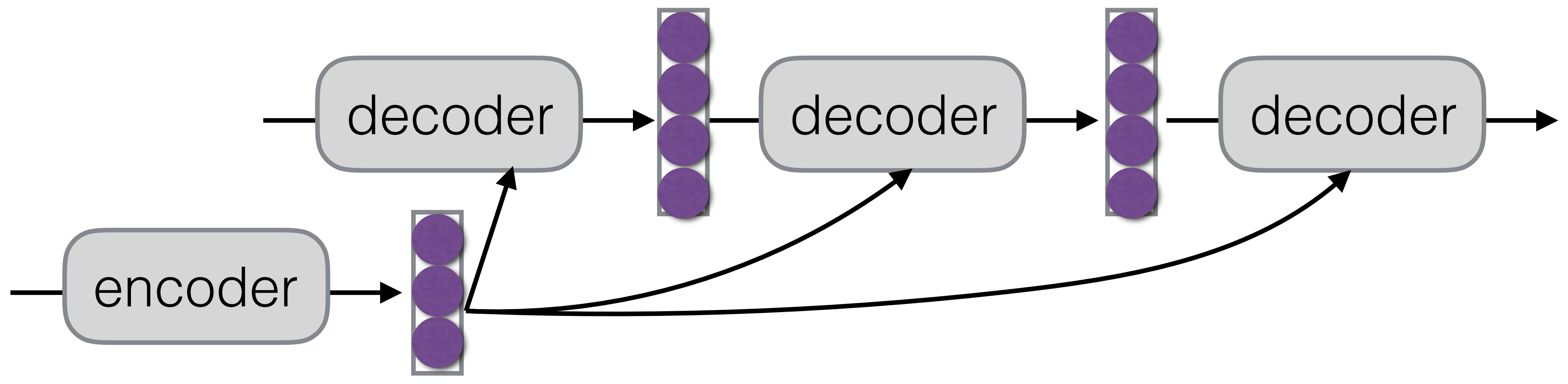
- Initialize decoder w/ encoder (Sutskever et al. 2014)



- Transform (can be different dimensions)



- Input at every time step (Kalchbrenner & Blunsom 2013)



Methods of Generation

The Generation Problem

- We have a model of P(Y|X), how do we use it to generate a sentence?
- Two methods:
  - **Sampling:** Try to generate a *random* sentence according to the probability distribution.
  - **Argmax:** Try to generate the sentence with the *highest* probability.

Ancestral Sampling

- **Randomly generate** words one-by-one.

```
while $y_{j-1}$ != "</s>":
  $y_j \sim P(y_j \mid X, y_1, \ldots, y_{j-1})$
```

- An **exact method** for sampling from P(X), no further work needed.

Greedy Search

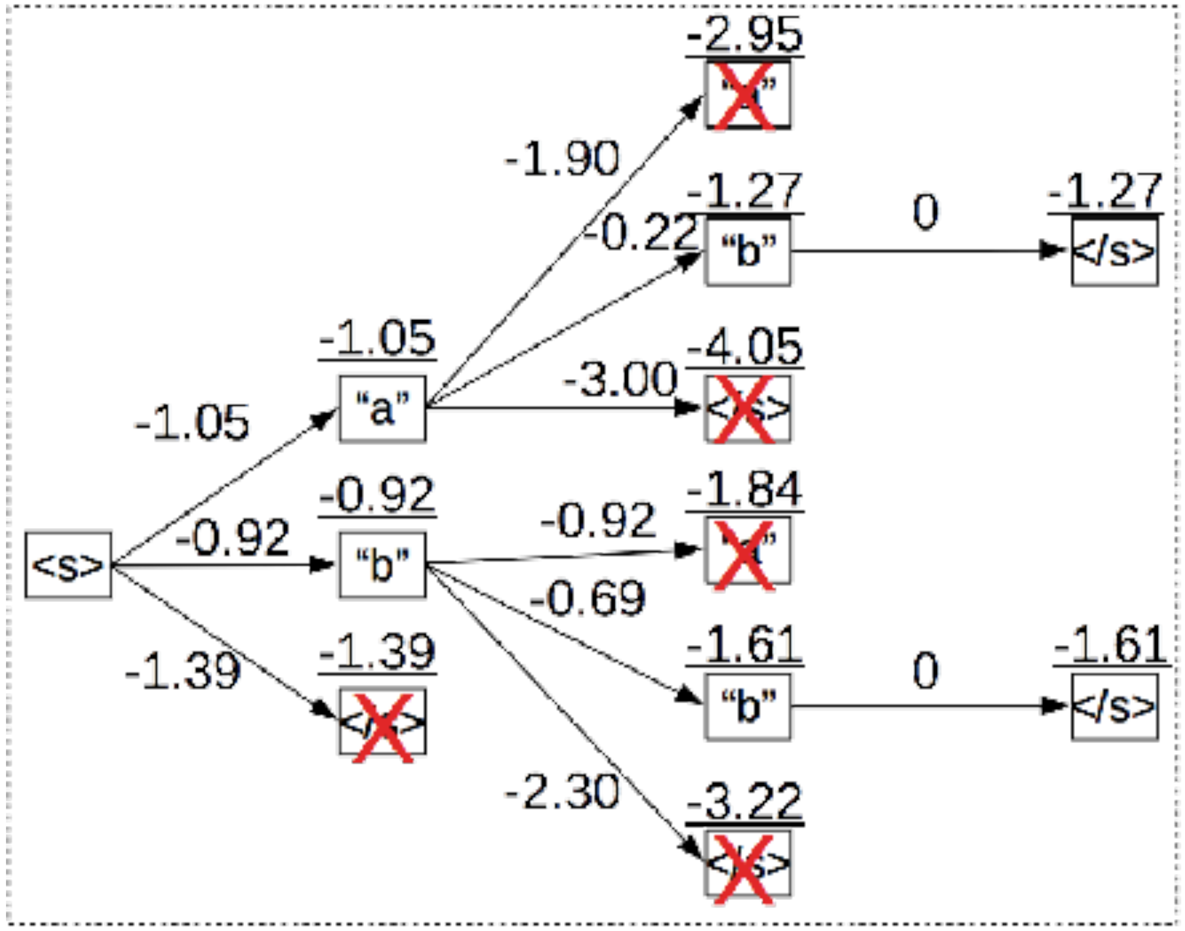
- One by one, pick the single highest-probability word

```
while $y_{j-1}$ != "</s>":
  $y_j = \text{argmax } P(y_j \mid X, y_1, \ldots, y_{j-1})$
```

- **Not exact, real problems:**
  - Will often generate the "easy" words first
  - Will prefer multiple common words to one rare word

Beam Search

- Instead of picking one high-probability word, maintain several paths



- Some in reading materials, more in a later class

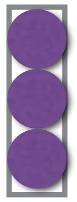
Attention

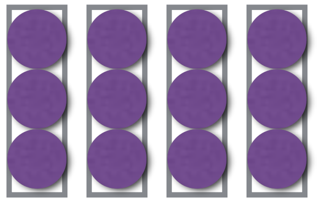
Sentence Representations

Problem!

"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 $\longrightarrow$ 

this is an example $\longrightarrow$ 

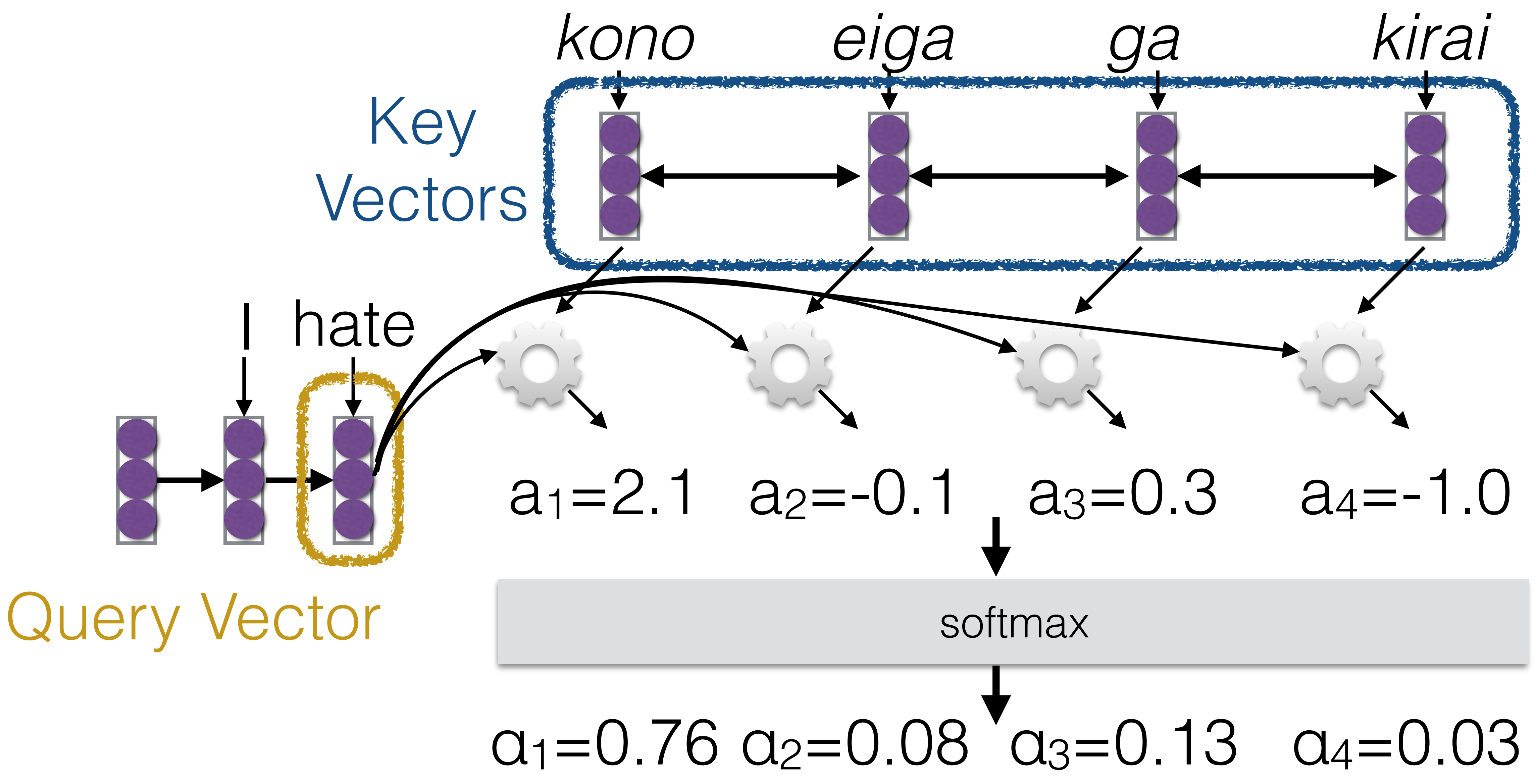
Basic Idea

(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

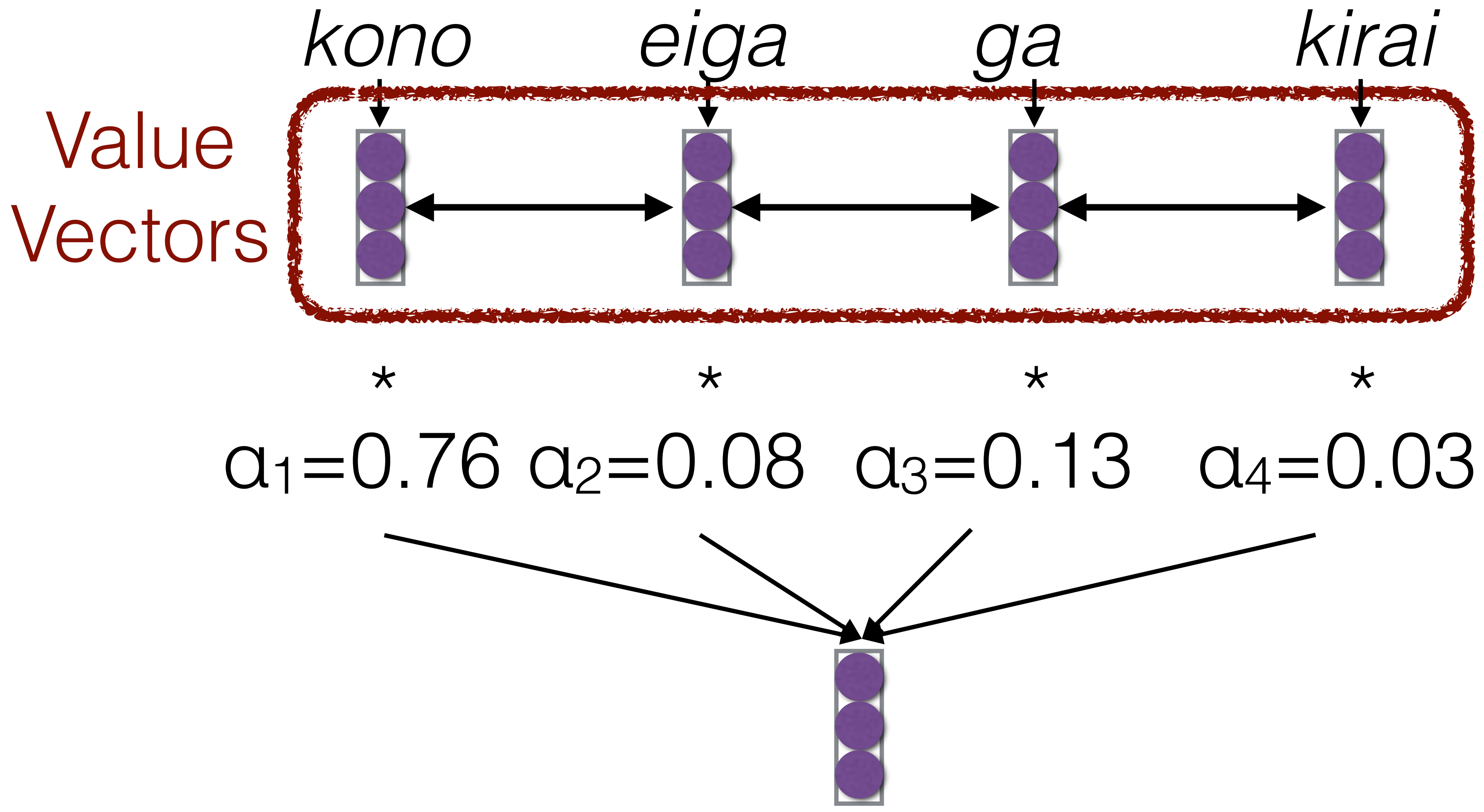
Calculating Attention (1)

- 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



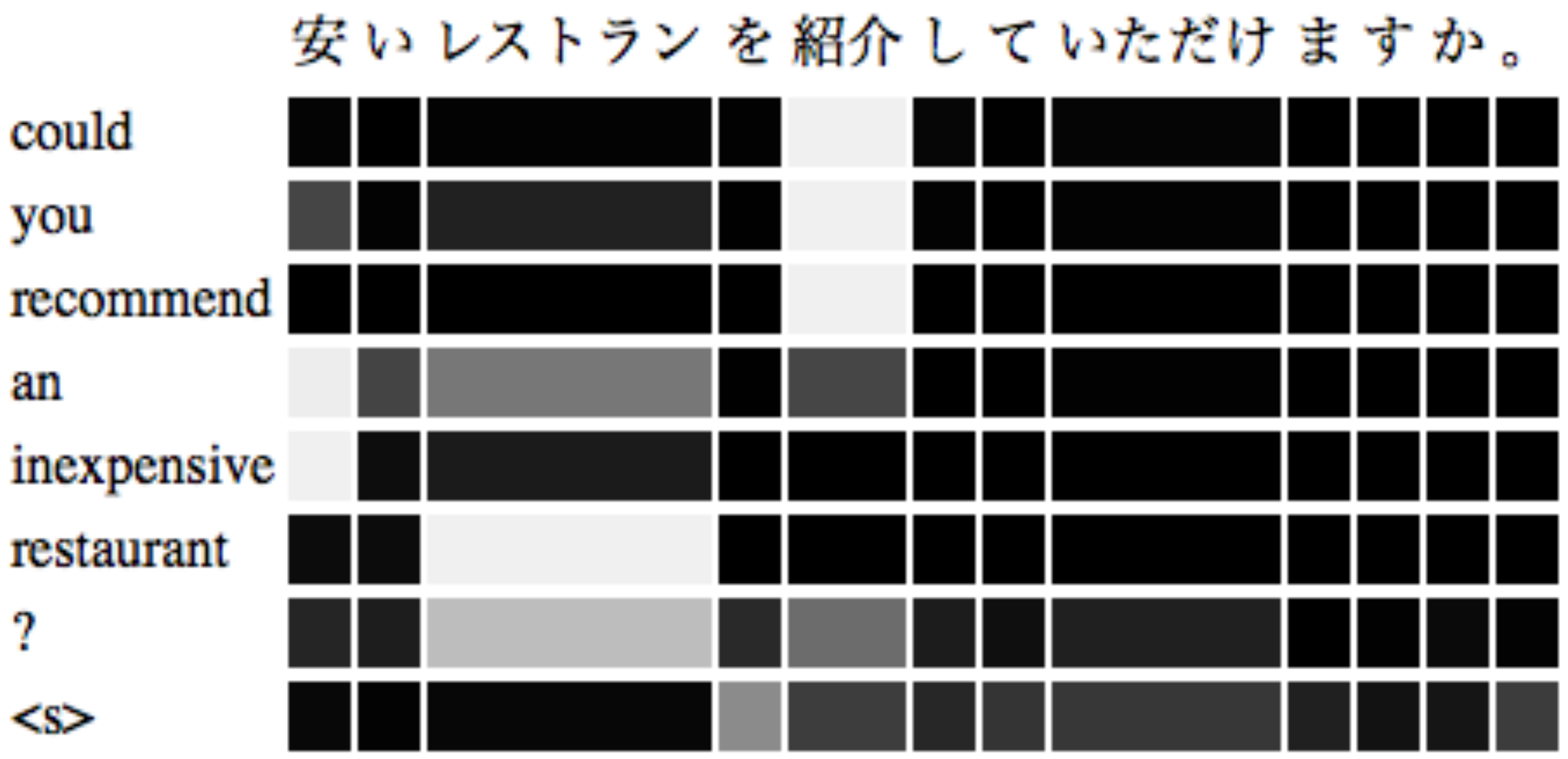
Calculating Attention (2)

- 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.

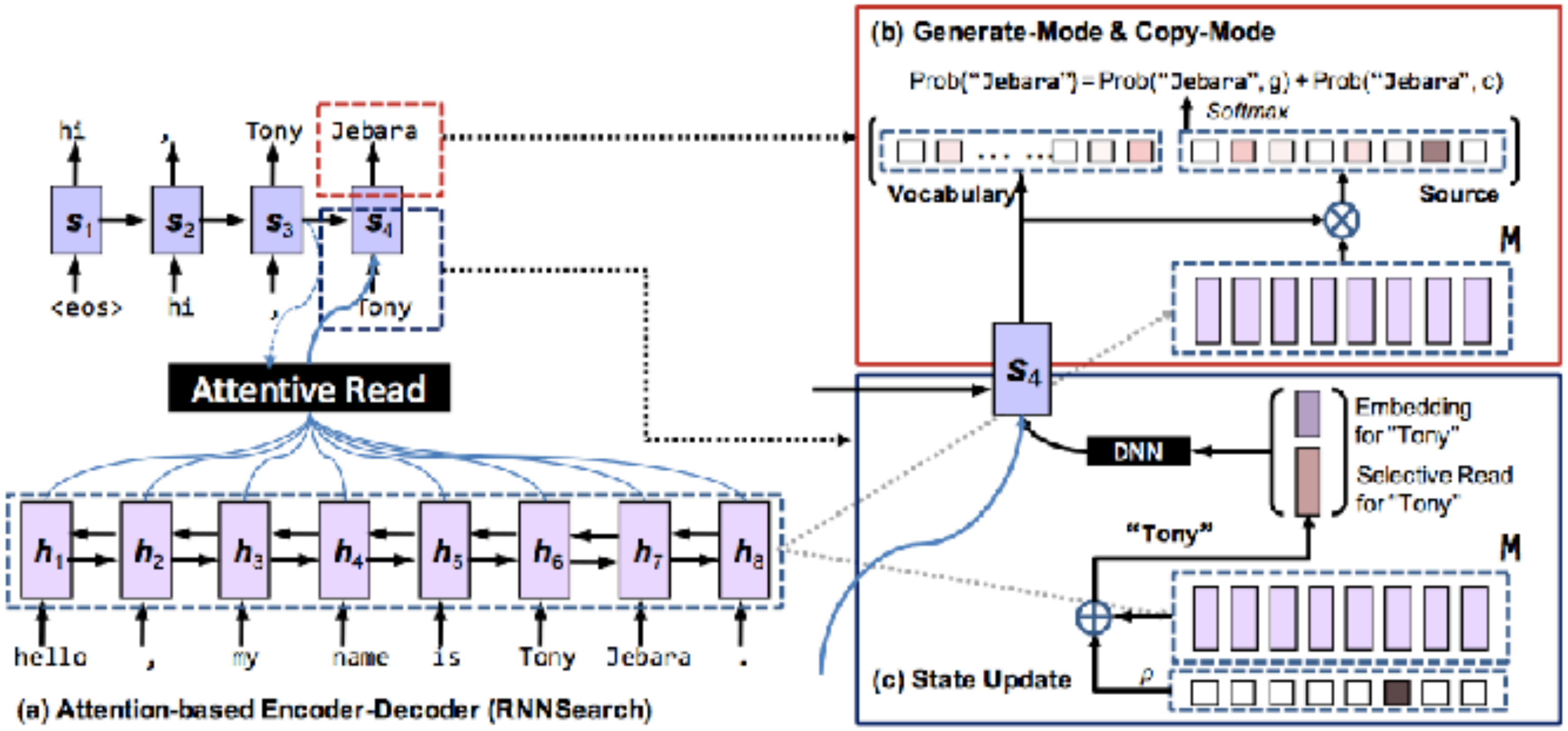
- **Scaled Dot Product** (Vaswani et al. 2017)
  - Problem: scale of dot product increases as dimensions get larger
  - Fix: scale by size of the vector

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

What do we Attend To?

Input Sentence

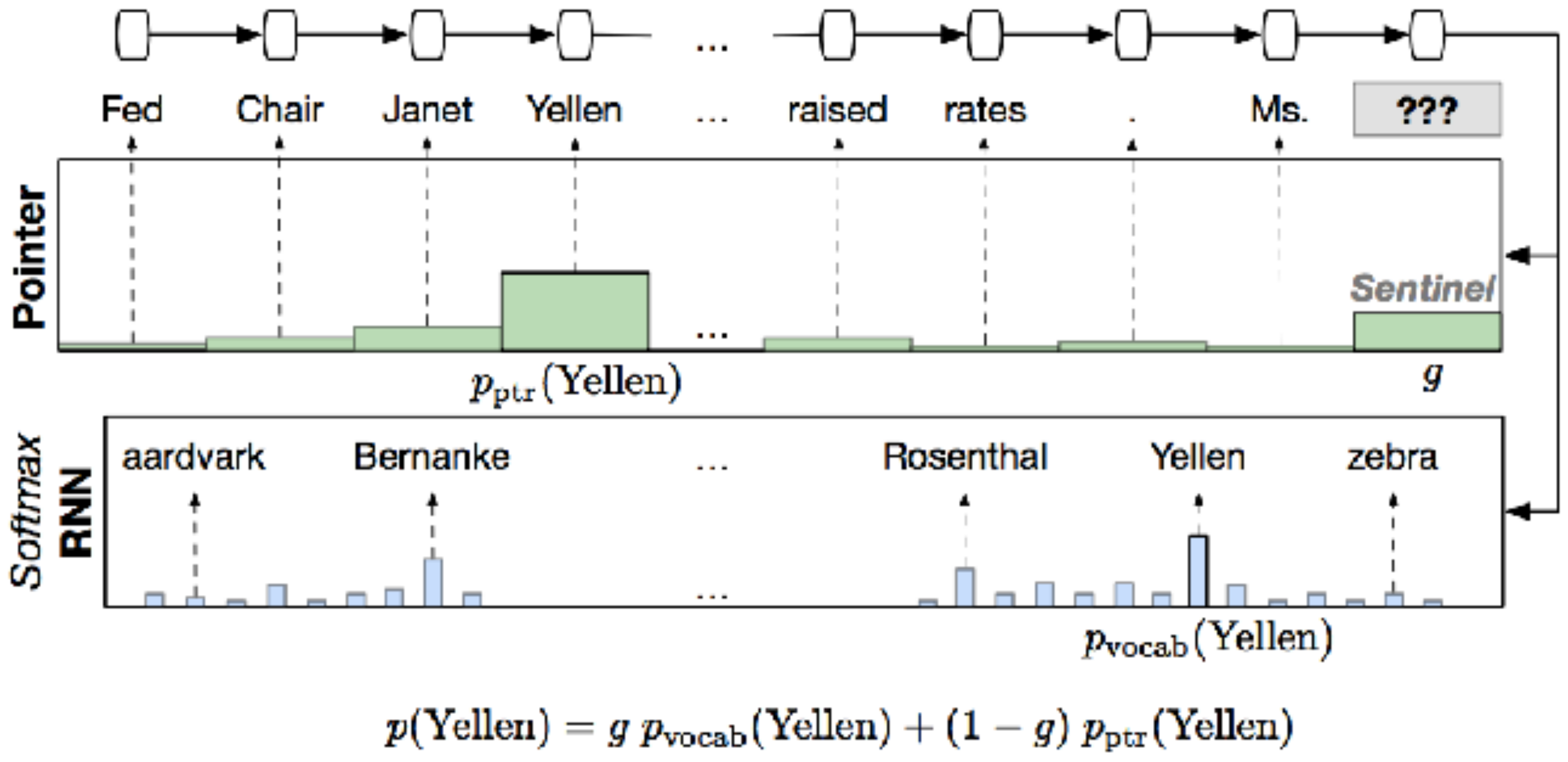
- Like the previous explanation
- But also, more directly
    - **Copying mechanism** (Gu et al. 2016)


(b) Generate-Mode & Copy-Mode
Prob("Jebara") = Prob("Jebara", g) + Prob("Jebara", c)
Softmax
Vocabulary
Source
M
hi
,
Tony
Jebara
<eos>
hi
,
Tony
Attentive Read
DNN
Embedding for "Tony"
Selective Read for "Tony"
"Tony"
M
ρ
hello
,
my
name
is
Tony
Jebara
.
(a) Attention-based Encoder-Decoder (RNNSearch)
(c) State Update

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

Previously Generated Things

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

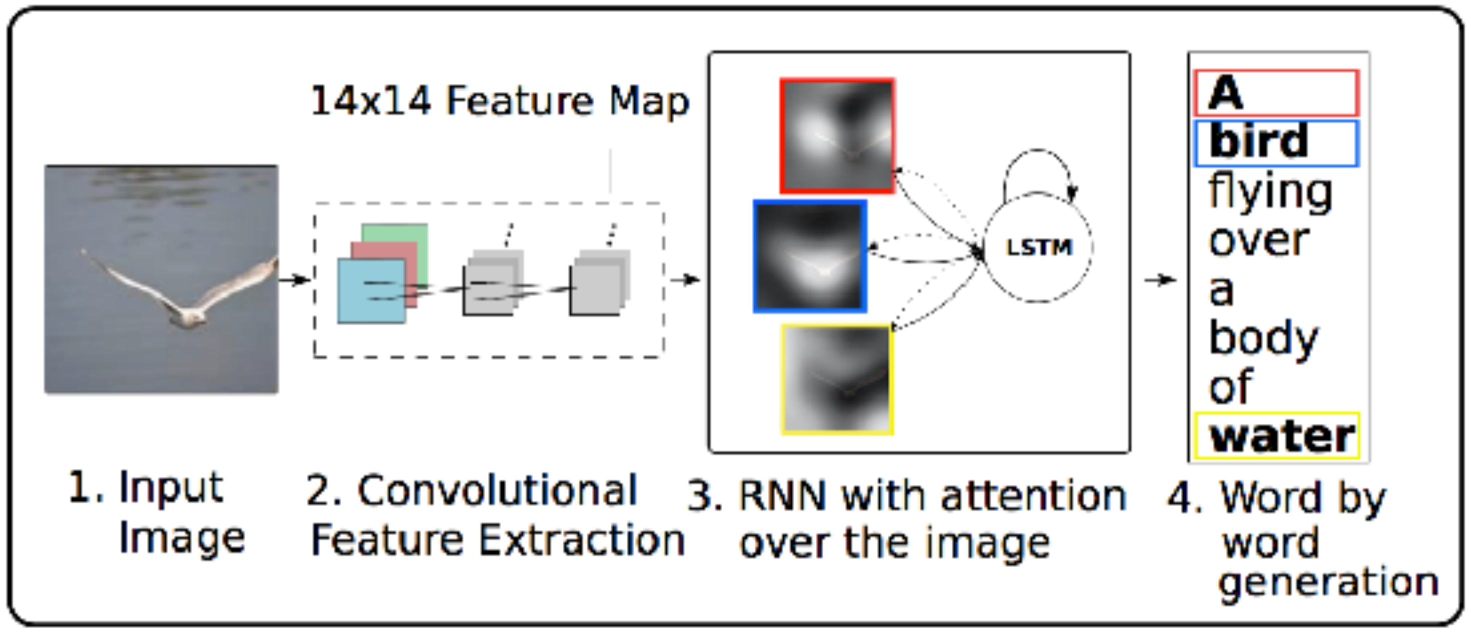


Fed
Chair
Janet
Yellen
...
raised
rates
.
Ms.
???
Pointer
Sentinel
$p_{\mathrm{ptr}}(\mathrm{Yellen})$
g
Softmax
RNN
aardvark
Bernanke
Rosenthal
Yellen
zebra
$p_{\mathrm{vocab}}(\mathrm{Yellen})$
$p(\mathrm{Yellen}) = g\, p_{\mathrm{vocab}}(\mathrm{Yellen}) + (1-g)\, p_{\mathrm{ptr}}(\mathrm{Yellen})$

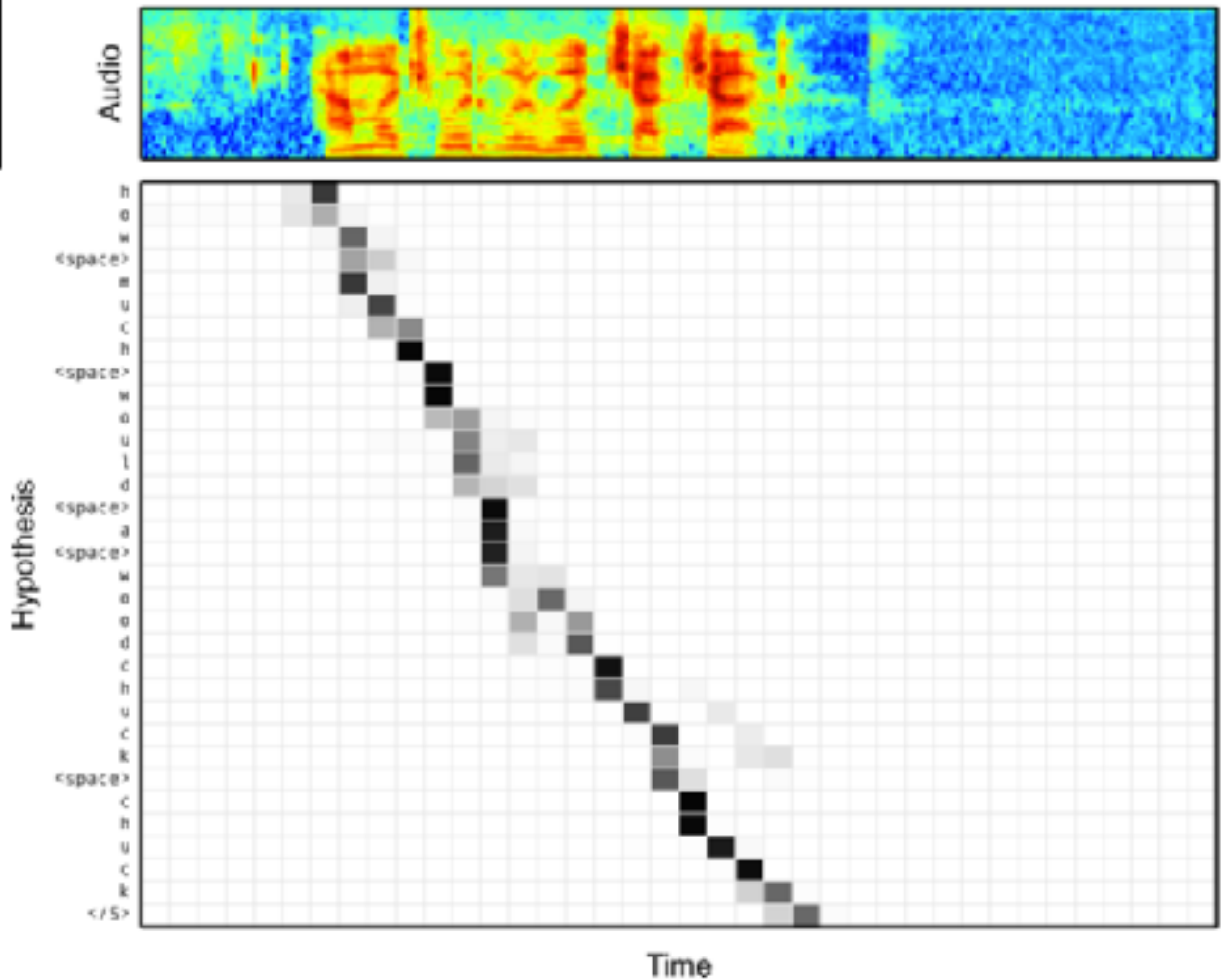
- In translation, attend to either input or previous output (Vaswani et al. 2017)

Various Modalities

- Images (Xu et al. 2015)

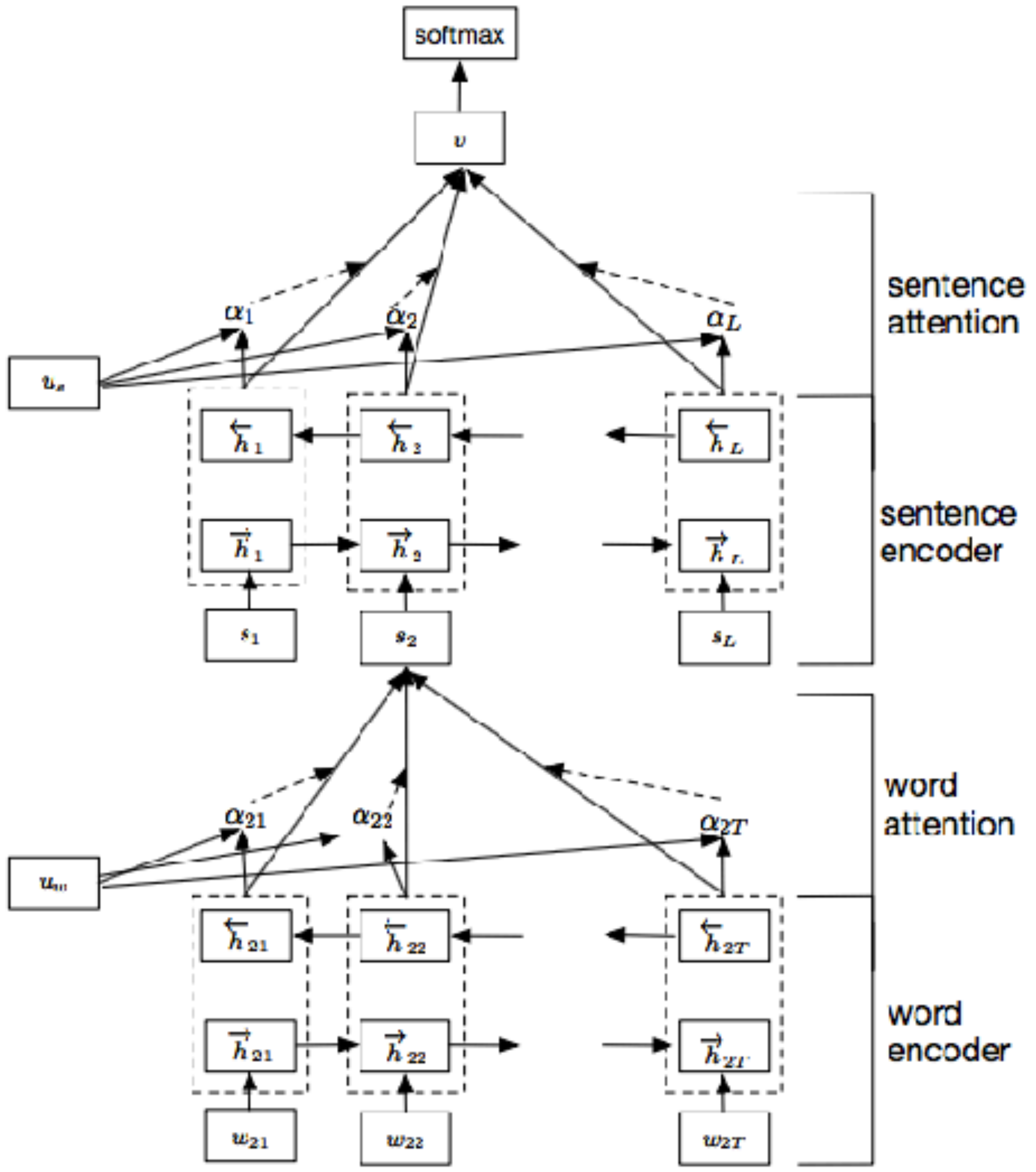

14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

- Speech (Chan et al. 2015)



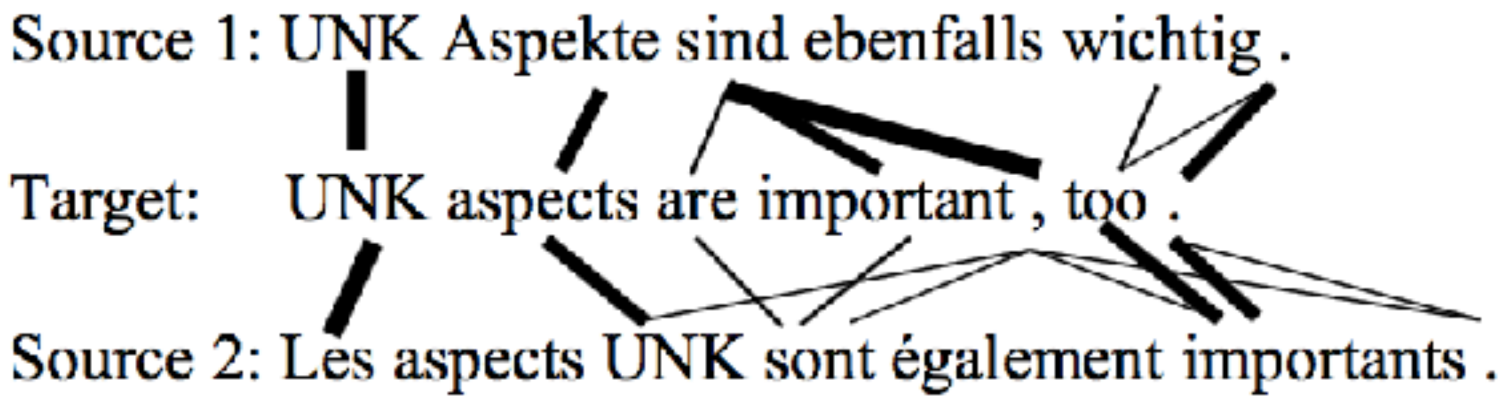
Hierarchical Structures
(Yang et al. 2016)

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



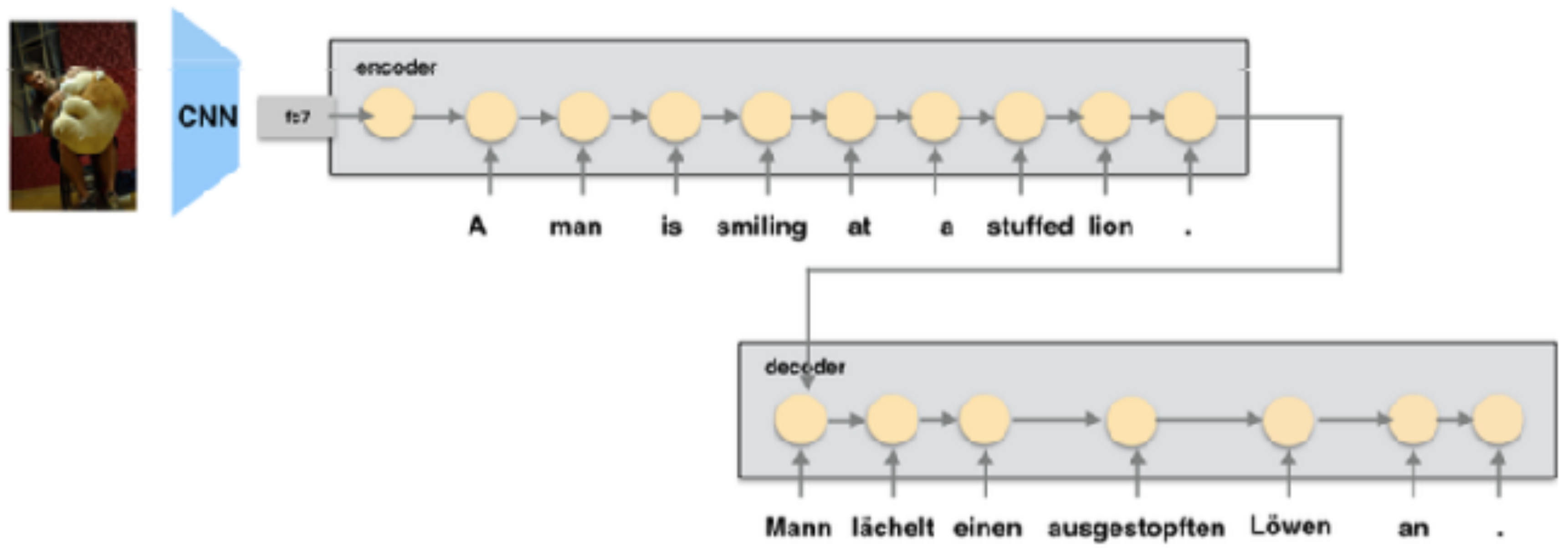
Multiple Sources

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


Source 1: UNK Aspekte sind ebenfalls wichtig .
Target: UNK aspects are important , too .
Source 2: Les aspects UNK sont également importants .

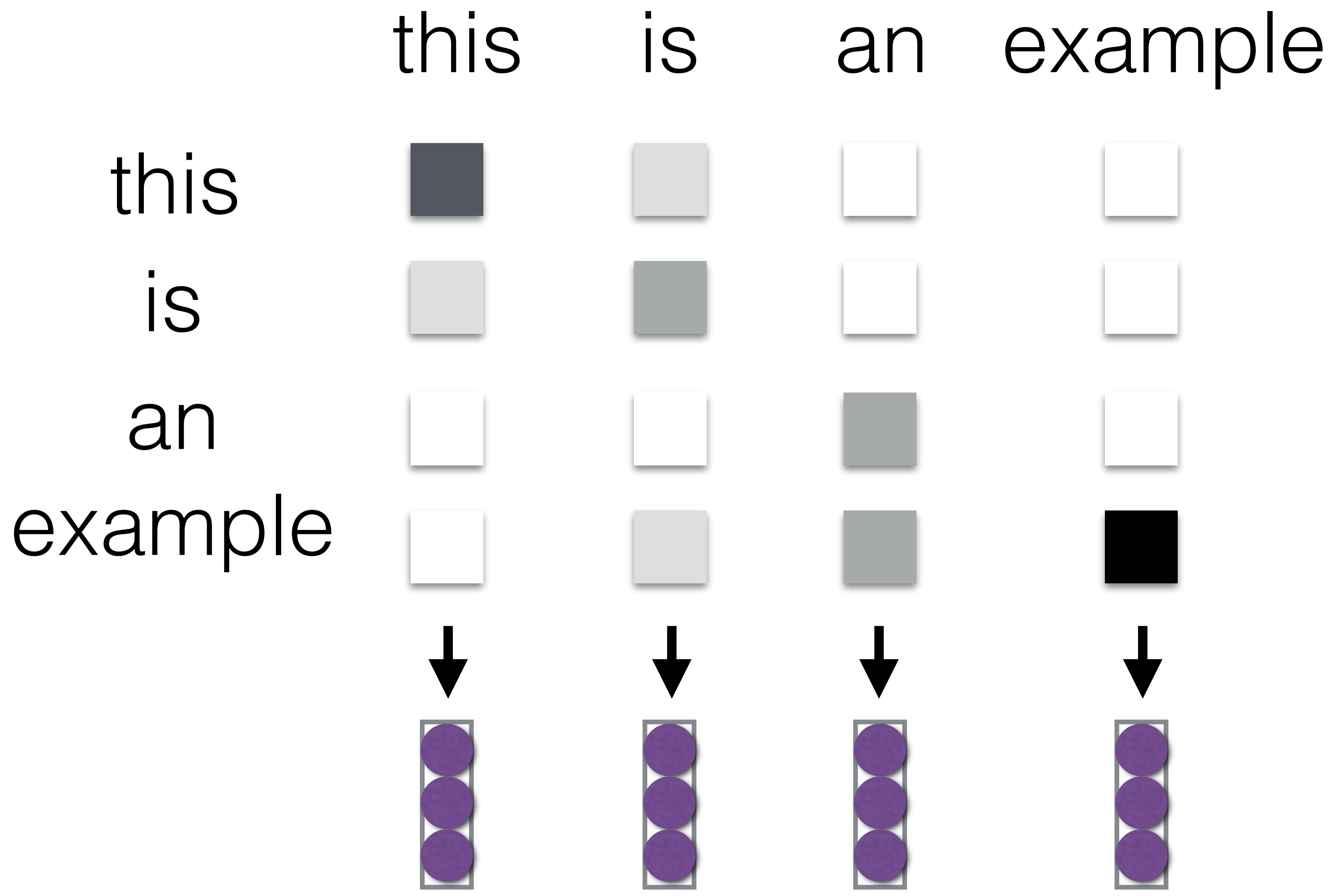
  - Libovicky and Helcl (2017) compare multiple strategies

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


CNN
fc7
encoder
A man is smiling at a stuffed lion .
decoder
Mann lächelt einen ausgestopften Löwen an .

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

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



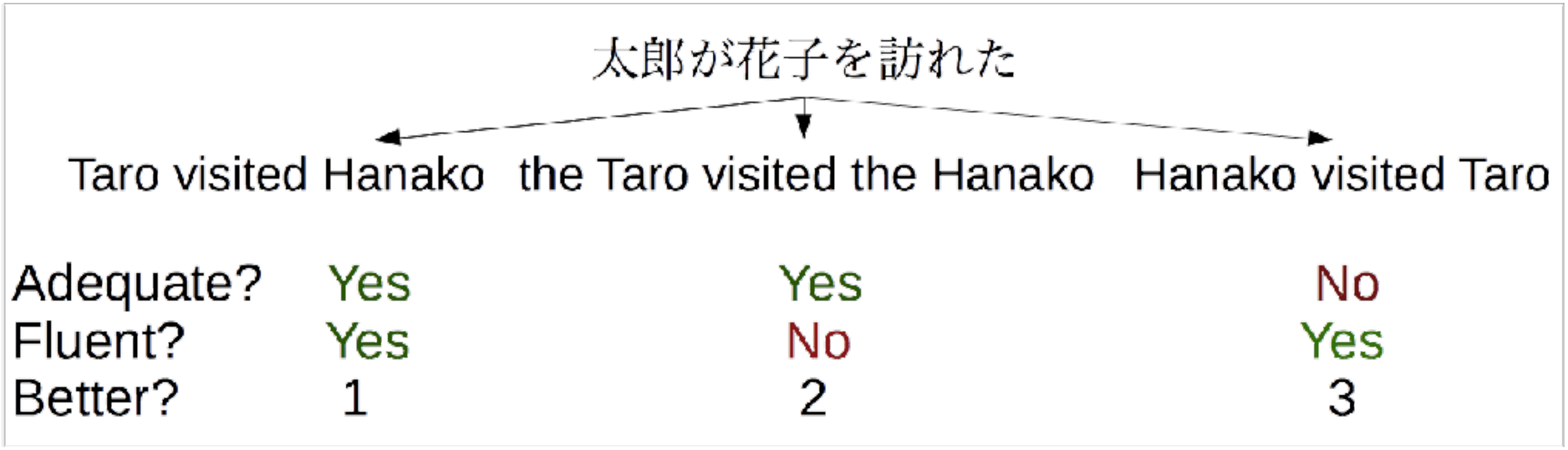
How do we Evaluate?

Basic Evaluation Paradigm

- Use parallel test set
- Use system to generate translations
- Compare target translations w/ reference

Human Evaluation

- Ask a human to do evaluation

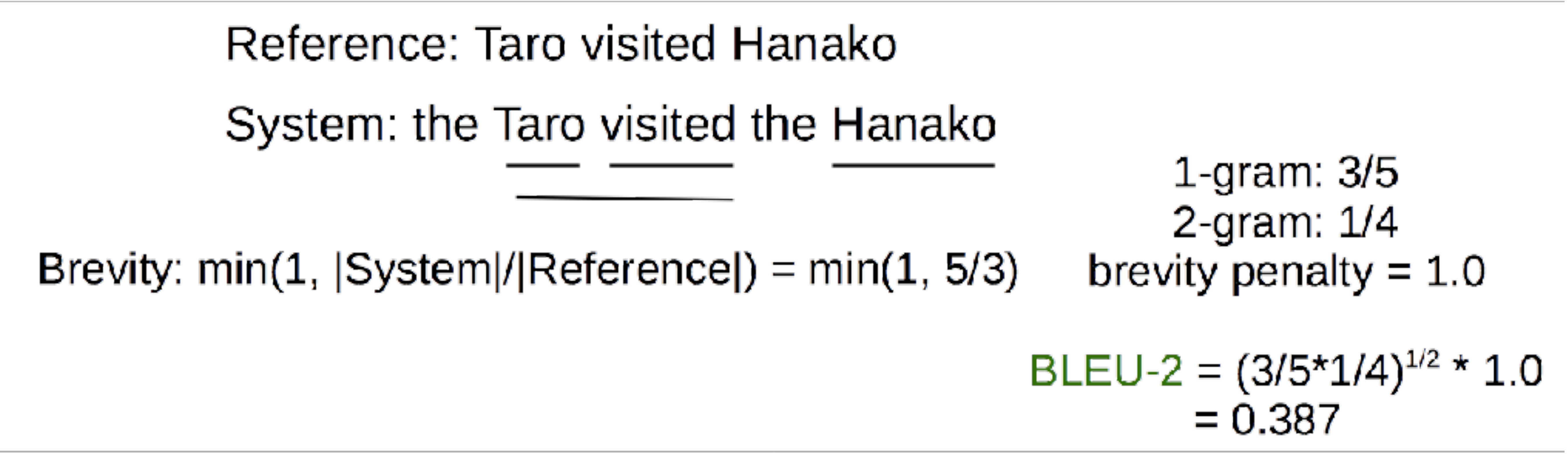

太郎が花子を訪れた

	Taro visited Hanako	the Taro visited the Hanako	Hanako visited Taro
Adequate?	Yes	Yes	No
Fluent?	Yes	No	Yes
Better?	1	2	3

- Final goal, but slow, expensive, and sometimes inconsistent

BLEU

- Works by comparing n-gram overlap w/ reference



Reference: Taro visited Hanako

System: the Taro visited the Hanako

1-gram: 3/5
2-gram: 1/4

Brevity: min(1, |System|/|Reference|) = min(1, 5/3)

brevity penalty = 1.0

BLEU-2 = $(3/5*1/4)^{1/2} * 1.0$
= 0.387

- **Pros:** Easy to use, good for measuring system improvement
- **Cons:** Often doesn’t match human eval, bad for comparing very different systems

METEOR

- Like BLEU in overall principle, with many other tricks: consider paraphrases, reordering, and function word/content word difference

- **Pros:** Generally significantly better than BLEU, esp. for high-resource languages

- **Cons:** Requires extra resources for new languages (although these can be made automatically), and more complicated

Perplexity

- Calculate the perplexity of the words in the held-out set *without* doing generation
- **Pros:** Naturally solves multiple-reference problem!
- **Cons:** Doesn’t consider decoding or actually generating output.
- May be reasonable for problems with lots of ambiguity.

Questions?