CS11-747 Neural Networks for NLP

Conditioned Generation

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

Site
https://phontron.com/class/nn4nlp2017/

Language Models

- Language models are generative models of text

$$s \sim P(x)$$

“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/)

Conditioned Language Models

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

<u>Input *X*</u>	<u>Output *Y* (**Text**)</u>	<u>Task</u>
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

Formulation and Modeling

Calculating the Probability of a Sentence

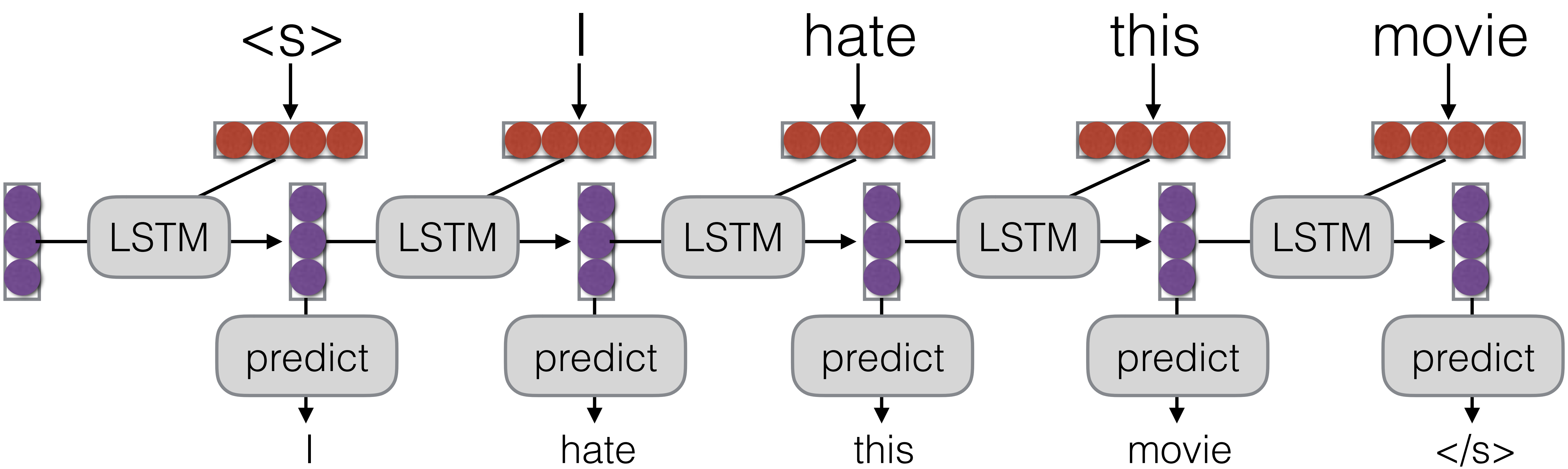
$$P(X) = \prod_{i=1}^{I} P(\underbrace{x_i}_{\text{Next Word}} \mid \underbrace{x_1, \ldots, x_{i-1}}_{\text{Context}})$$

Conditional Language Models

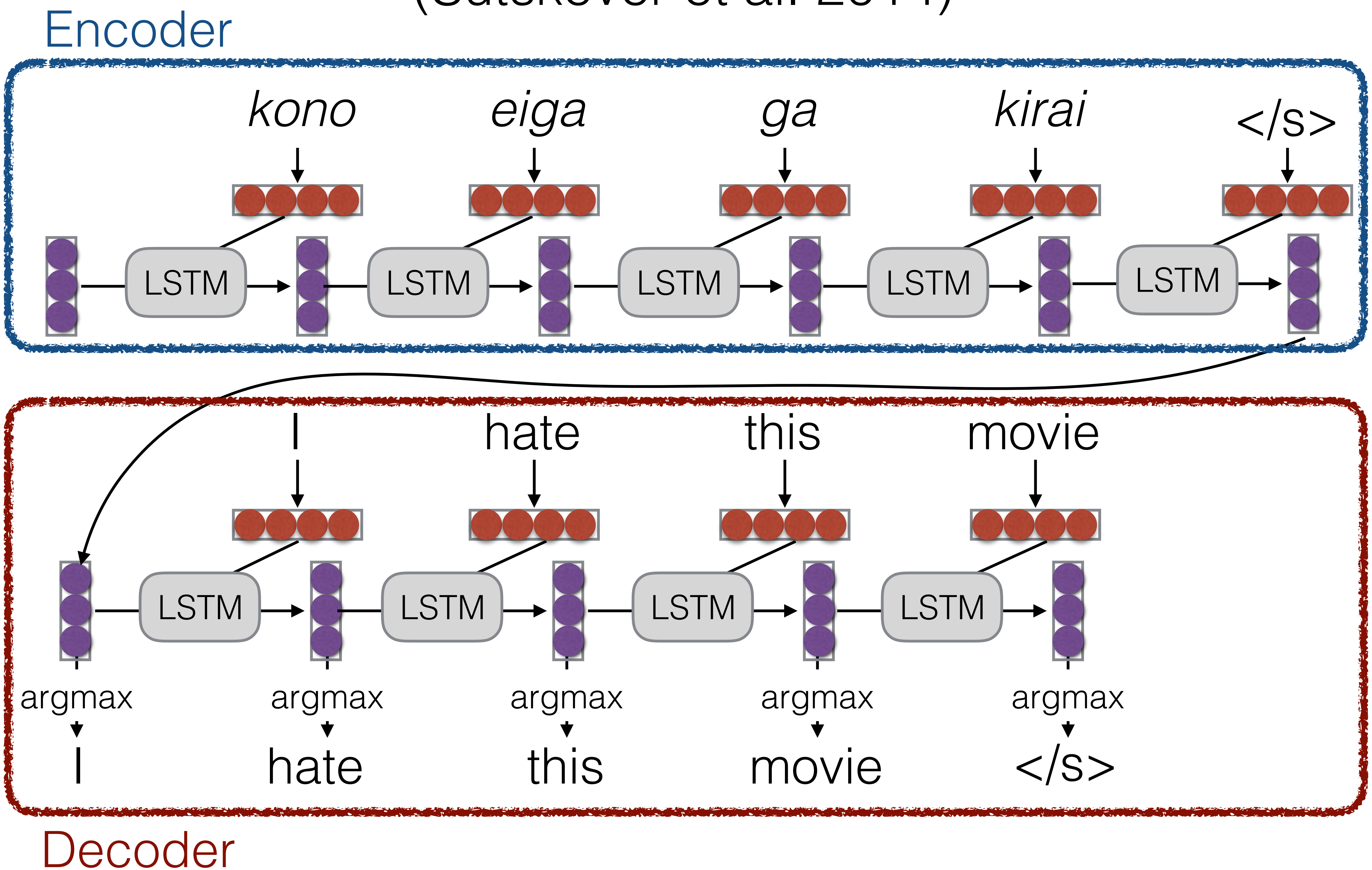
$$P(Y|X) = \prod_{j=1}^{J} P(y_j \mid \underline{X}, y_1, \ldots, y_{j-1})$$

Added Context!

(One Type of) Language Model
(Mikolov et al. 2011)

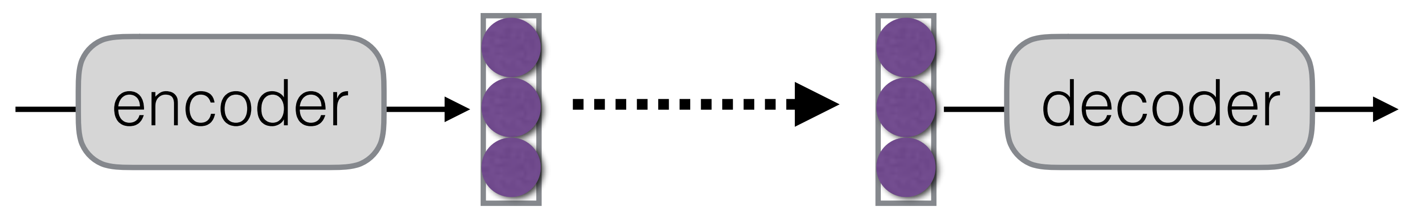


(One Type of) Conditional Language Model
(Sutskever et al. 2014)



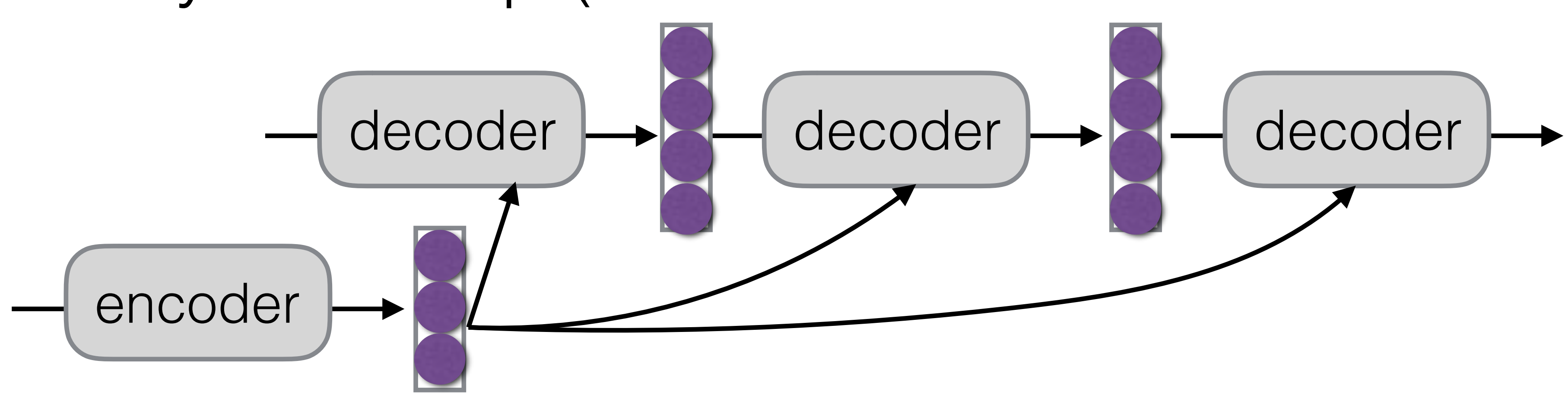
How to Pass Hidden State?

- 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.

$$\boxed{\begin{aligned}&\text{while } y_{j-1} \text{ != ``</s>'':}\\ &\quad y_j \sim P(y_j \mid X, y_1, \ldots, y_{j-1})\end{aligned}}$$

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

Greedy Search

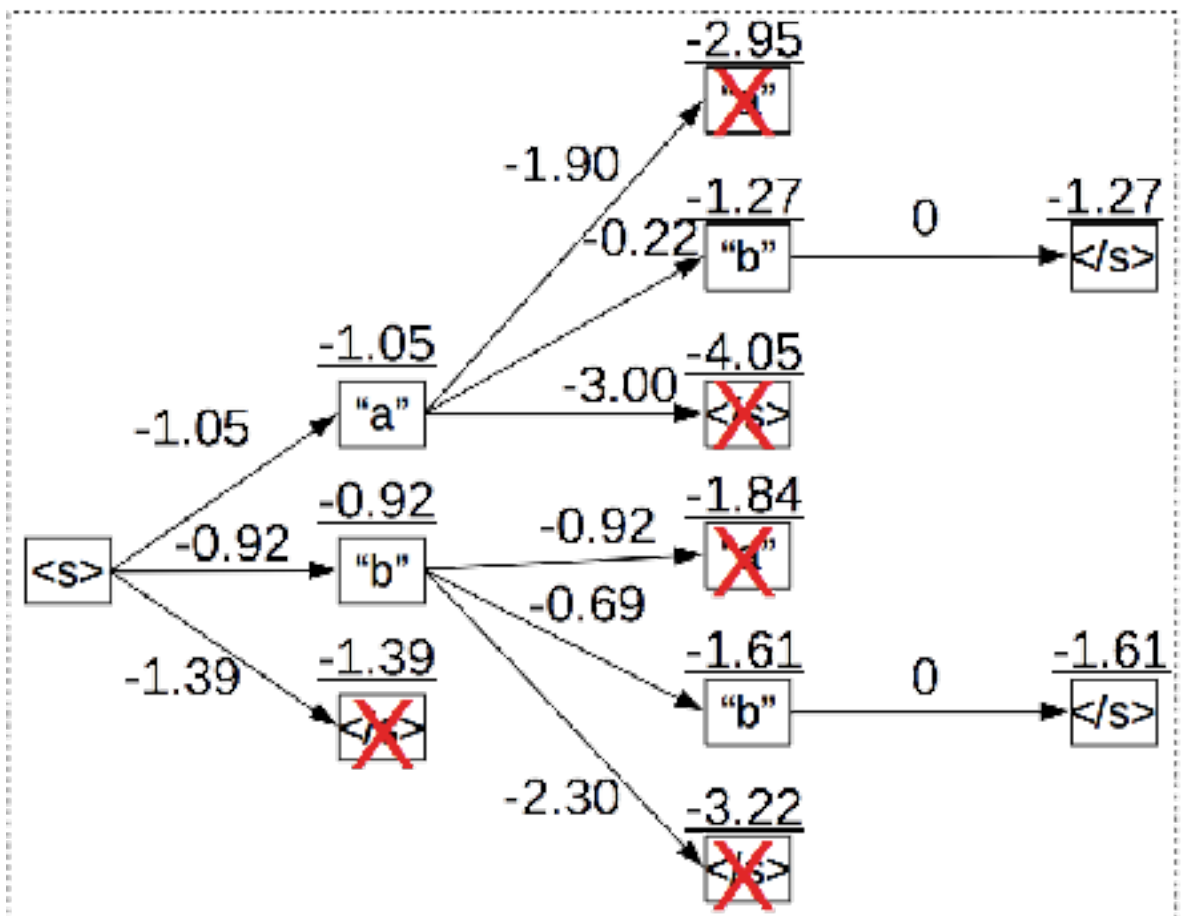
- One by one, pick the single highest-probability word

```
while y_{j-1} != "</s>":
  y_j = argmax P(y_j | X, y_1, …, 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

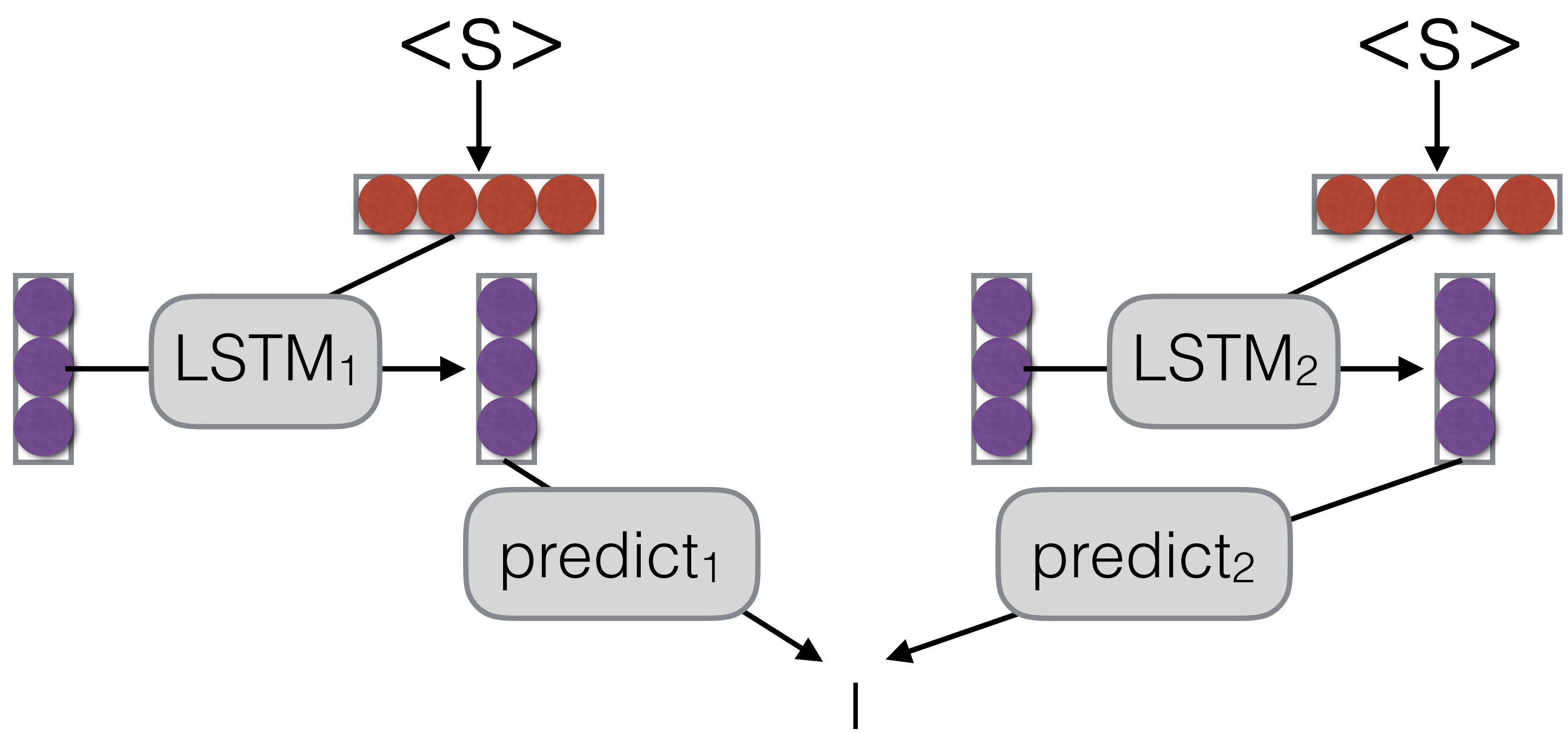
Let’s Try it Out!

`enc_dec.py`

Model Ensembling

Ensembling

- Combine predictions from multiple models



- Why?
  - Multiple models make somewhat uncorrelated errors
  - Models tend to be more uncertain when they are about to make errors
  - Smooths over idiosyncrasies of the model

Linear Interpolation

- Take a weighted average of the M model probabilities

$$P(y_j \mid X, y_1, \ldots, y_{j-1}) = \sum_{m=1}^{M} \underbrace{P_m(y_j \mid X, y_1, \ldots, y_{j-1})}_{\text{Probability according to model } m} \underbrace{P(m \mid X, y_1, \ldots, y_{j-1})}_{\text{Probability of model } m}$$

- Second term often set to uniform distribution 1/M

Log-linear Interpolation

- Weighted combination of log probabilities, normalize

$$P(y_j \mid X, y_1, \ldots, y_{j-1}) =$$

$$\underbrace{\mathrm{softmax}}_{\text{Normalize}} \left(\sum_{m=1}^{M} \underbrace{\lambda_m(X, y_1, \ldots, y_{j-1})}_{\text{Interpolation coefficient for model } m} \underbrace{\log P_m(y_j \mid X, y_1, \ldots, y_{j-1})}_{\text{Log probability of model } m} \right)$$

- Interpolation coefficient often set to uniform distribution 1/M

Linear or Log Linear?

- Think of it in logic!
- **Linear:** “Logical OR”
  - the interpolated model likes any choice that a model gives a high probability
  - use models with models that capture different traits
  - necessary when any model can assign zero probability
- **Log Linear:** “Logical AND”
  - interpolated model only likes choices where all models agree
  - use when you want to restrict possible answers

Parameter Averaging

- **Problem:** Ensembling means we have to use *M* models at test time, increasing our time/memory complexity

- Parameter averaging is a cheap way to get some good effects of ensembling

- Basically, write out models several times near the end of training, and take the average of parameters

Ensemble Distillation (e.g. Kim et al. 2016)

- **Problem:** parameter averaging only works for models within the same run
- Knowledge distillation trains a model to **copy the ensemble**
  - Specifically, it tries to match the description over predicted words
  - Why? We want the model to make the same mistakes as an ensemble
- Shown to increase accuracy notably

Stacking

- What if we have two very different models where prediction of outputs is done in very different ways?
- e.g. a word-by-word translation model and character-by-character translation model
- Stacking uses the **output of one system in calculating features for another** system

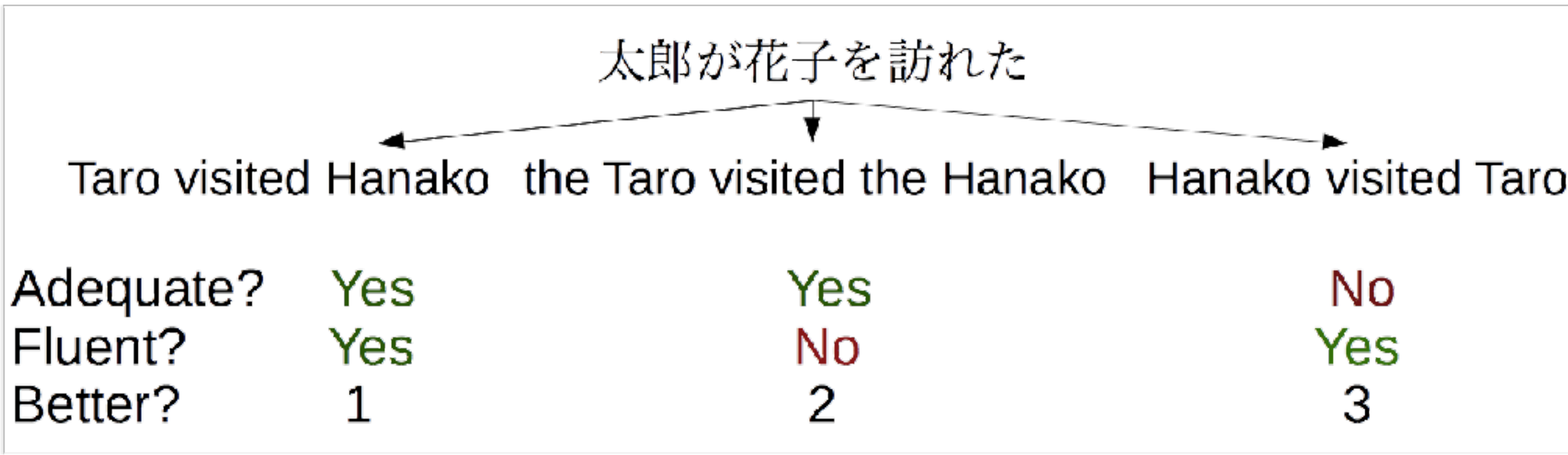
How do we Evaluate?

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



- 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.

What Do We Condition On?

From Structured Data
(e.g. Wen et al 2015)

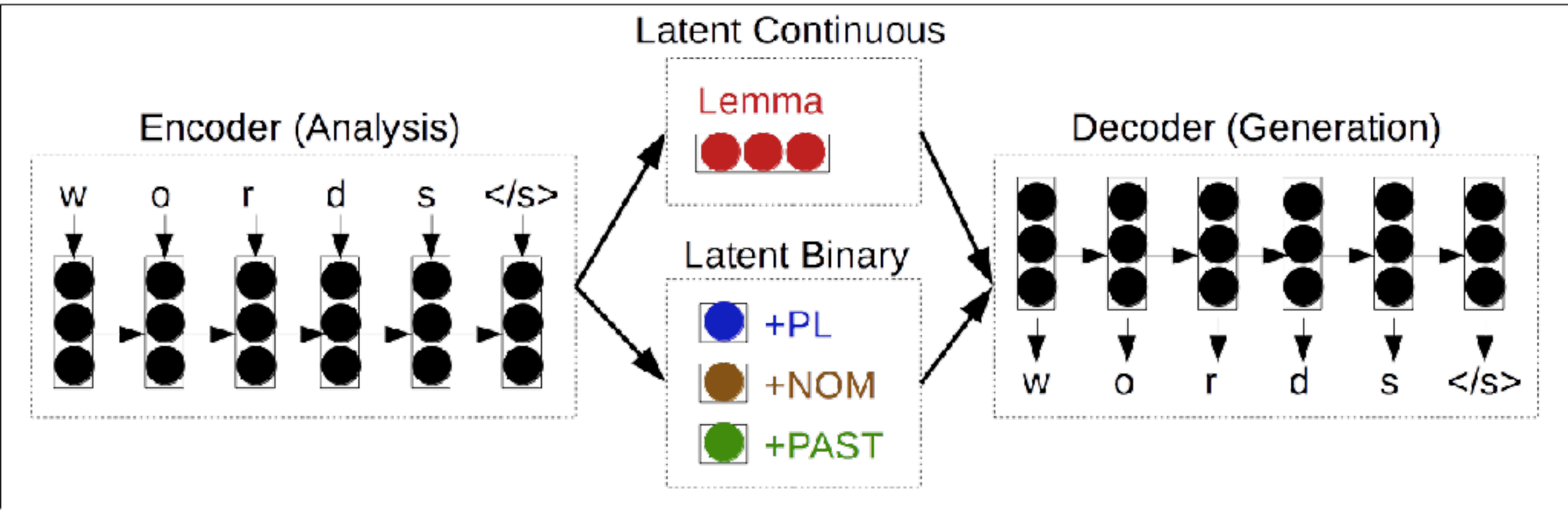
- When you say “Natural Language Generation” to an old-school NLPer, it means this

	SF Restaurant	SF Hotel
act type	inform, inform_only, reject, confirm, select, request, reqmore, goodbye	
shared	name, type, *pricerange, price, phone, address, postcode, *area, *near	
specific	*food *goodformeal ***kids-allowed**	***hasinternet** ***acceptscards** ***dogs-allowed**

bold=binary slots, *=slots can take “don’t care” value

From Input + Labels
(e.g. Zhou and Neubig 2017)

- For example, word + morphological tags -> inflected word



- Other options: politeness/gender in translation, etc.

From Images
(e.g. Karpathy et al. 2015)

- Input is image features, output is text

training image

"A Tabby cat is leaning on a wooden table, with one paw on a laser mouse and the other on a black laptop"

Other Auxiliary Information

- Name of a recipe + ingredients -> recipe (Kiddon et al. 2016)
- TED talk description -> TED talk (Hoang et al. 2016)
- etc. etc.

Questions?