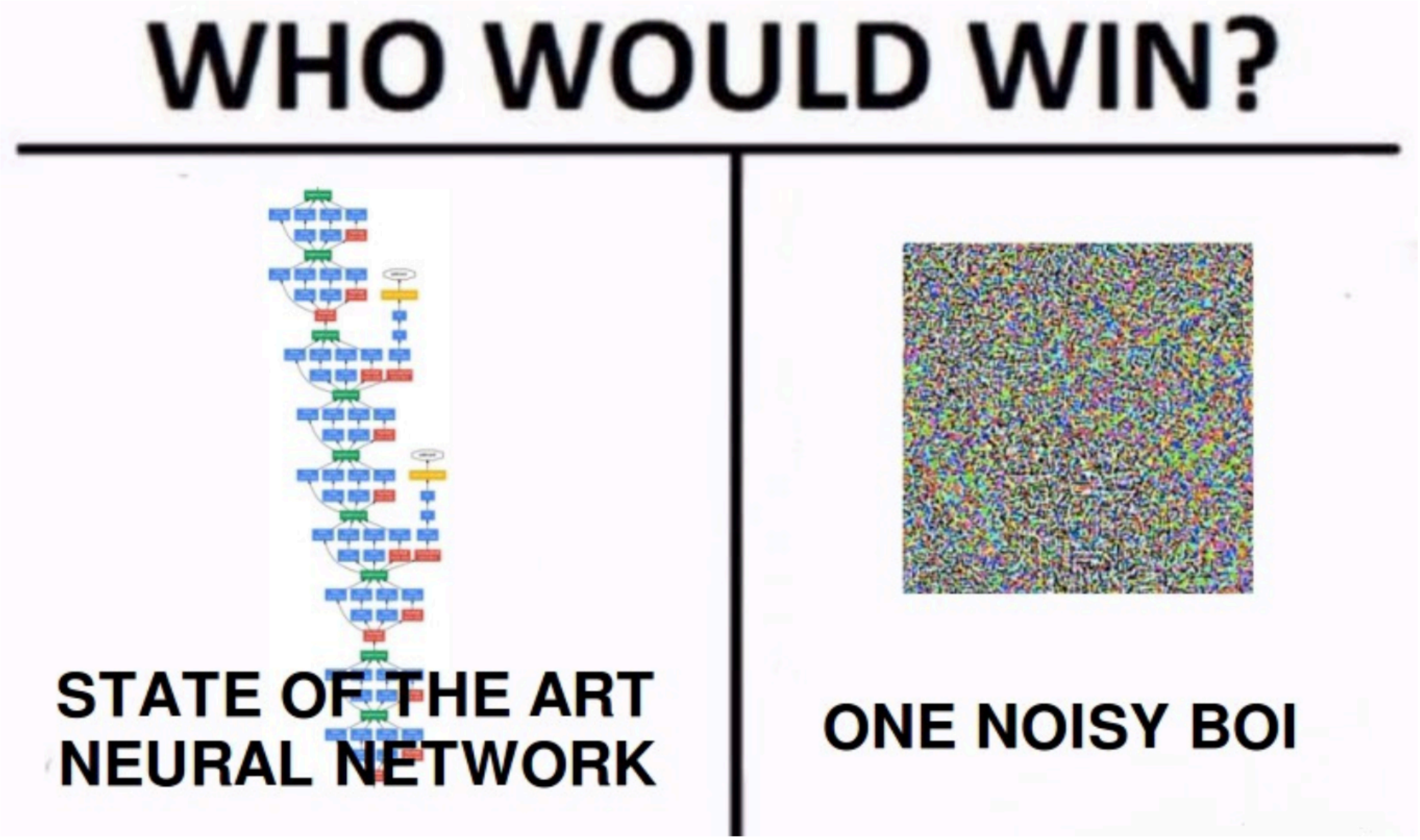
CS388: Natural Language Processing

Lecture 10: Interpreting NNs, Neural CRFs

Greg Durrett

TEXAS
The University of Texas at Austin

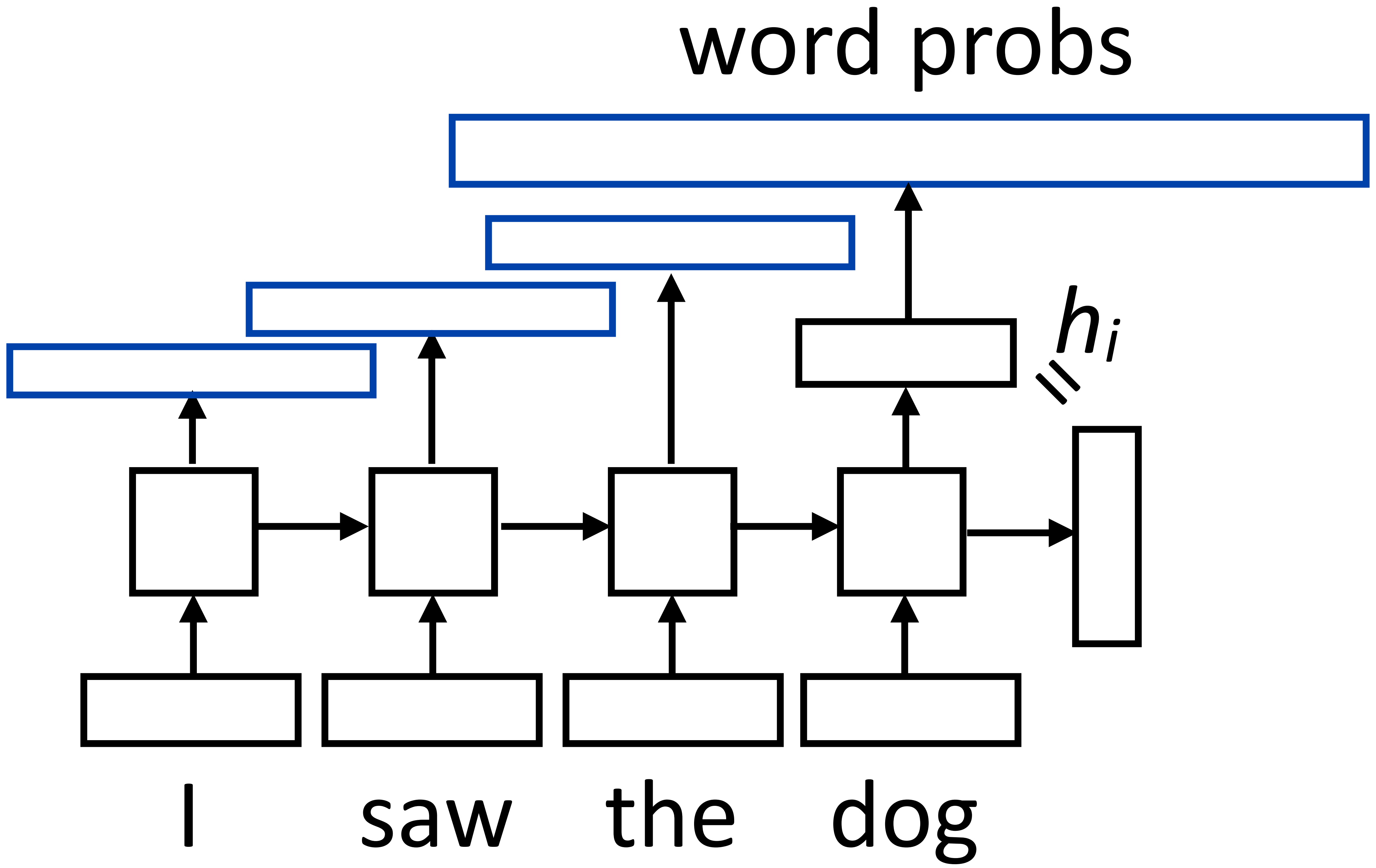


credit: Daniel Geng and Rishi Veerapaneni, ML @ Berkeley

Administrivia

- Mini 2 due in one week

Recall: RNNLMs



$$P(w|\text{context}) = \text{softmax}(W\mathbf{h}_i)$$

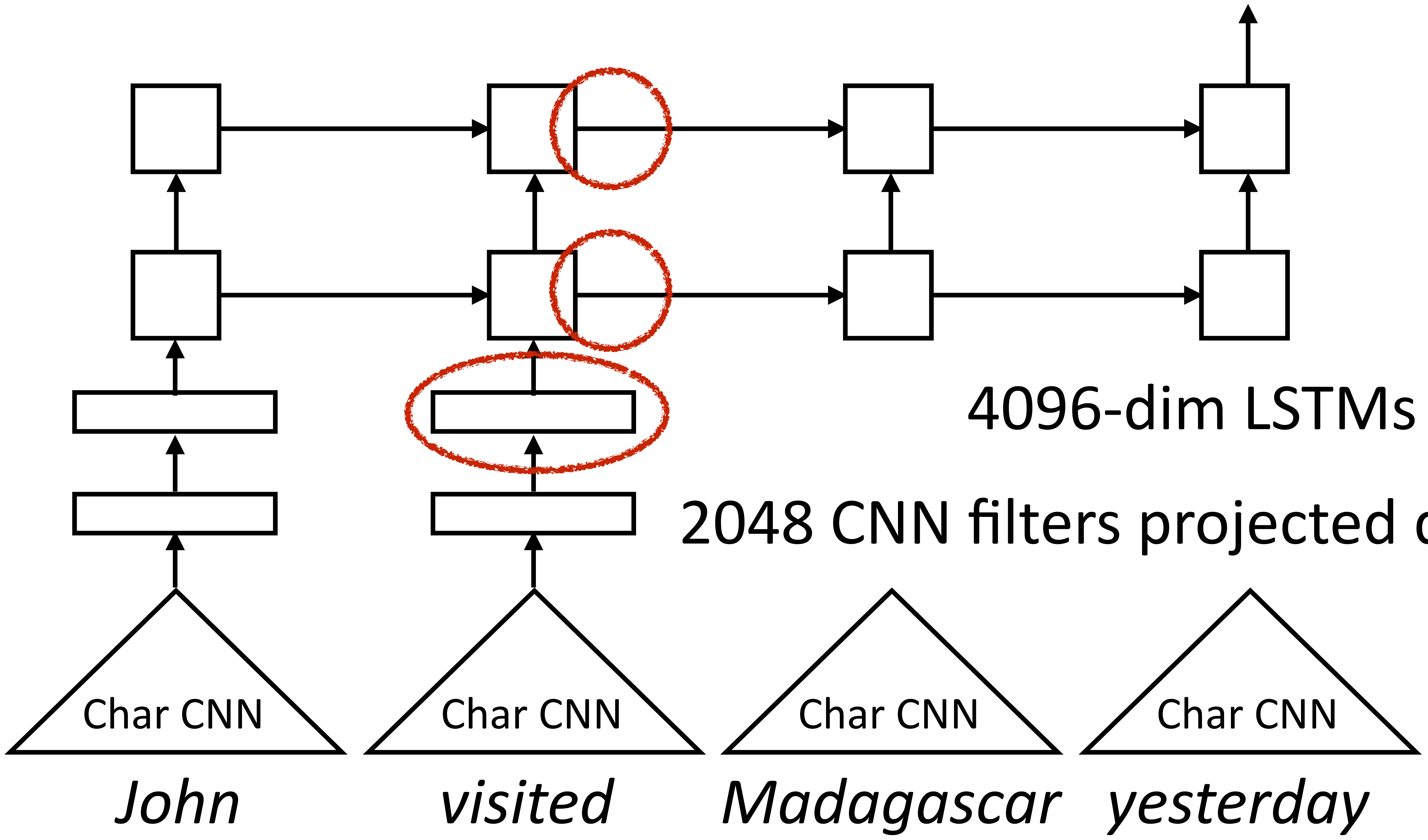
- *W* is a (vocab size) x (hidden size) matrix

- Backpropagate through the network to simultaneously learn to predict next word given previous words at all positions
- Batch by grabbing many contiguous sequences of text from different parts of a large corpus

Recall: ELMo

- CNN over each word => RNN

next word



Representation of *visited* (plus vectors from backwards LM)

4096-dim LSTMs w/ 512-dim projections

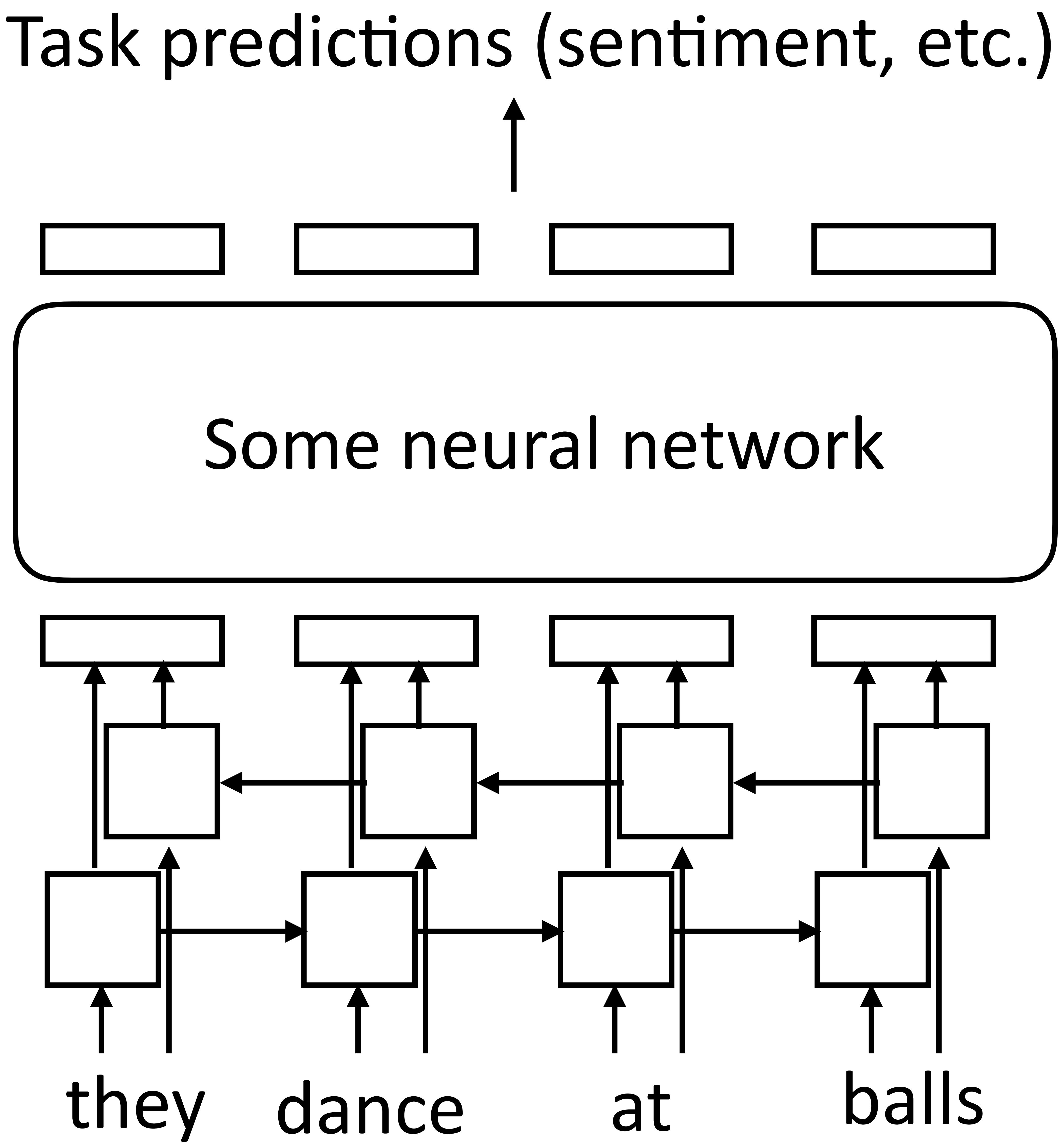
2048 CNN filters projected down to 512-dim

Peters et al. (2018)

Recall: ELMo

- Take those embeddings and feed them into whatever architecture you want to use for your task

- For ELMo, best to use *frozen* embeddings: update the weights of your network but keep ELMo's parameters frozen



Peters, Ruder, Smith (2019)

This Lecture

- Explaining neural networks' predictions

- Neural CRFs

Explaining NNs

What is an Explanation?

- Given a data instance, identify properties of the input/model that led to a particular decision being made

the movie was great features = (I[*great*], I[*the*])

- Suppose weight = (+5, +0), decision = +. what's the explanation?
- Suppose weight = (+5, +3), what's the explanation?
- Suppose weight = (+0.1, +5), what's the explanation?
- Explanation != "what a human would do". So any analysis of explanations has to intrinsically be about our model

Idea 1: Looking at Weights

- Is the maximum weight always right?

that movie was not great , in fact it was terrible !

- Feats = unigrams and bigrams
  w(*not great*) = -5, w(*great*) = +5, w(*terrible*) = -3
- Classified as negative; what's the explanation?
- *not great* and *great* cancel, don't really contribute to the classification decision. Correlated features make explanations confusing
- How can we define this? Deleting *great* would probably have little effect on the classification score

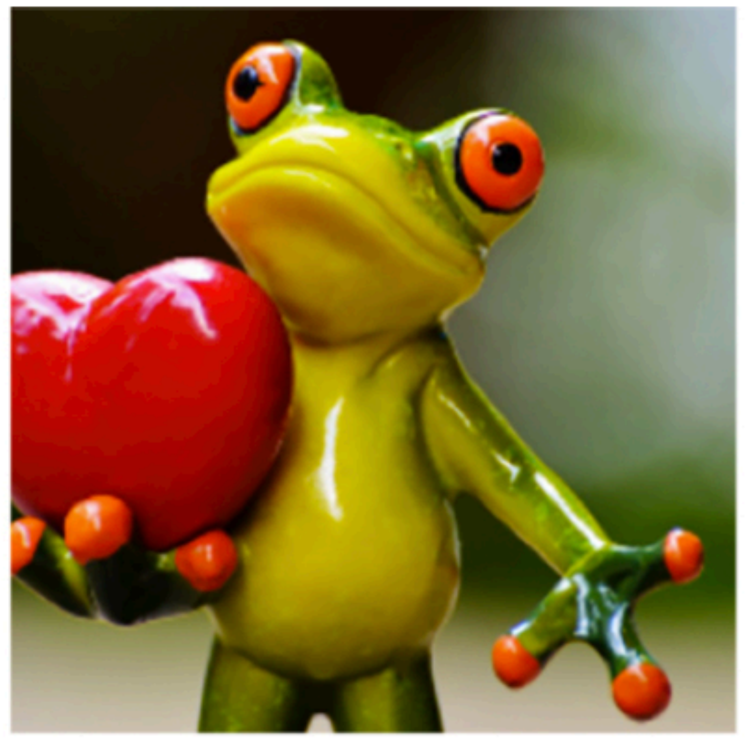
Idea 2: Counterfactuals

	Model
that movie was not great , in fact it was terrible !	—
that movie was not ____ , in fact it was terrible !	—
that movie was not great , in fact it was _____ !	+

- Perturb input many times and assess the impact on the model's prediction
- LIME: Locally-Interpretable Model-Agnostic Explanations
  - *Local* because we'll do work to learn how to interpret this one example
  - *Model-agnostic*: treat model as black box

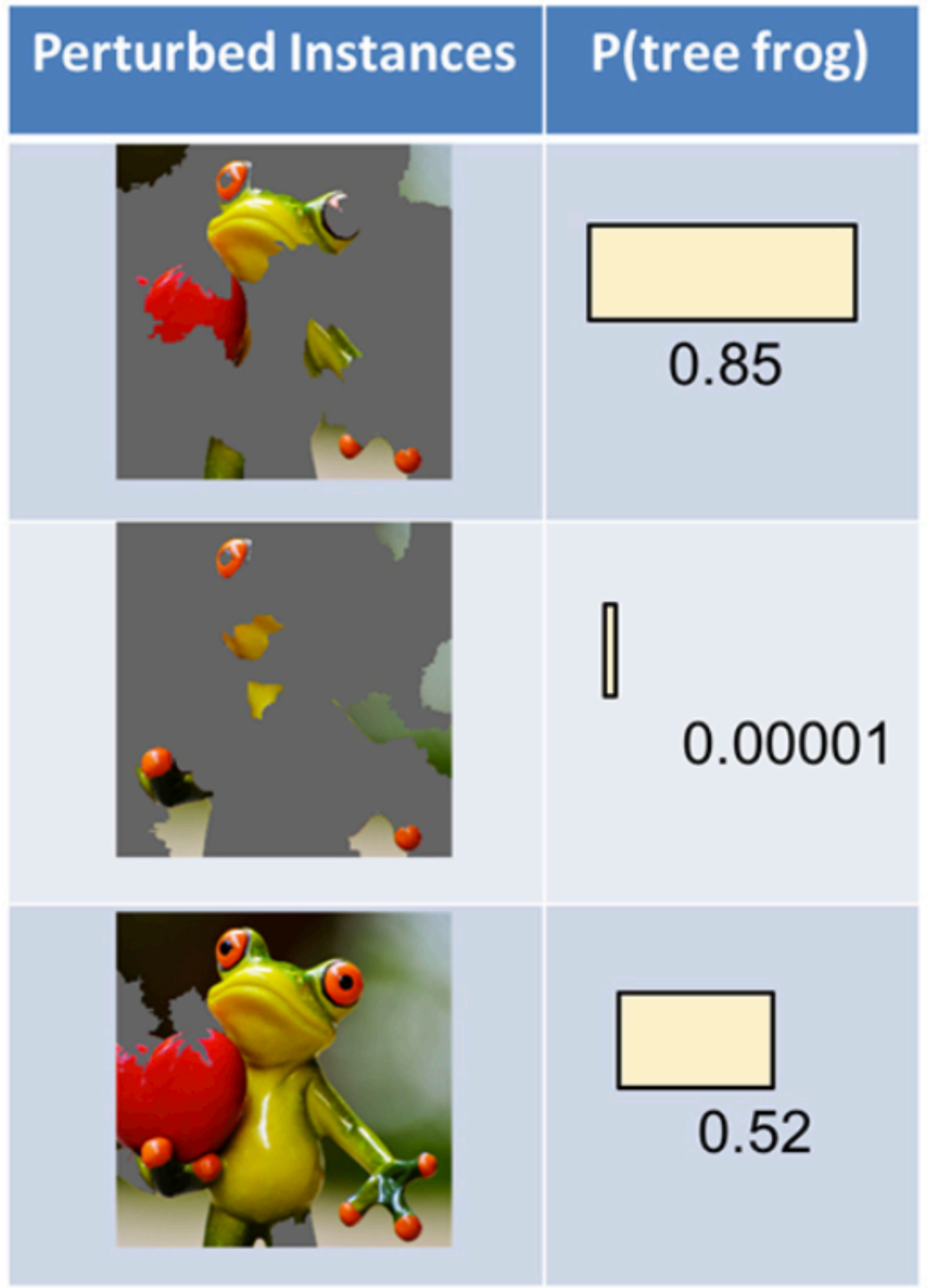
Ribeiro et al. (2016)

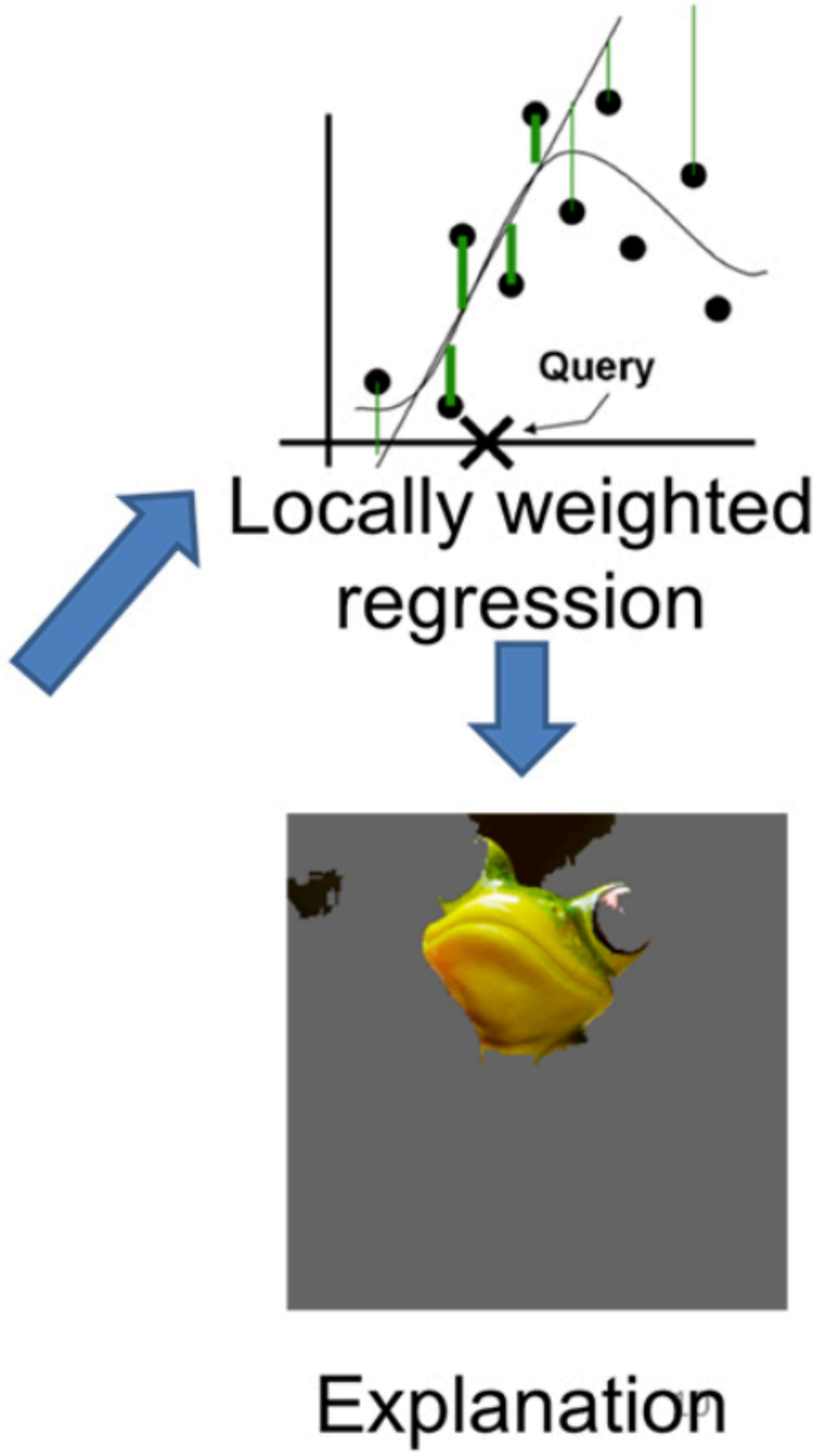
LIME



Original Image

Interpretable Components





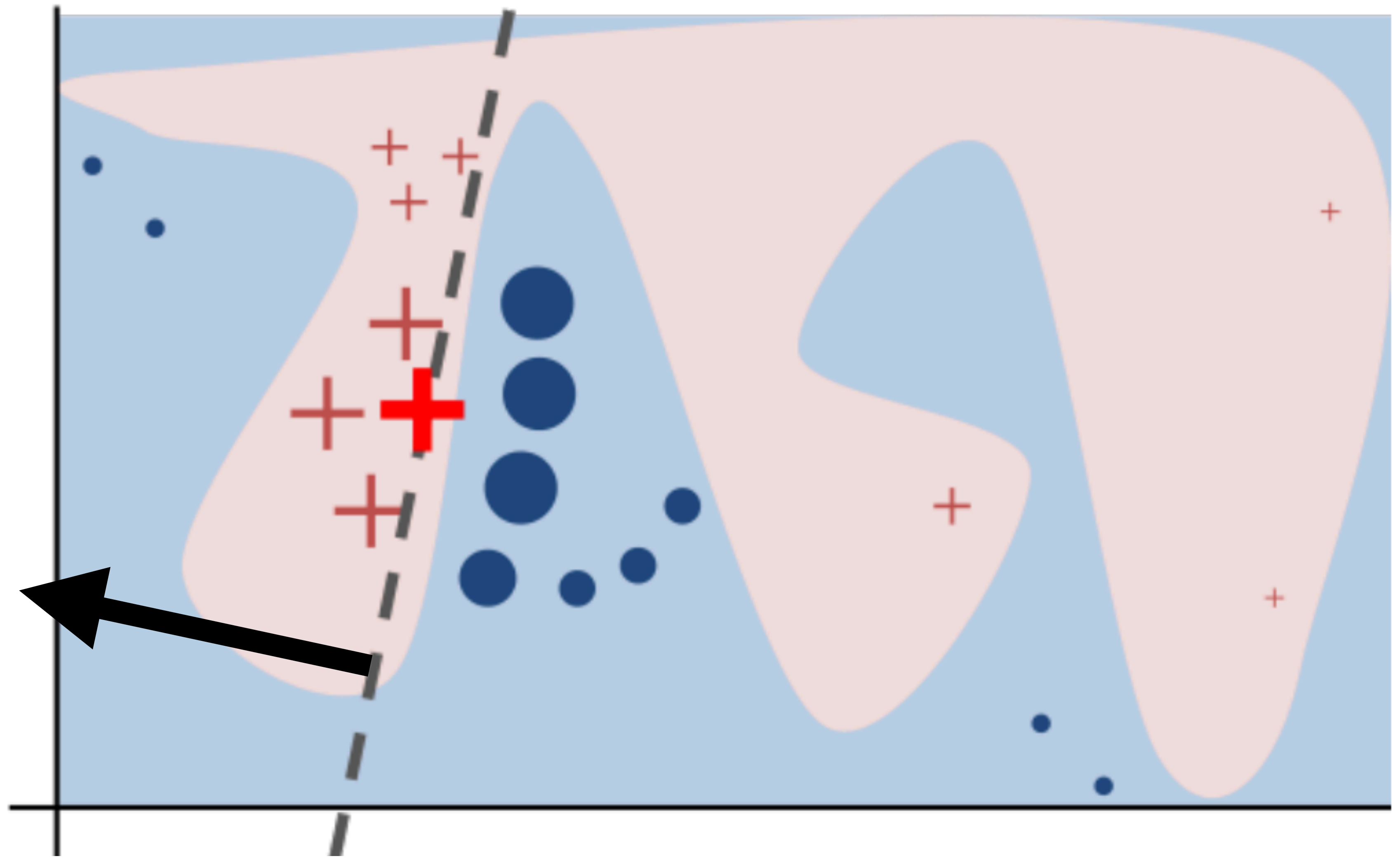
- Break input into components (for text classification: unigrams)
- Check predictions on subsets of those
- Train a model to predict predictions, look at that model's weights

https://www.oreilly.com/learning/introduction-to-local-interpretable-model-agnostic-explanations-lime

LIME

- Break down input into many small pieces so the explanation is interpretable
  $x \in \mathbb{R}^d \rightarrow x' \in \{0,1\}^{d'}$
- Draw samples z' by perturbing x', then reconstruct z from z' and compute f(z) on that
- Now learn a model to predict f(z) based on z'. This model's weights will serve as the explanation for the decision

why it's +



- If z' is very coarse, can interpret but can't learn a good model of the boundary. If z' is too fine-grained, can interpret but not predict (e.g., z' = z)

Ribeiro et al. (2016)

LIME

Algorithm 1 Sparse Linear Explanations using LIME

Require: Classifier f, Number of samples N
Require: Instance x, and its interpretable version x'
Require: Similarity kernel π_x, Length of explanation K

$\mathcal{Z} \leftarrow \{\}$
for $i \in \{1, 2, 3, ..., N\}$ **do**
    $z'_i \leftarrow sample\_around(x')$
    $\mathcal{Z} \leftarrow \mathcal{Z} \cup \langle z'_i, f(z_i), \pi_x(z_i)\rangle$
end for
$w \leftarrow$ K-Lasso$(\mathcal{Z}, K)$ $\triangleright$ with z'_i as features, $f(z)$ as target
return w

- Use a sparse linear model to achieve a sparse explanation

Ribeiro et al. (2016)

LIME

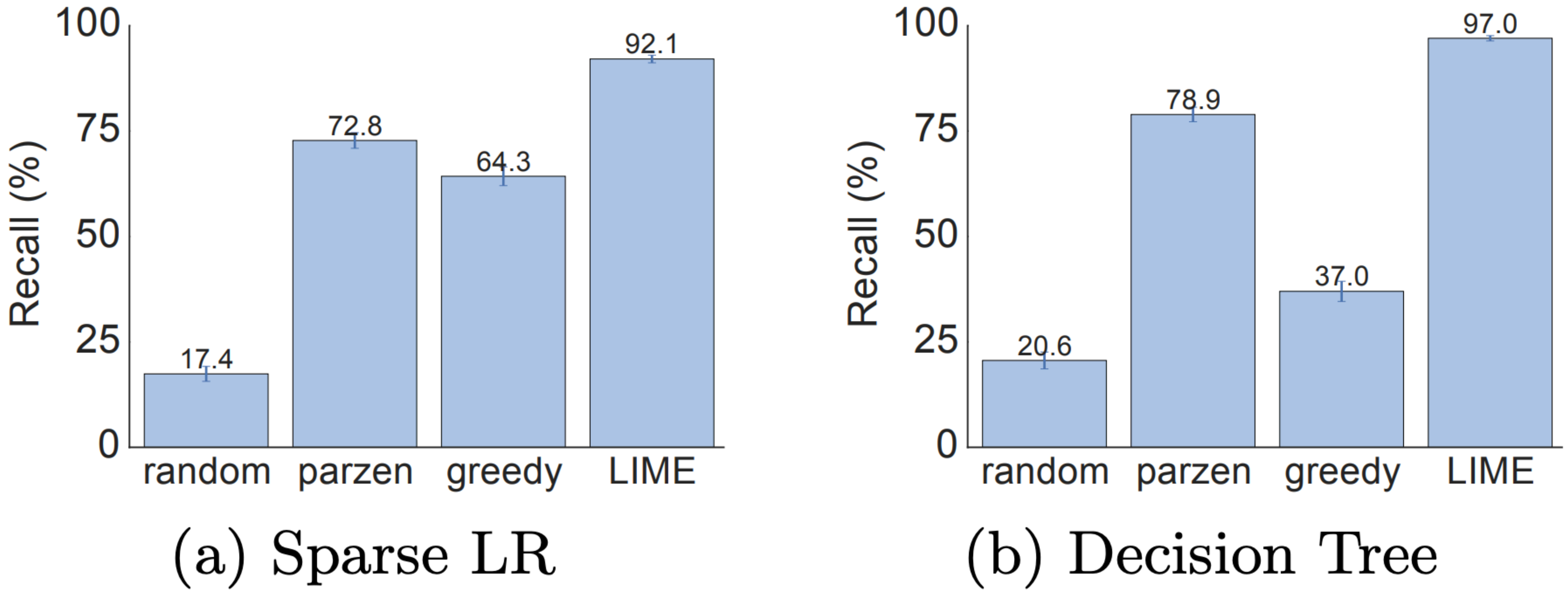
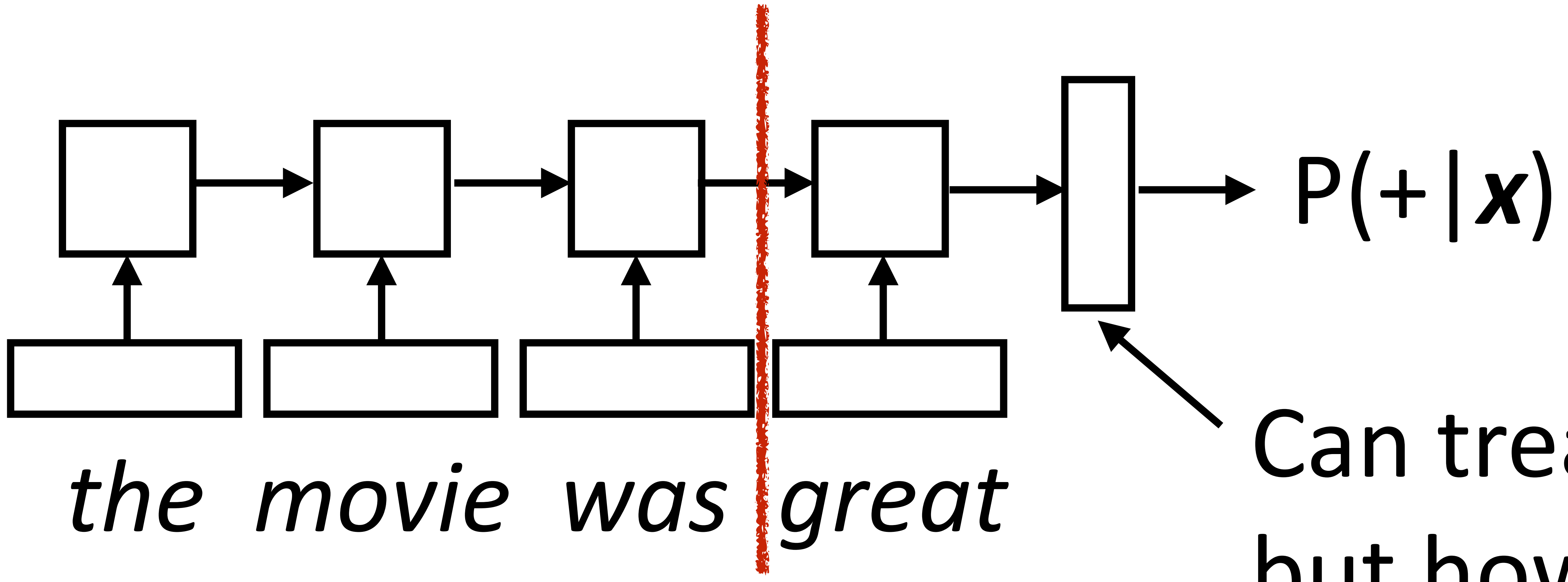


Figure 6: Recall on truly important features for two interpretable classifiers on the books dataset.

- Train a sparse model (only looks at 10 features of each example), then try to use LIME to recover the features. Greedy: remove features to make predicted class prob drop by as much as possible

Idea 3: Weights Revisited

- LIME is very complex, but looking at weights is too simple



Can treat this layer like a linear model, but how to connect it to input? Often hundreds of features

- Suppose forget gate is very low and the first three words are forgotten
- How can we generally assess impact of a word on the prediction?
- We don't have "weights", but what can tell us about the impact of the input on the output?

Gradient-Based Methods

S_c = score of class c

I_0 = current image

- Approximate score with a first-order Taylor series approximation around the current image

$$S_c(I) \approx w^T I + b$$

$$w = \left. \frac{\partial S_c}{\partial I} \right|_{I_0}$$

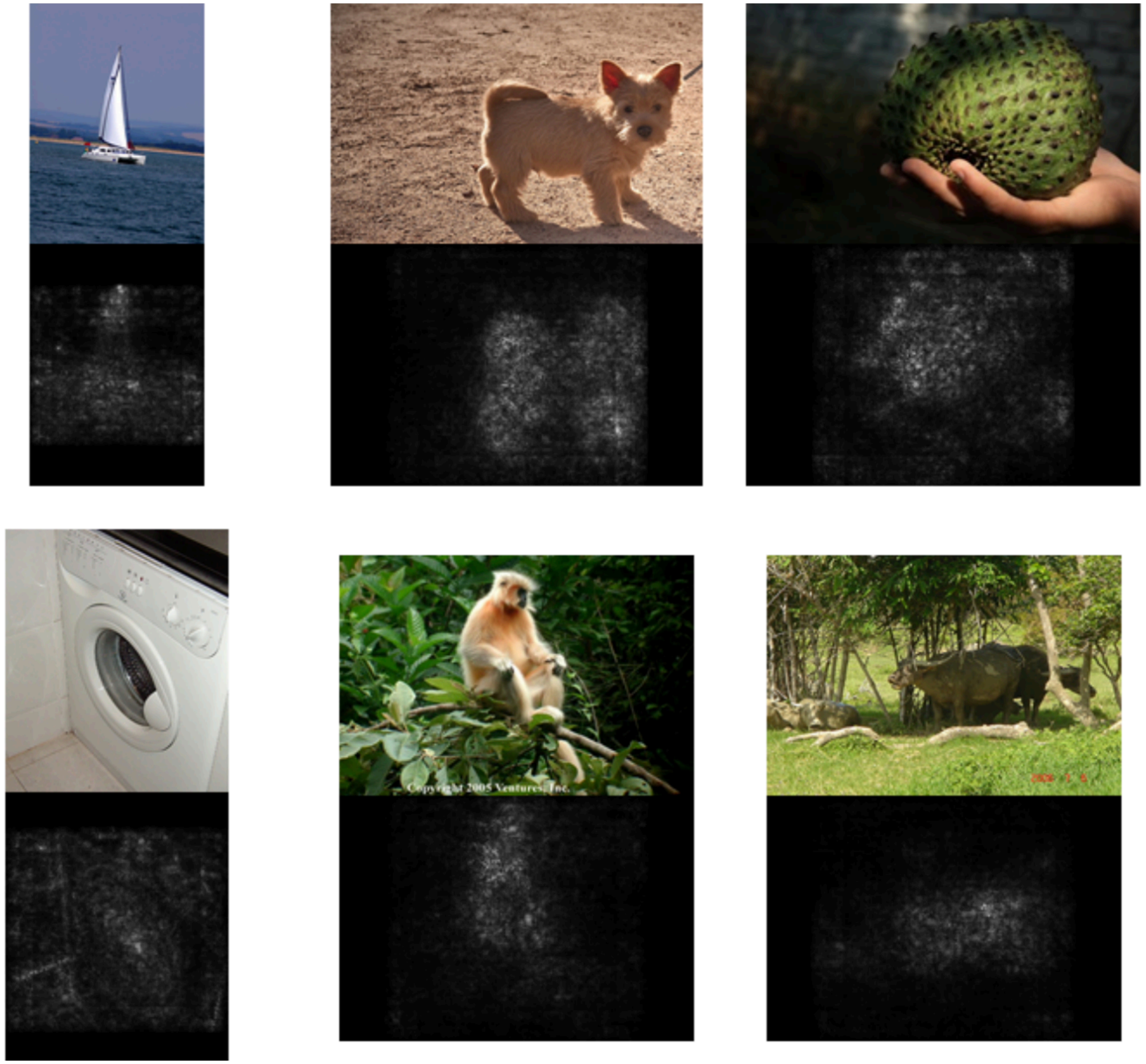
- Higher gradient magnitude = small change in pixels leads to large change in prediction

- To get single magnitude for a pixel, max over color channels. Can do the same for a word (max over vector positions)

- Sanity check: does this make sense for linear models?

Simonyan et al. (2013)

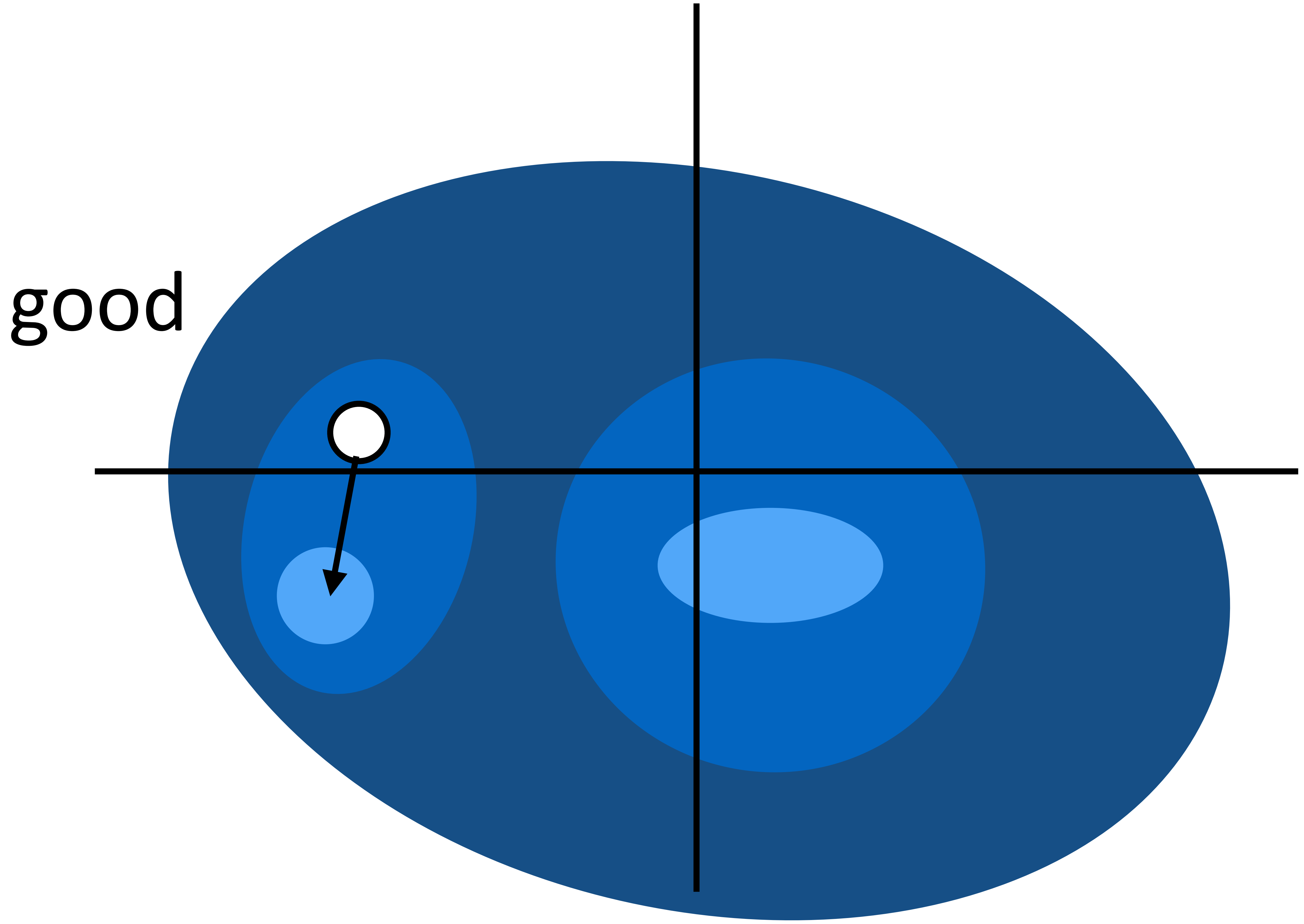
Gradient-Based Methods

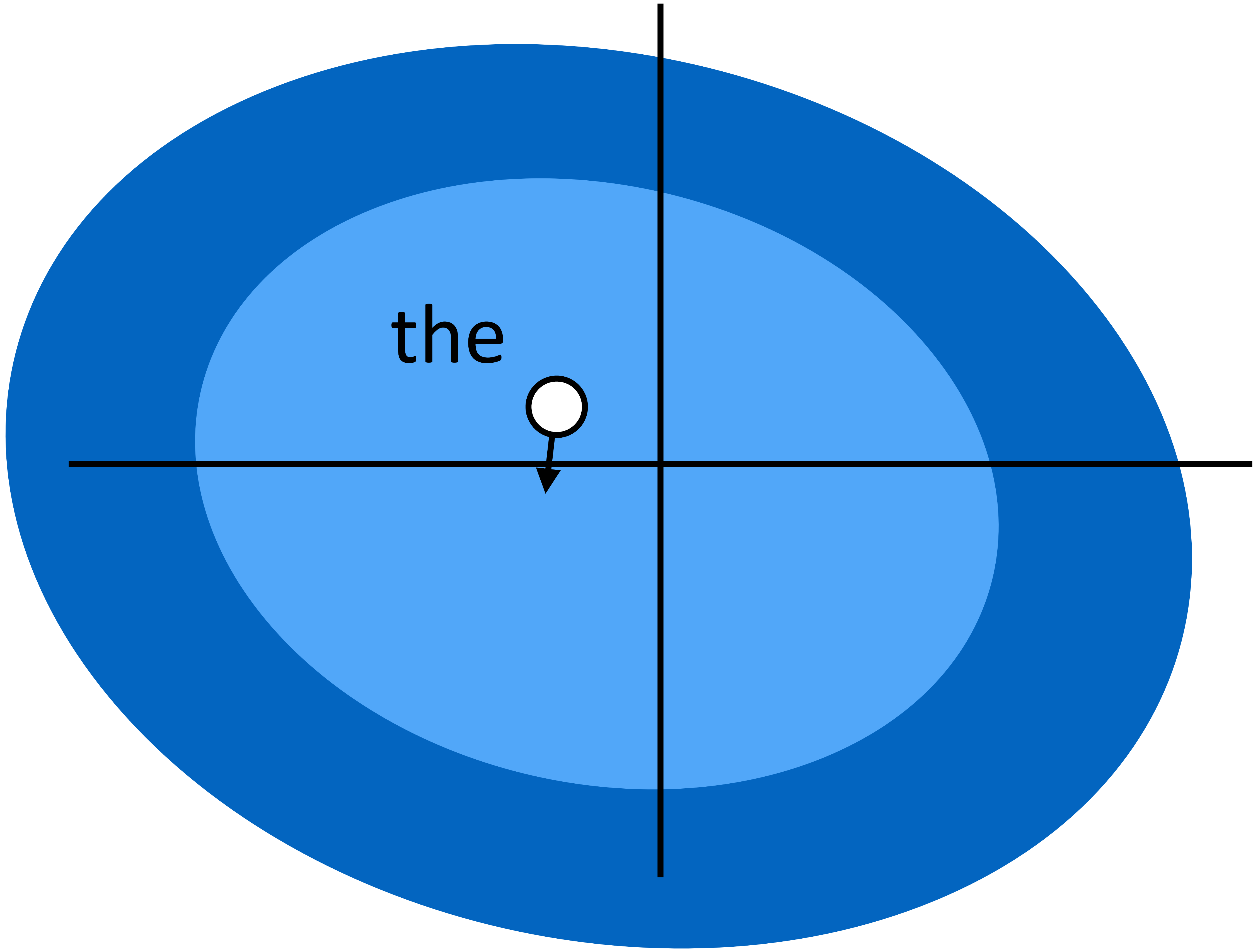


Simonyan et al. (2013)

Gradient-based Method

- axes = word vector values. Lighter color = higher positive class probability





- Changing the word makes a difference: seems like the word is having some impact
- Changing the word locally has little effect: this word doesn't matter much

Gradients vs. LIME

- Explanation methods should predict features which, when deleted, cause the prediction to flip
- 1) Rank all features with the method. 2) Delete features and see how long it takes to flip the decision

	20news		Movie	
	LR	**MLP**	**LR**	**MLP**
random	0.8617	0.8880	0.6586	0.6843
LIME-500	0.4394	0.5330	0.1747	0.1973
LIME-1000	0.3098	0.4164	0.0811	0.1034
LIME-1500	0.2607	0.3566	0.0613	0.0800
LIME-2000	0.2336	0.3235	0.0547	0.0743
LIME-5000	0.1895	0.2589	0.0474	0.0664
omission	**0.1595**	0.2662	**0.0449**	0.0644
saliency	-	**0.2228**	-	**0.0639**

Table 3: The % of words that needs to be deleted to change the prediction (the switching point).

- Omission: like the greedy algorithm from LIME comparison
- Saliency (gradient method) is better at finding the flip points than LIME (but only slightly)

Nguyen (2018)

Explaining Sequence Models

- These models might work well for bag-of-words models, but what about other tasks?

  I went to the store => Je suis allé au magasin

  I _____ to the store => ???

- Translation system might totally break down, need to stay on the data manifold
- Sample similar datapoints from a variational autoencoder (VAE), more complex approach that requires another model

Alvarez-Melis and Jaakkola (2019)

Idea 3: Probing

- Train a model for task X and learn to predict task Y

- E.g.: take ELMo representations, freeze them, then try to predict POS representations with just a softmax layer

Model	**Acc.**
Collobert et al. (2011)	97.3
Ma and Hovy (2016)	97.6
Ling et al. (2015)	**97.8**
CoVe, First Layer	*93.3*
CoVe, Second Layer	92.8
biLM, First Layer	*97.3*
biLM, Second Layer	96.8

- Doesn't "explain" a prediction but can illuminate what models are and aren't able to capture

Takeaways

- Looking at weights is generally hard for neural networks
- LIME is a good method for generating interpretable explanations, but not always easy to get right
- Gradient-based techniques can provide explanations, but these aren't perfect. Very "local" and don't consider what happens if a word changes to a different word
- Probing tasks can tell you generally what your network might be doing but are hard to interpret

Neural CRF Basics

NER Revisited

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG

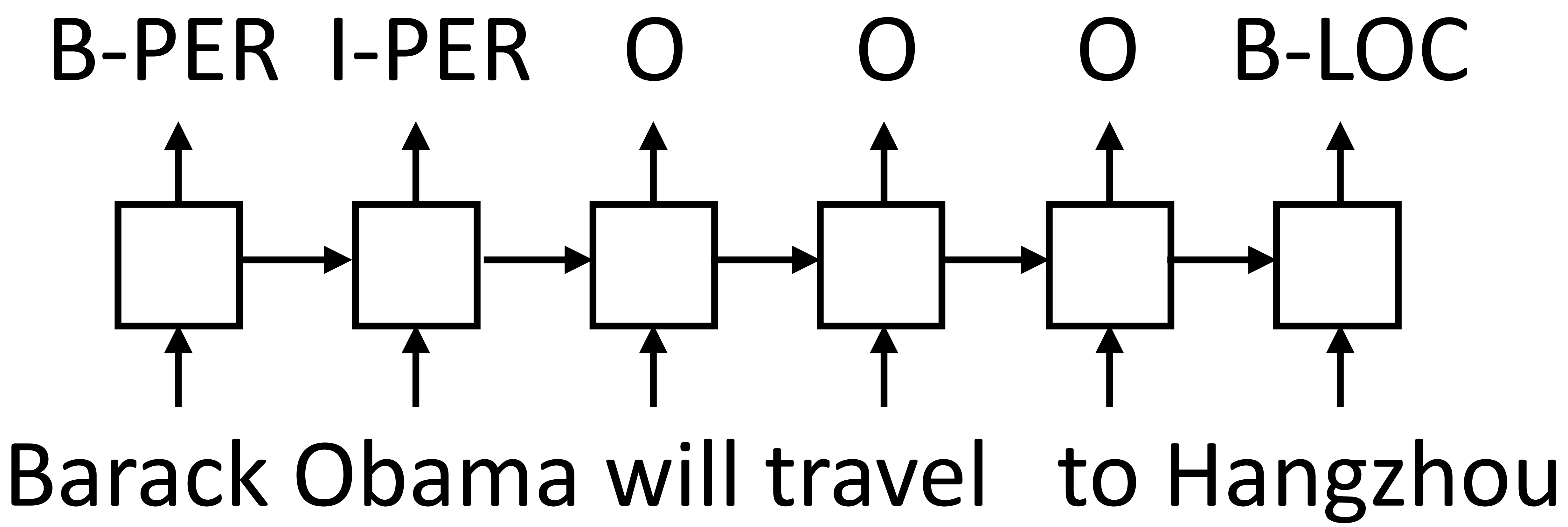
- Features in CRFs: I[tag=B-LOC & curr_word=*Hangzhou*], I[tag=B-LOC & prev_word=*to*], I[tag=B-LOC & curr_prefix=*Han*]
- Linear model over features
- Downsides:
  - Lexical features mean that words need to be seen in the training data
  - Linear model can't capture feature conjunctions as effectively (doesn't work well to look at more than 2 words with a single feature)

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



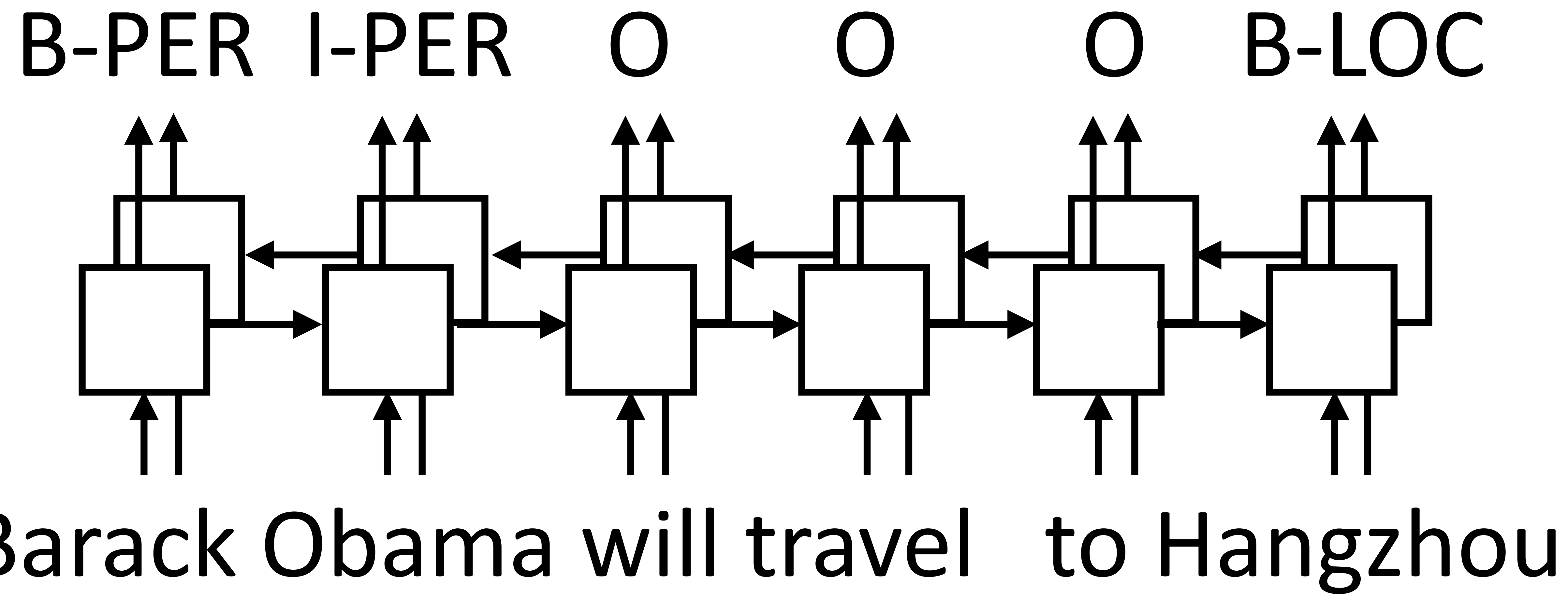
- Transducer (LM-like model)
- What are the strengths and weaknesses of this model compared to CRFs?

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



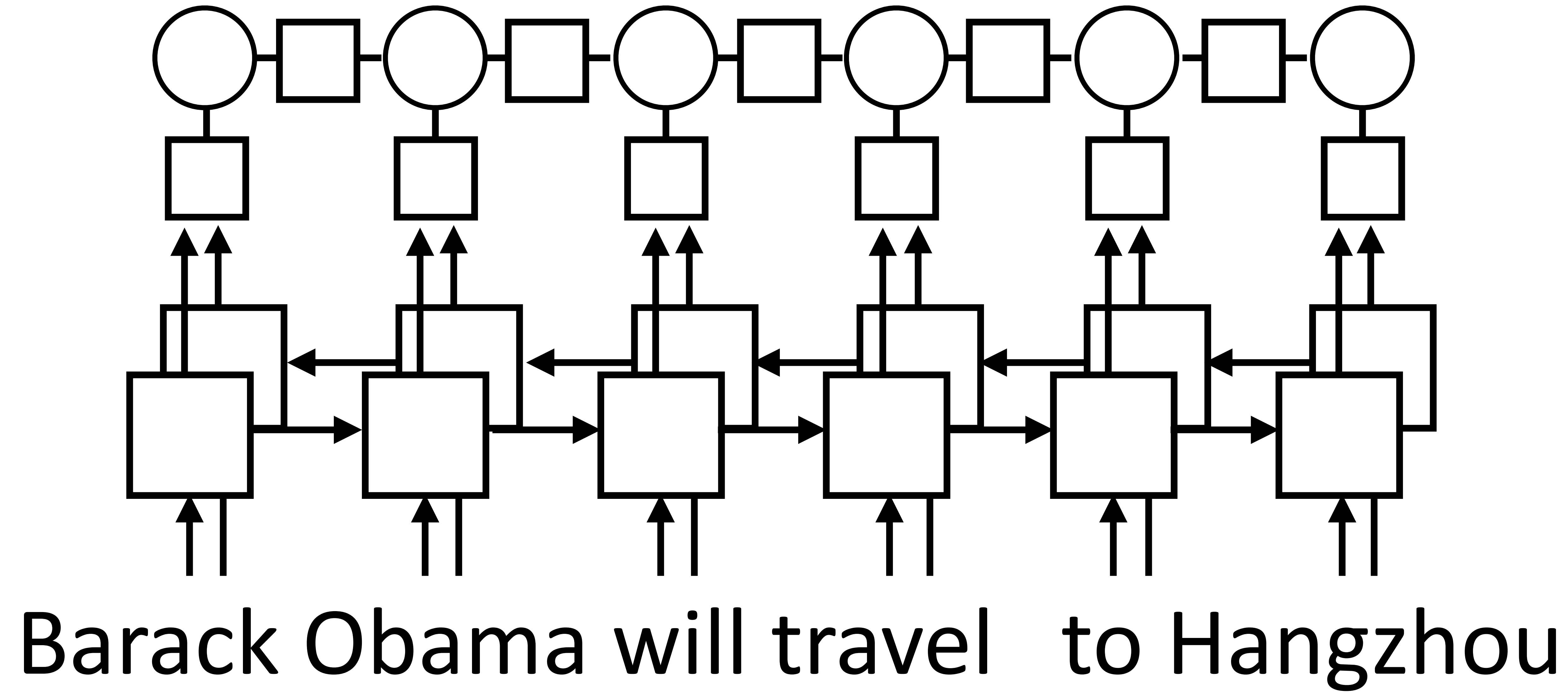
- Bidirectional transducer model
- What are the strengths and weaknesses of this model compared to CRFs?

Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

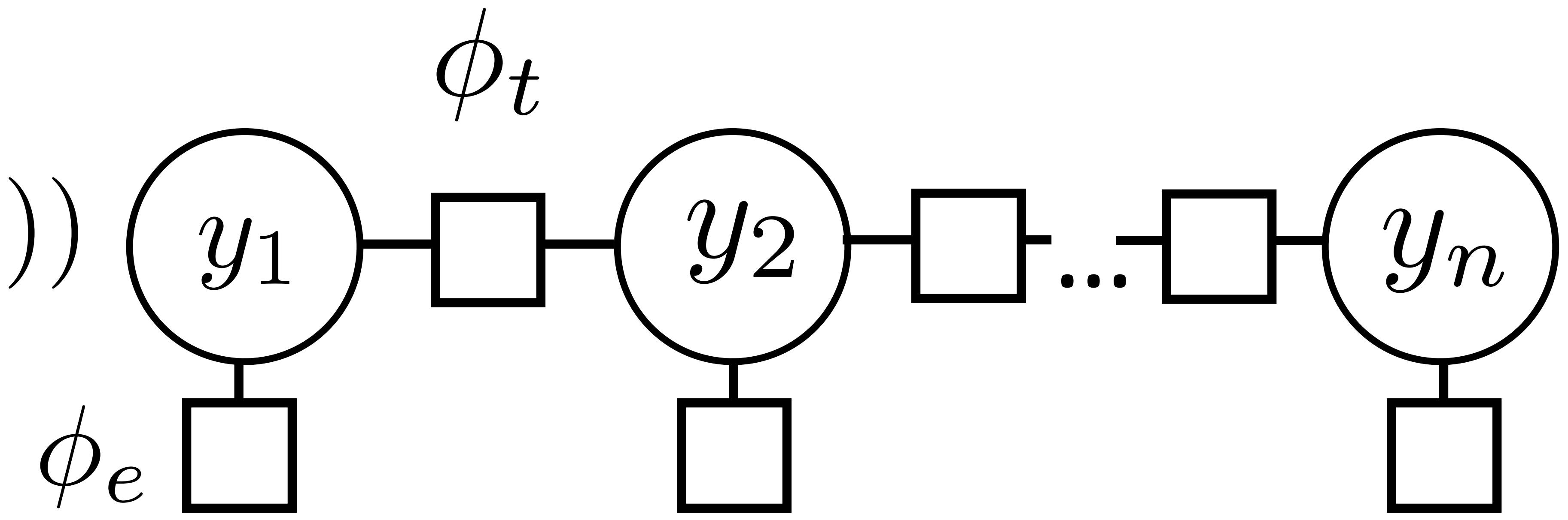
PERSON LOC ORG



- Neural CRFs: bidirectional LSTMs (or some NN) compute emission potentials, capture structural constraints in transition potentials

Neural CRFs

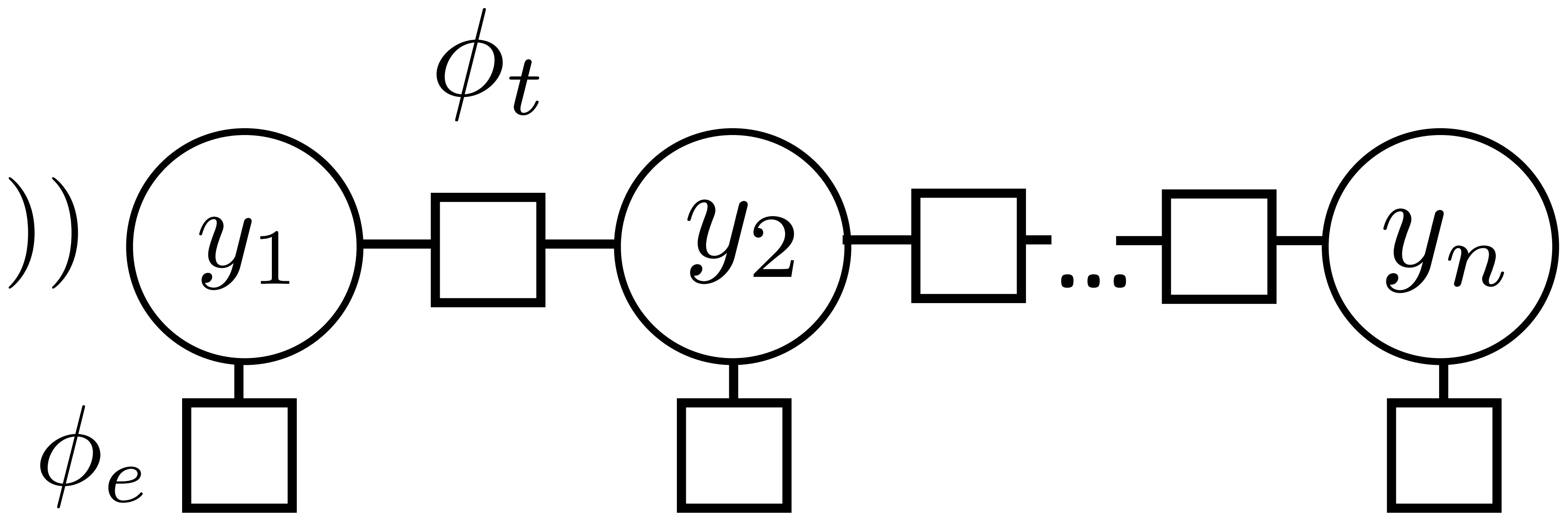
$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$
- Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$ *W* is a num_tags x len(*f*) matrix
- *f*(*i*, ***x***) could be the output of a feedforward neural network looking at the words around position *i*, or the *i*th output of an LSTM, …
- Neural network computes unnormalized potentials that are consumed and "normalized" by a structured model
- Inference: compute *f*, use Viterbi

Computing Gradients

$$P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\prod_{i=2}^{n}\exp(\phi_t(y_{i-1}, y_i))\prod_{i=1}^{n}\exp(\phi_e(y_i, i, \mathbf{x}))$$



- Conventional: $\phi_e(y_i, i, \mathbf{x}) = w^\top f_e(y_i, i, \mathbf{x})$

- Neural: $\phi_e(y_i, i, \mathbf{x}) = W_{y_i}^\top f(i, \mathbf{x})$

$$\frac{\partial \mathcal{L}}{\partial \phi_{e,i}} = -P(y_i = s|\mathbf{x}) + I[s \text{ is gold}]$$

"error signal", compute with F-B

chain rule say to multiply together, gives our update

- For linear model: $\frac{\partial \phi_{e,i}}{w_i} = f_{e,i}(y_i, i, \mathbf{x})$

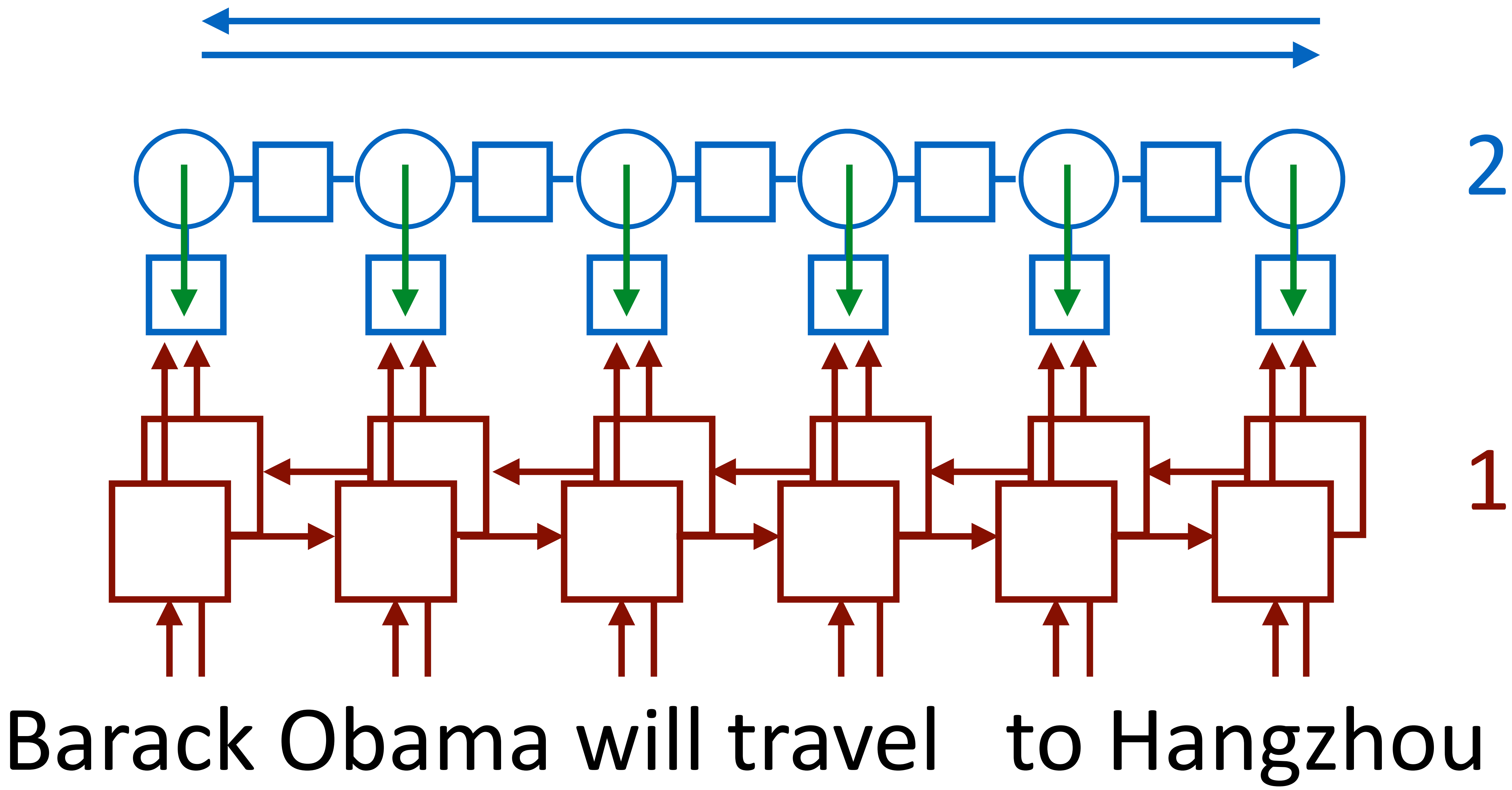
- For neural model: compute gradient of phi w.r.t. parameters of neural net

Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



2) Run forward-backward

3) Compute error signal

1) Compute f(**x**)

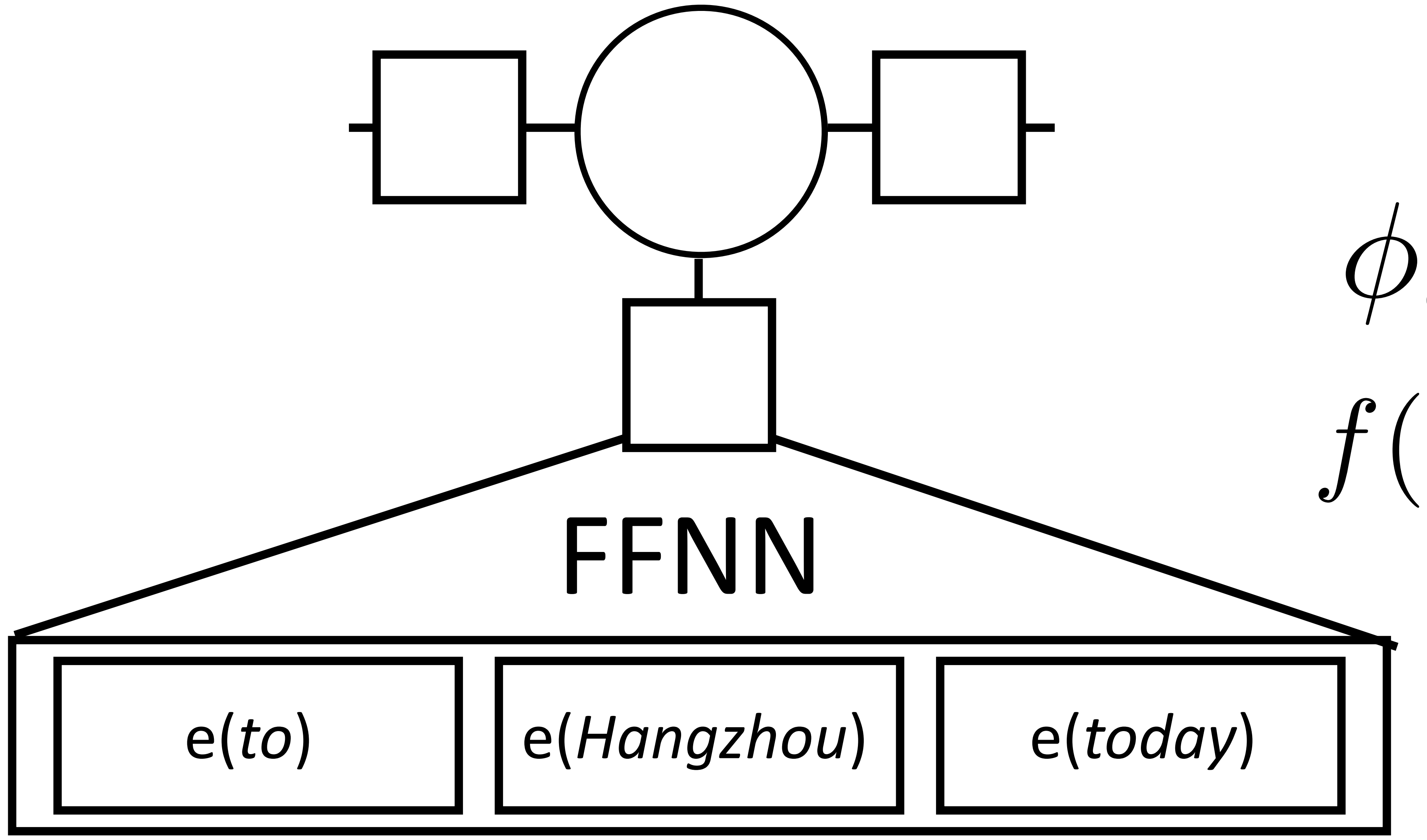
4) Backprop (no knowledge of CRF structure required)

FFNN Neural CRF for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



previous word curr word next word

to ***Hangzhou*** *today*

$$\phi_e = Wg(Vf(\mathbf{x}, i))$$

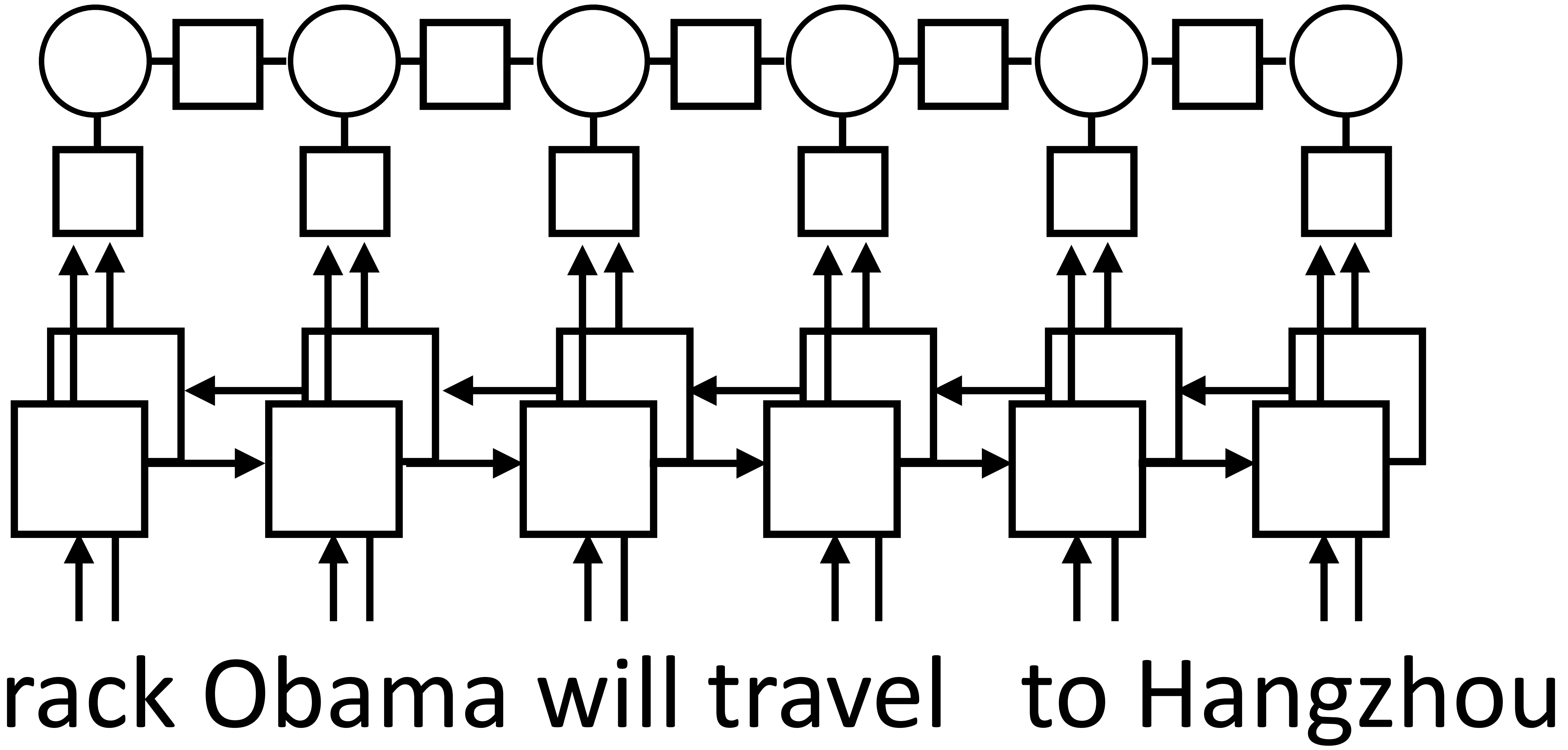
$$f(\mathbf{x}, i) = [\mathrm{emb}(\mathbf{x}_{i-1}), \mathrm{emb}(\mathbf{x}_i), \mathrm{emb}(\mathbf{x}_{i+1})]$$

LSTM Neural CRFs

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

PERSON LOC ORG



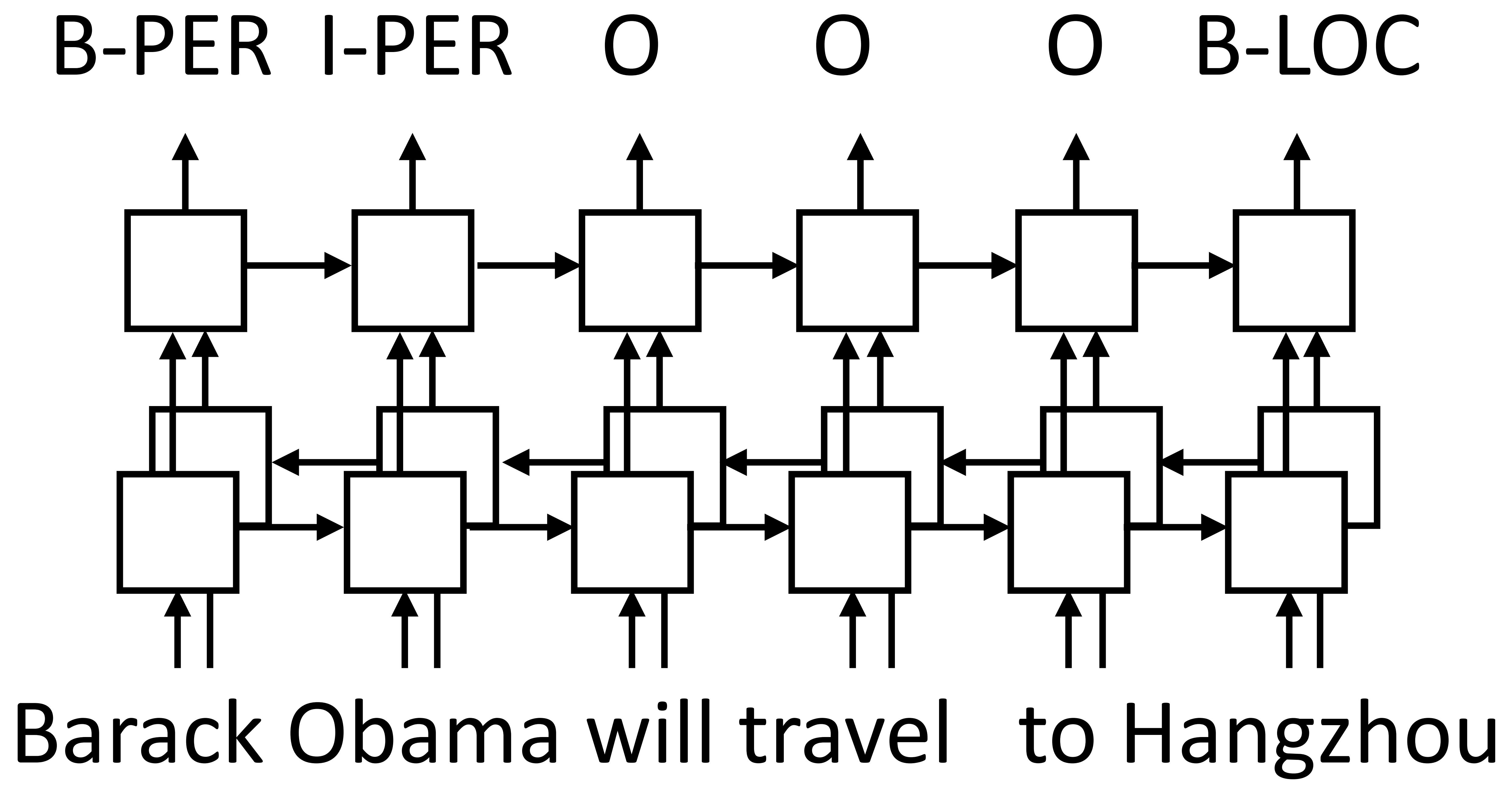
- Bidirectional LSTMs compute emission (or transition) potentials

LSTMs for NER

B-PER I-PER O O O B-LOC O O O B-ORG O O

Barack Obama will travel to Hangzhou today for the G20 meeting .

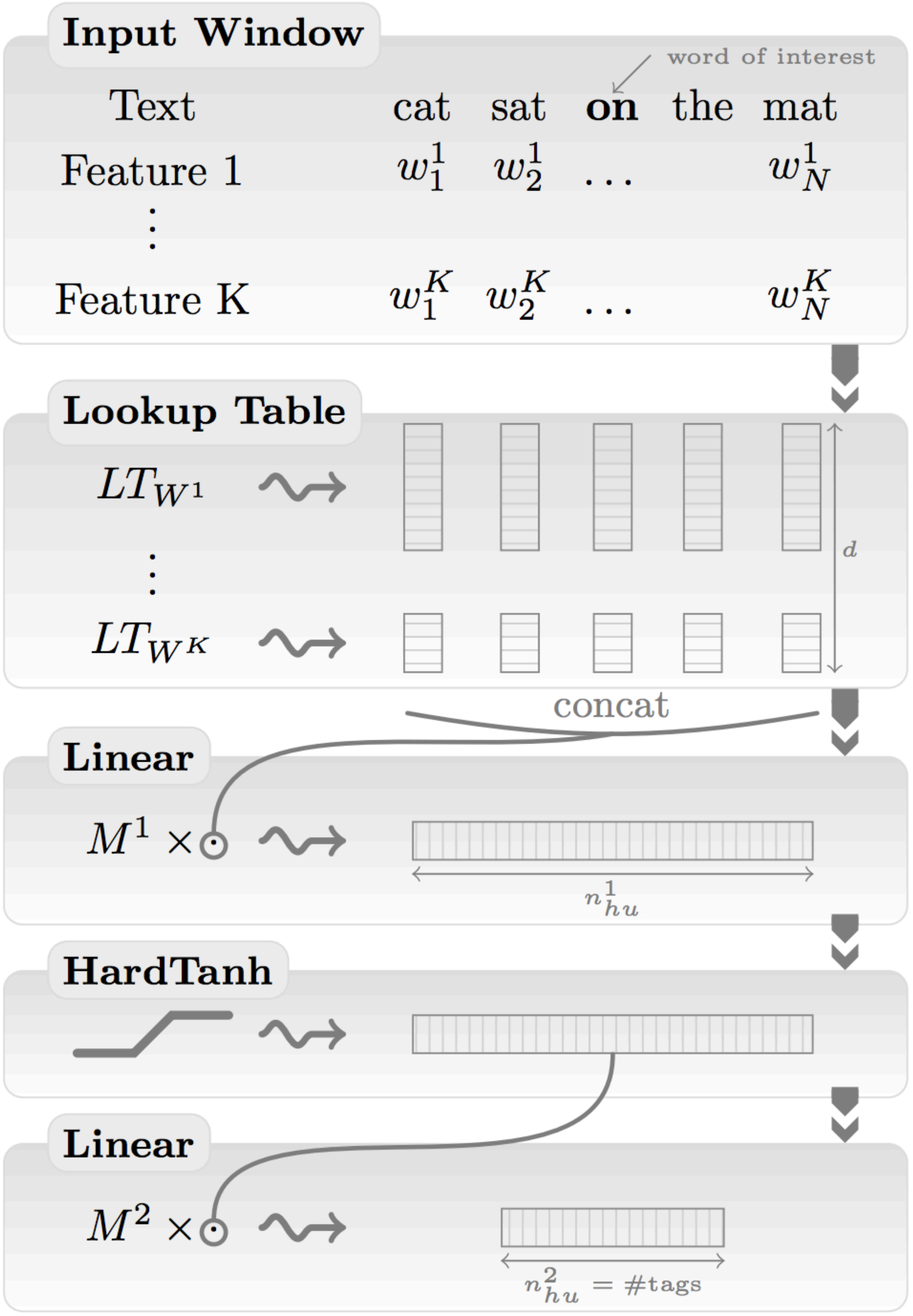
PERSON LOC ORG



- How does this compare to neural CRF?

"NLP (Almost) From Scratch"

Approach	POS (PWA)	CHUNK (F1)	NER (F1)	SRL (F1)
Benchmark Systems	97.24	94.29	89.31	77.92
NN+WLL	96.31	89.13	79.53	55.40
NN+SLL	96.37	90.33	81.47	70.99
NN+WLL+LM1	97.05	91.91	85.68	58.18
NN+SLL+LM1	97.10	93.65	87.58	73.84
NN+WLL+LM2	97.14	92.04	86.96	58.34
NN+SLL+LM2	97.20	93.63	88.67	74.15

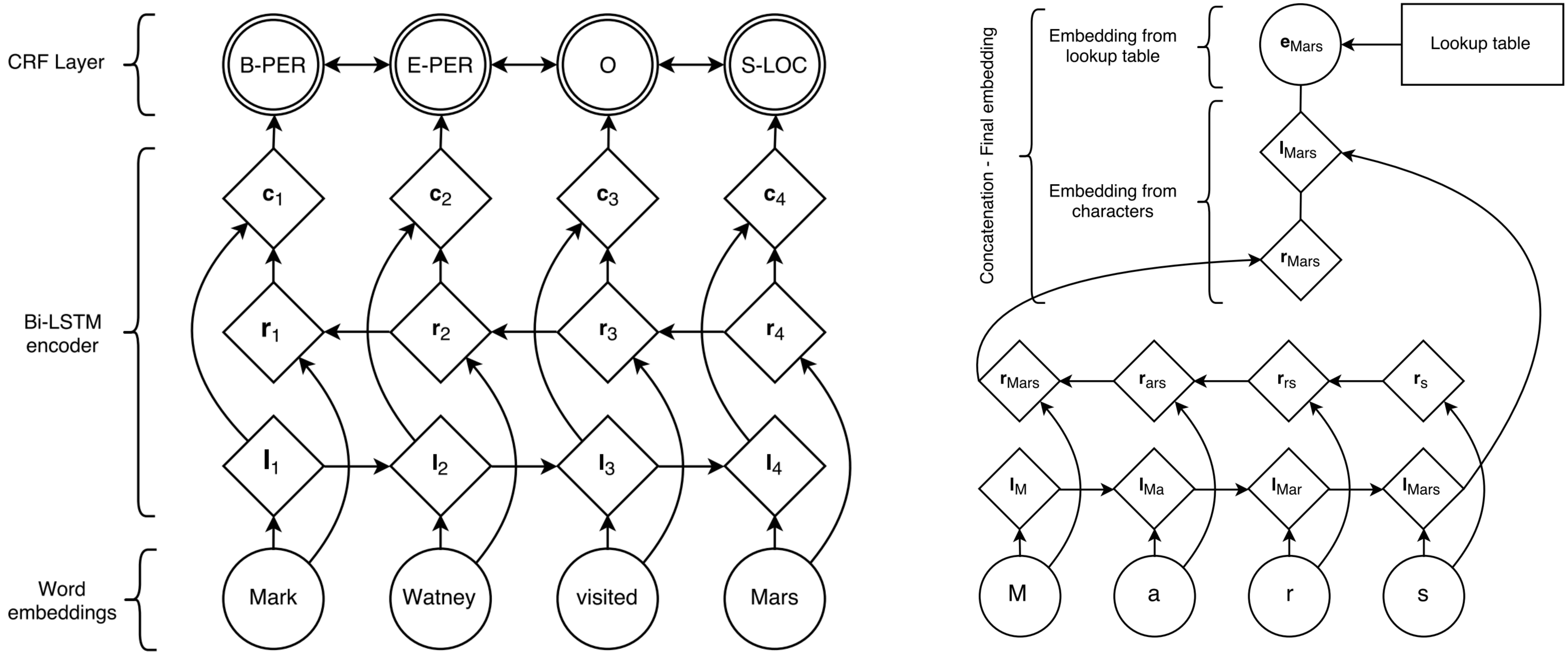


- WLL: independent classification; SLL: neural CRF
- LM2: word vectors learned from a precursor to word2vec/GloVe, trained for 2 weeks (!) on Wikipedia

Collobert, Weston, et al. 2008, 2011

Neural CRFs with LSTMs

- Neural CRF using character LSTMs to compute word representations



Chiu and Nichols (2015), Lample et al. (2016)

Neural CRFs with LSTMs

- Chiu+Nichols: character CNNs instead of LSTMs
- Lin/Passos/Luo: use external resources like Wikipedia
- LSTM-CRF captures the important aspects of NER: word context (LSTM), sub-word features (character LSTMs), outside knowledge (word embeddings)

Model	**F_1**
Collobert et al. (2011)*	89.59
Lin and Wu (2009)	83.78
Lin and Wu (2009)*	90.90
Huang et al. (2015)*	90.10
Passos et al. (2014)	90.05
Passos et al. (2014)*	90.90
Luo et al. (2015)* + gaz	89.9
Luo et al. (2015)* + gaz + linking	**91.2**
Chiu and Nichols (2015)	90.69
Chiu and Nichols (2015)*	90.77
LSTM-CRF (no char)	90.20
LSTM-CRF	**90.94**

Chiu and Nichols (2015), Lample et al. (2016)

Takeaways

- Explanation methods: looking at weights, LIME, gradient-based
- All kinds of NNs can be integrated into CRFs for structured inference. Can be applied to NER, other tagging, parsing, …
- This concludes the ML/DL-heavy portion of the course. Starting Tuesday: syntax, then semantics