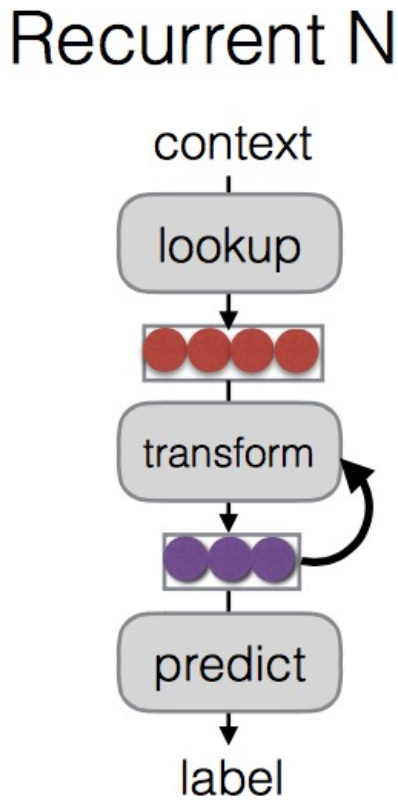
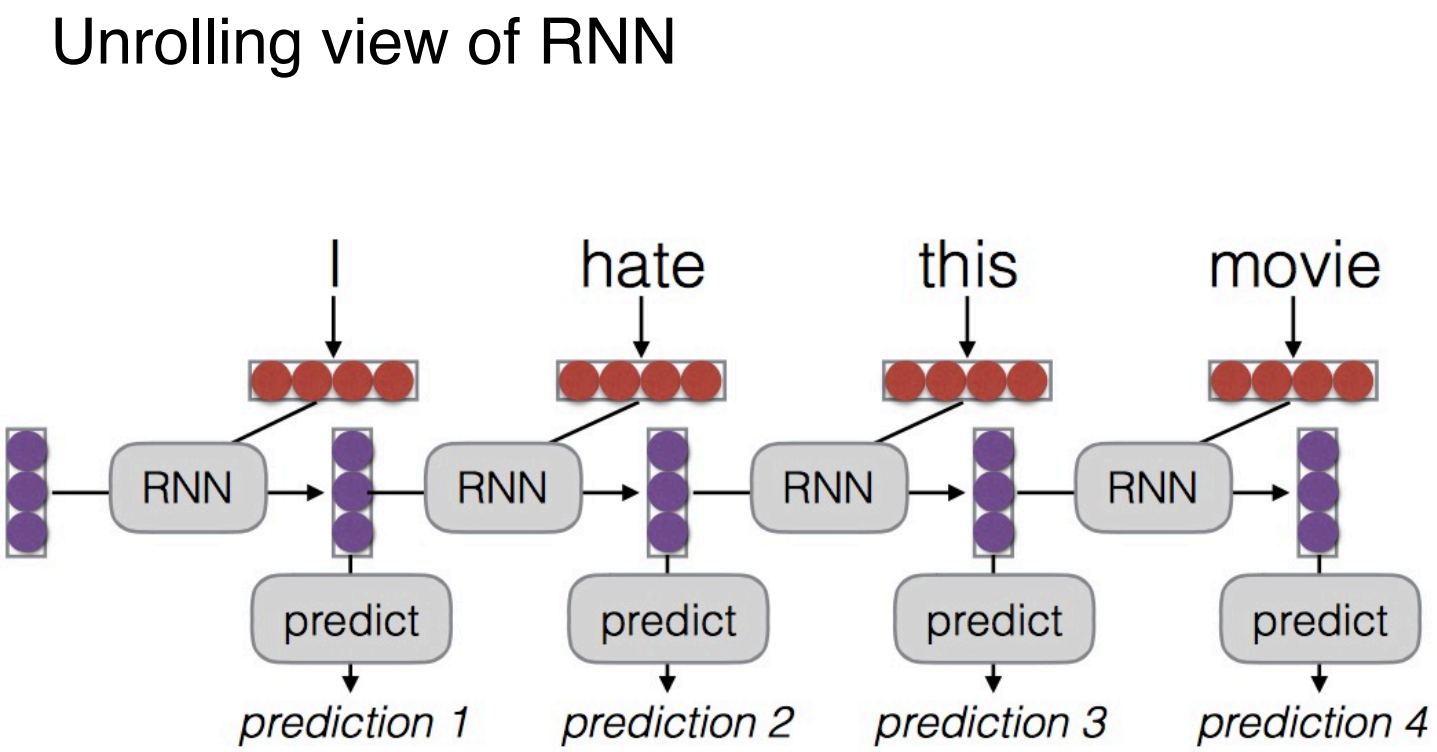
RNN with application in NLP

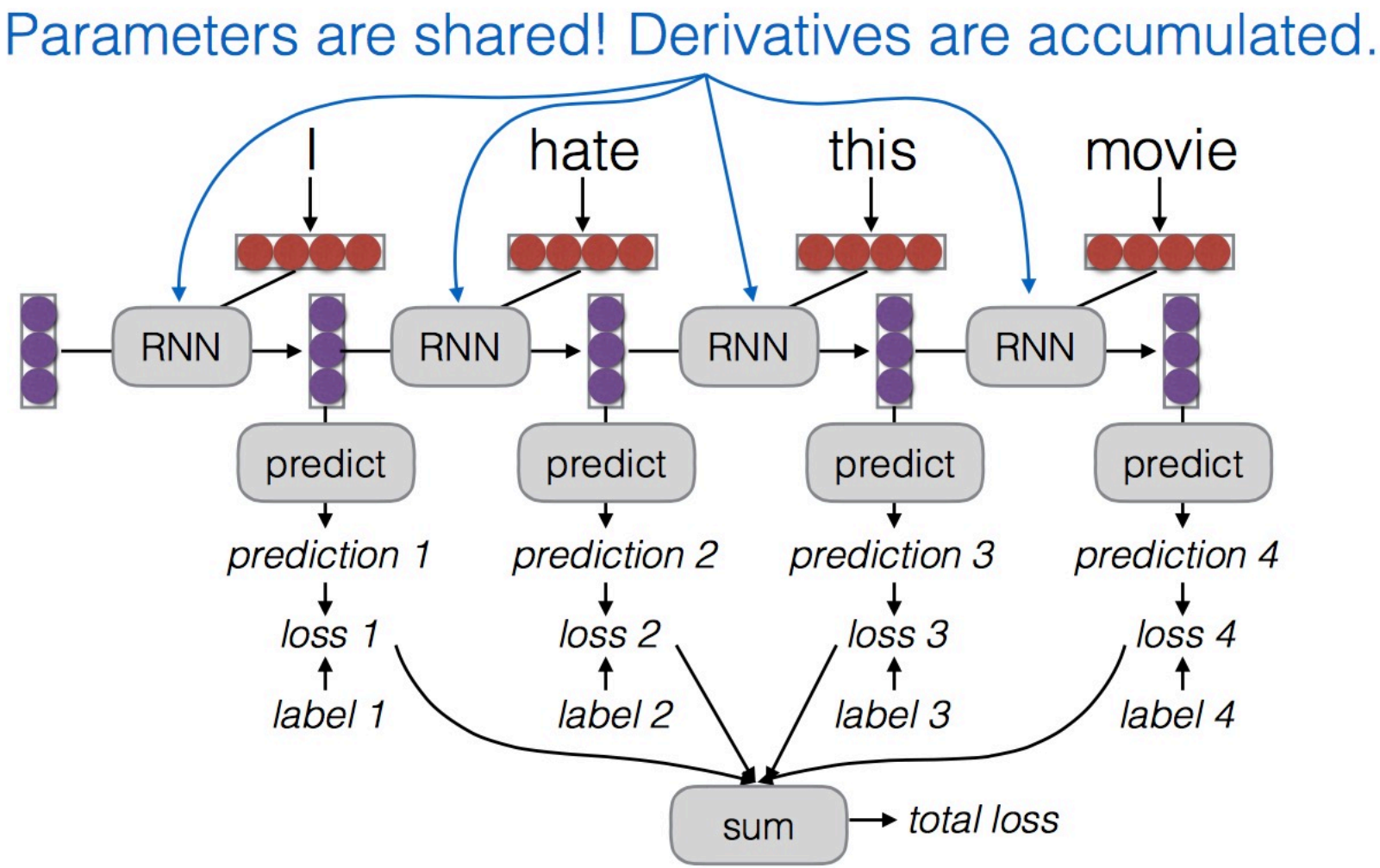
Presented by Yan Li

Recurrent Neural Networks



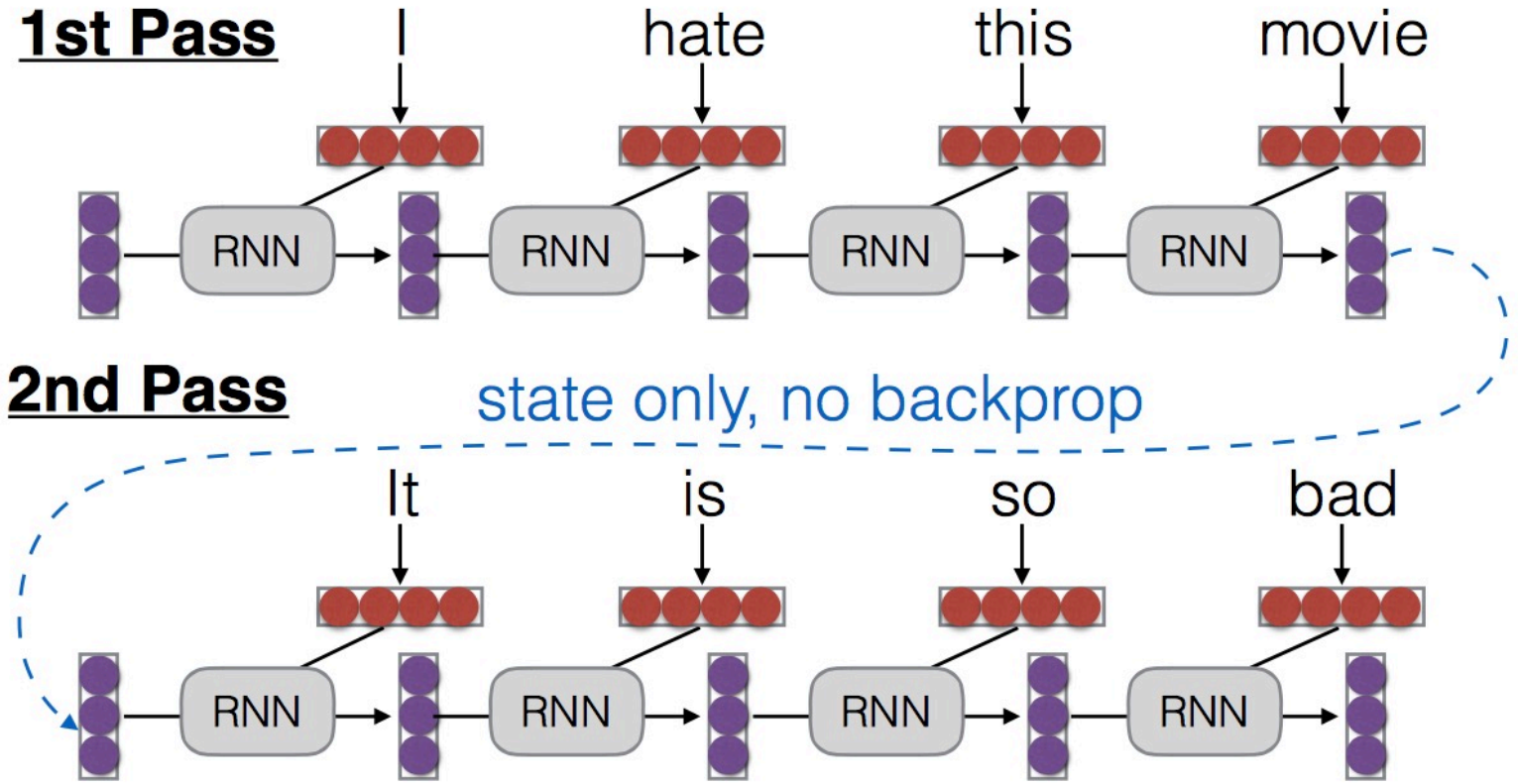


Parameter Tying



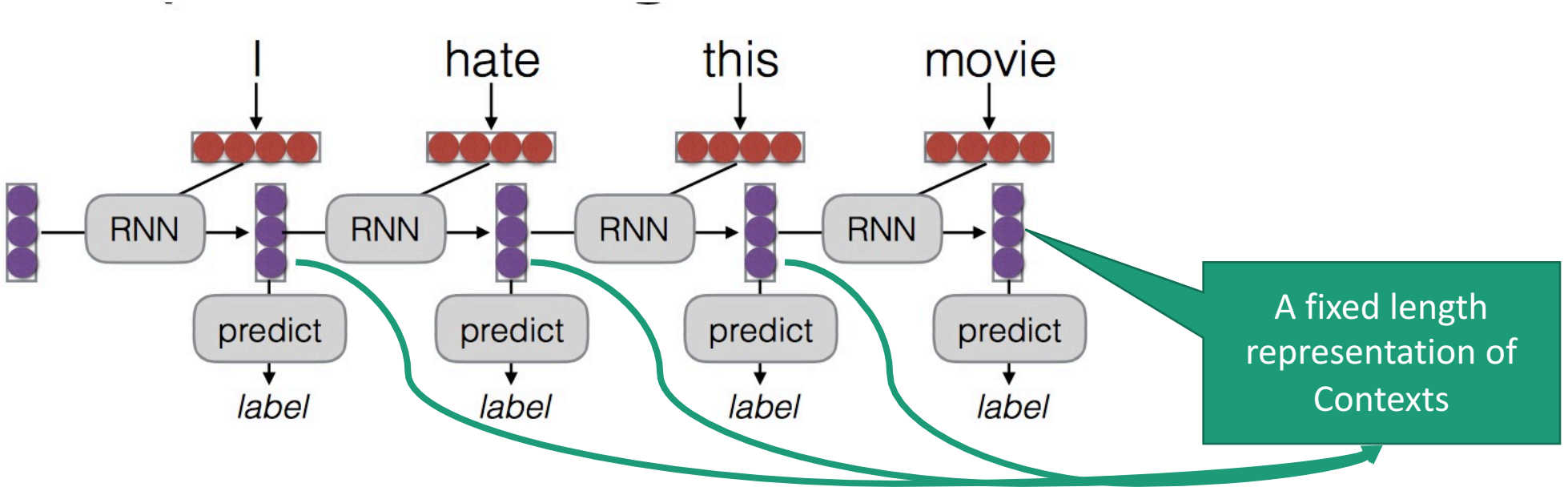
Truncated BPTT (back propagation through time)

- Backprop over shorter segments, initialize w/ the state from the previous segment



Represent a context within a sentence

- Read context up until that point, generate a representation for each token.



Tagging:

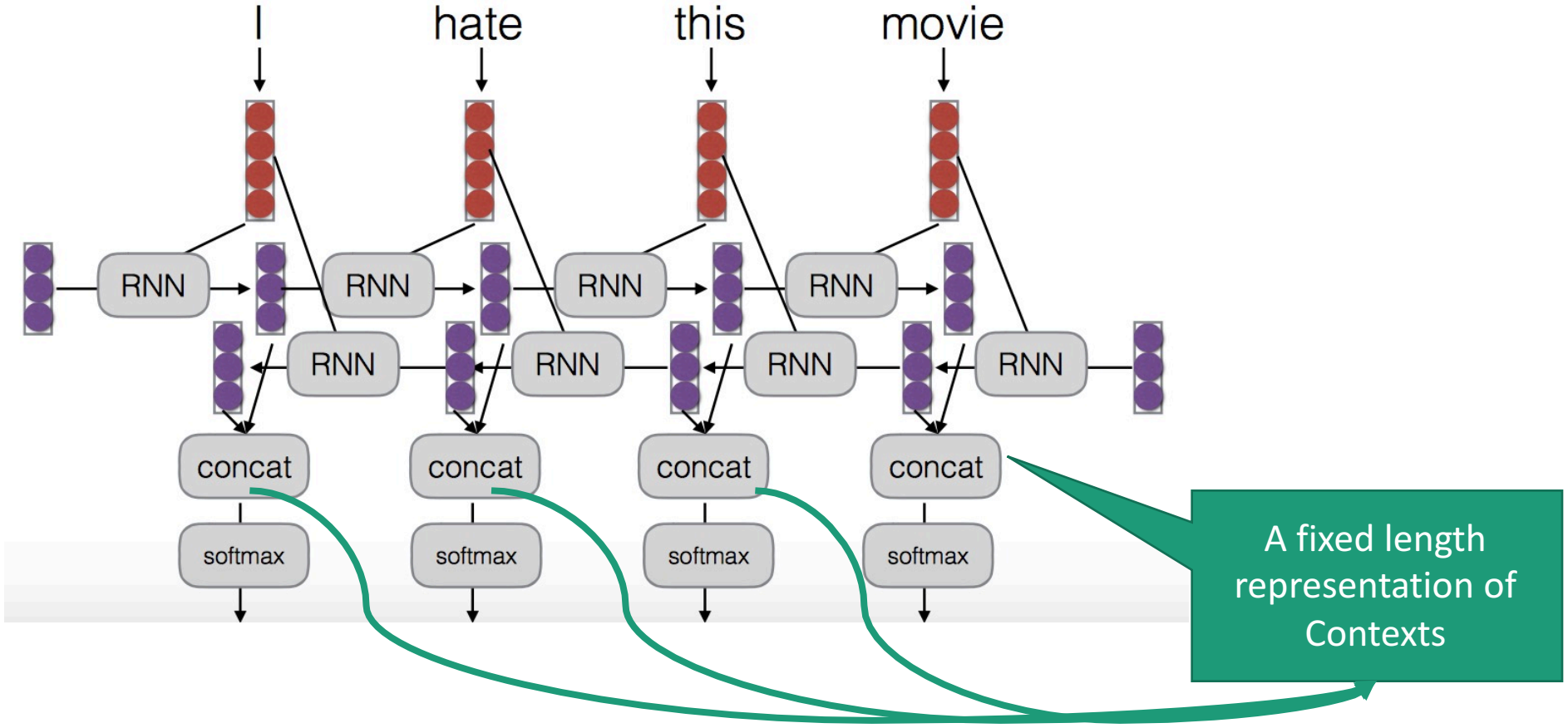
- Part-of-speech tagging (word-category: Noun, verb…)
- named entity tagging (inside, outside, beginning (IOB tagging))

Language Modeling: (The next word)

Very important! RNN based generative model

Bi-directional RNN

- A simple extension, run the RNN in both directions



Bi-directional RNN can work on tagging problems, or as encoder to generate representation for context. Usually not suitable for Language generation except gap filling.

Relationship of word representation in RNN and other methods

- Predict each word based on pervious words:

(can be used in both encoding and decoding)

$$P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1})$$

Next Word Context

N-gram: $P(x_i|x_1, \ldots, x_{i-1}) \approx P(x_i|x_{i-n}, \ldots, x_{i-1})$

RNN can handle long-term dependency.

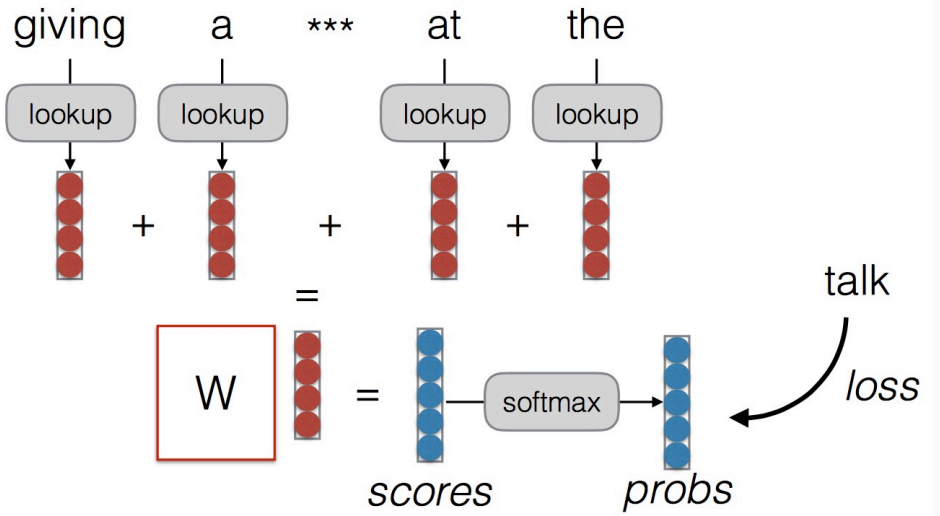
It does not need the assumption of N-gram

$$P(x_i|x_1, \ldots, x_{i-1}) = f(h_{i-1})$$

Relationship of word representation in RNN and other methods (cont.)

- Context Window Methods: Predict each word based on both before and after

Continuous bag-of-words (CBOW)
Predict word based on sum of surrounding embedding

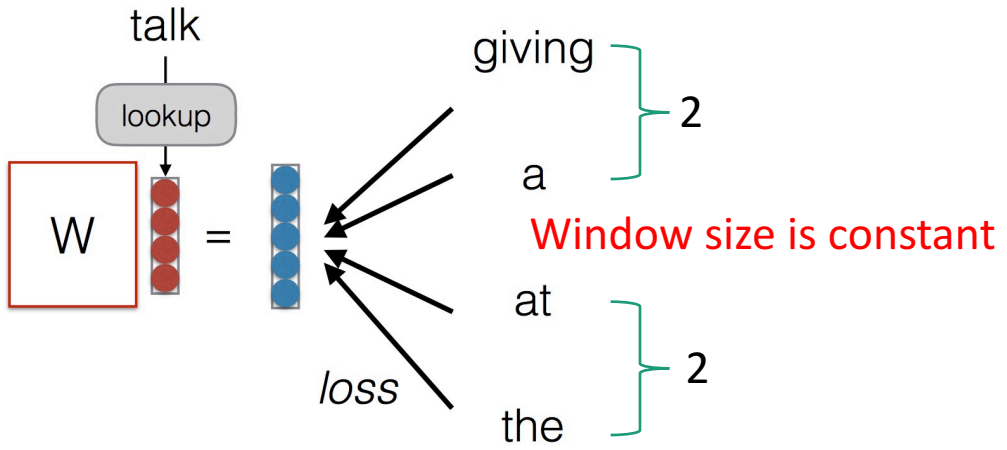


One has to predefined the window size.
The word order has not been considered.

Bi-directional RNN can be viewed as a context window method, whose window size is the whole context. And the order has been considered.

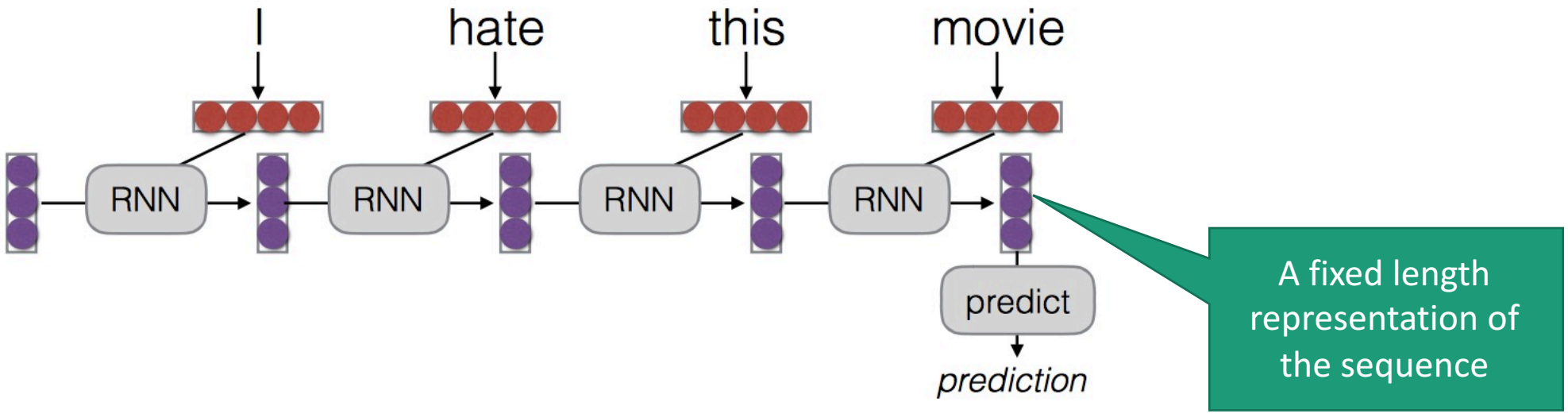
Skip-gram

Predict each word in the context given the word



Represent a sentence

- Read whole sentence, generate a representation of whole sentence and then make a prediction (information retrieval, conditional generation, sentence/document classification) based on it.



Conditioned Generation

- Sentence generation: Calculating the Probability of a Sentence

$$P(X) = \prod_{i=1}^{I} P(x_i \mid x_1, \ldots, x_{i-1})$$

Next Word — Context

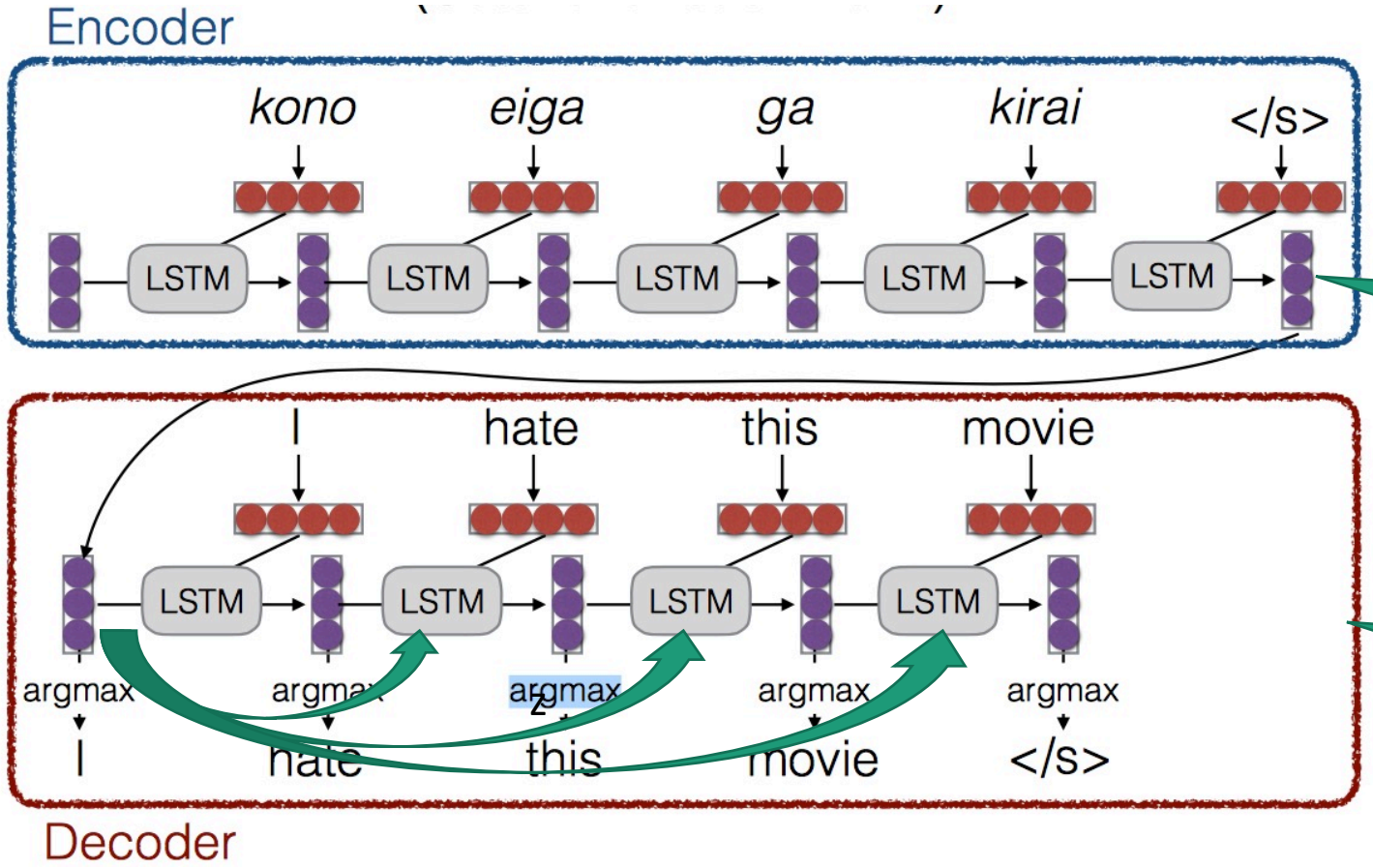
- Conditional Language Models

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

Added Context!

The auxiliary information "X" can be generated from different context in different problems:
(image: image captioning,
sentence in source language: Machine Translation,
Question: Q A)
….

Basic Seq2seq



A fixed length representation of the sequence as auxiliary information "X"

Decoder is a RNN based conditional language model

The last Hidden State of encoder (representation of sequence) can be passed to decoder via:

- Direct pass to initialize state (above picture)
- Transform then initialize
- Input at every time step of decoder

Generate sequence: We have a model of P(Y|X), how to generate a sentence.

- **Sampling:** Try to generate a *random* sentence according to the probability distribution. (Can get global optimum but very inefficient)
- **Argmax:** Try to generate the sentence with the *highest* probability.

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

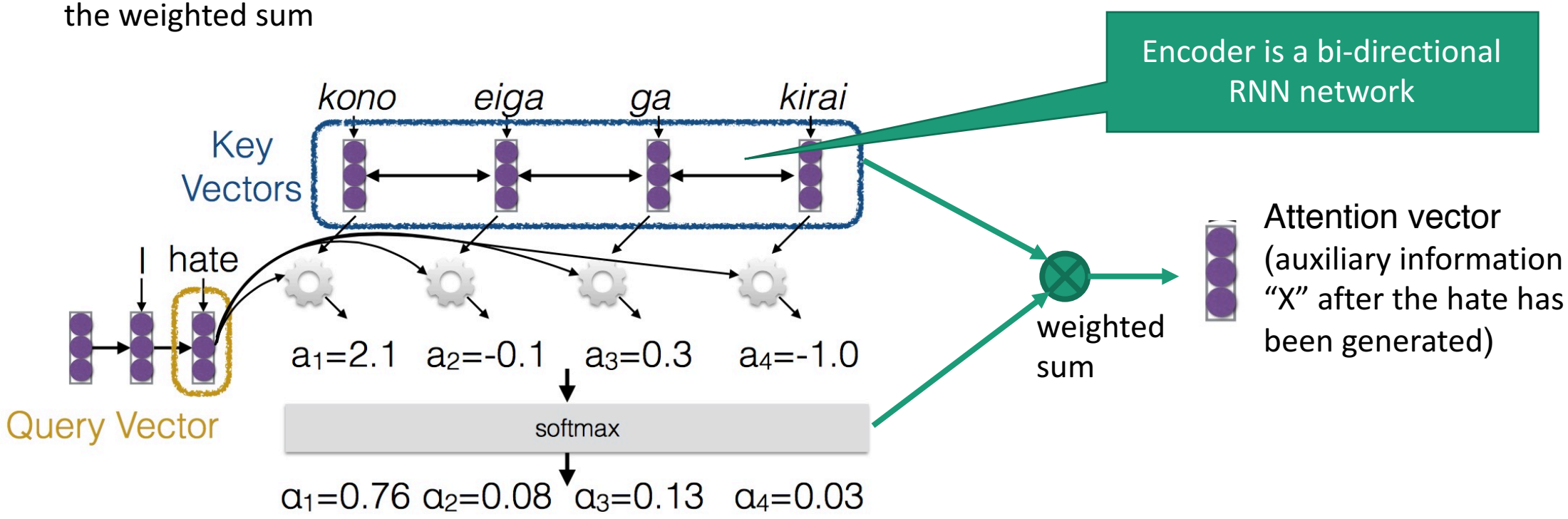
Not exact, real problems:
Will often generate the “easy” words first
Will prefer multiple common words to one rare word

- Beam Search: Instead of picking one high-probability word, maintain several paths (also a greed search similar like Argmax)

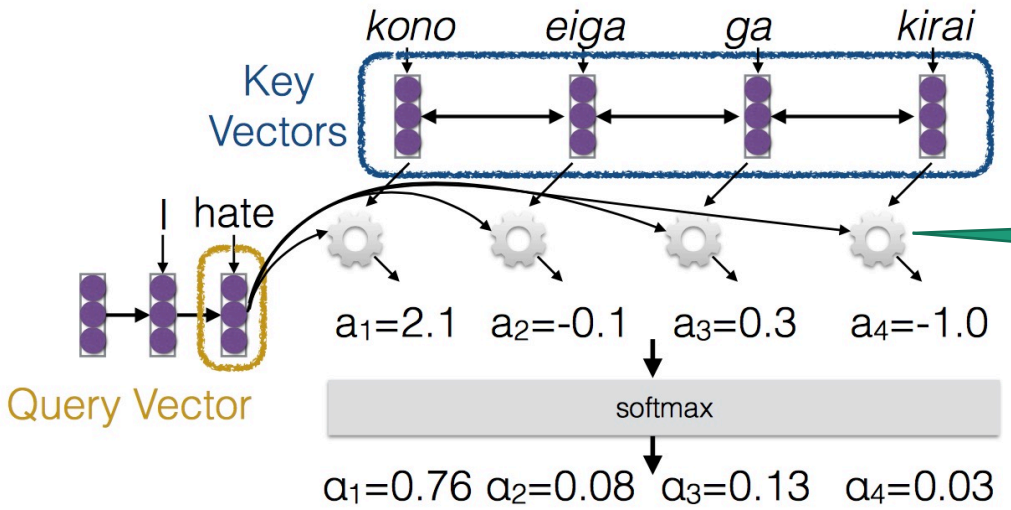
Attention

- Idea: it's hard to encode a sentence into one single vector, and when we do NMT the model only need to focus on local rather than golbal.

1. Use "query" vector (decoder state) and "key" vectors (all encoder states)
2. For each query-key pair, calculate weight
3. Normalize to add to one using softmax
4. Combine together value vectors (usually encoder states, like key vectors) by taking the weighted sum



Attention Score Functions

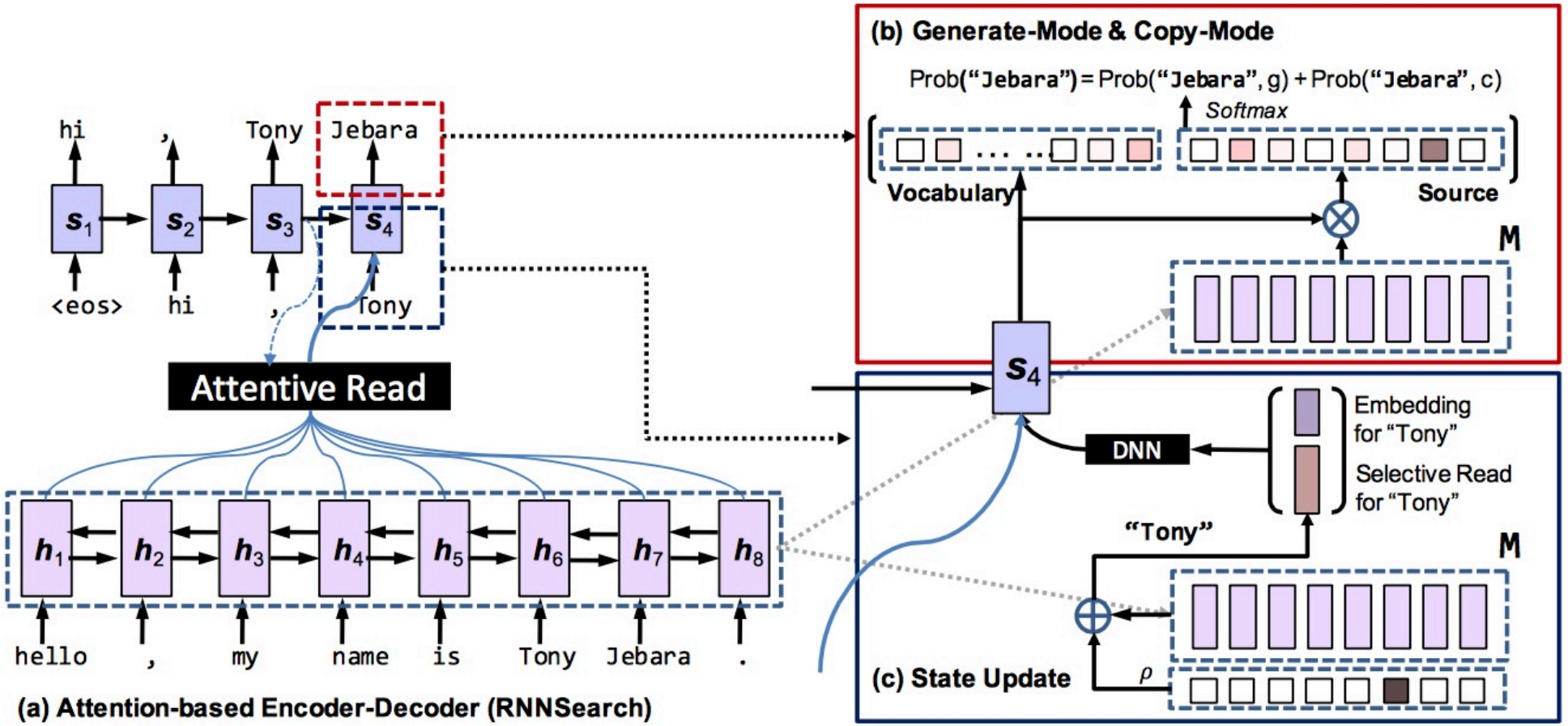


How to design the computational mechanism to compute attention score?

q is the query and ***k*** is the key

- **Multi-layer Perceptron:** $a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{w}_2^\intercal \tanh(W_1[\boldsymbol{q}; \boldsymbol{k}])$ (Flexible, often very good with large data)
- **Bilinear:** $a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal W \boldsymbol{k}$
- **Dot Product:** $a(\boldsymbol{q}, \boldsymbol{k}) = \boldsymbol{q}^\intercal \boldsymbol{k}$ (No parameters! But requires sizes to be the same.)
- **Scaled Dot Product:** $a(\boldsymbol{q}, \boldsymbol{k}) = \frac{\boldsymbol{q}^\intercal \boldsymbol{k}}{\sqrt{|\boldsymbol{k}|}}$ (scale by size of the vector as dot product increases as dimensions get larger)

Copying mechanism (Gu et al. 2016)



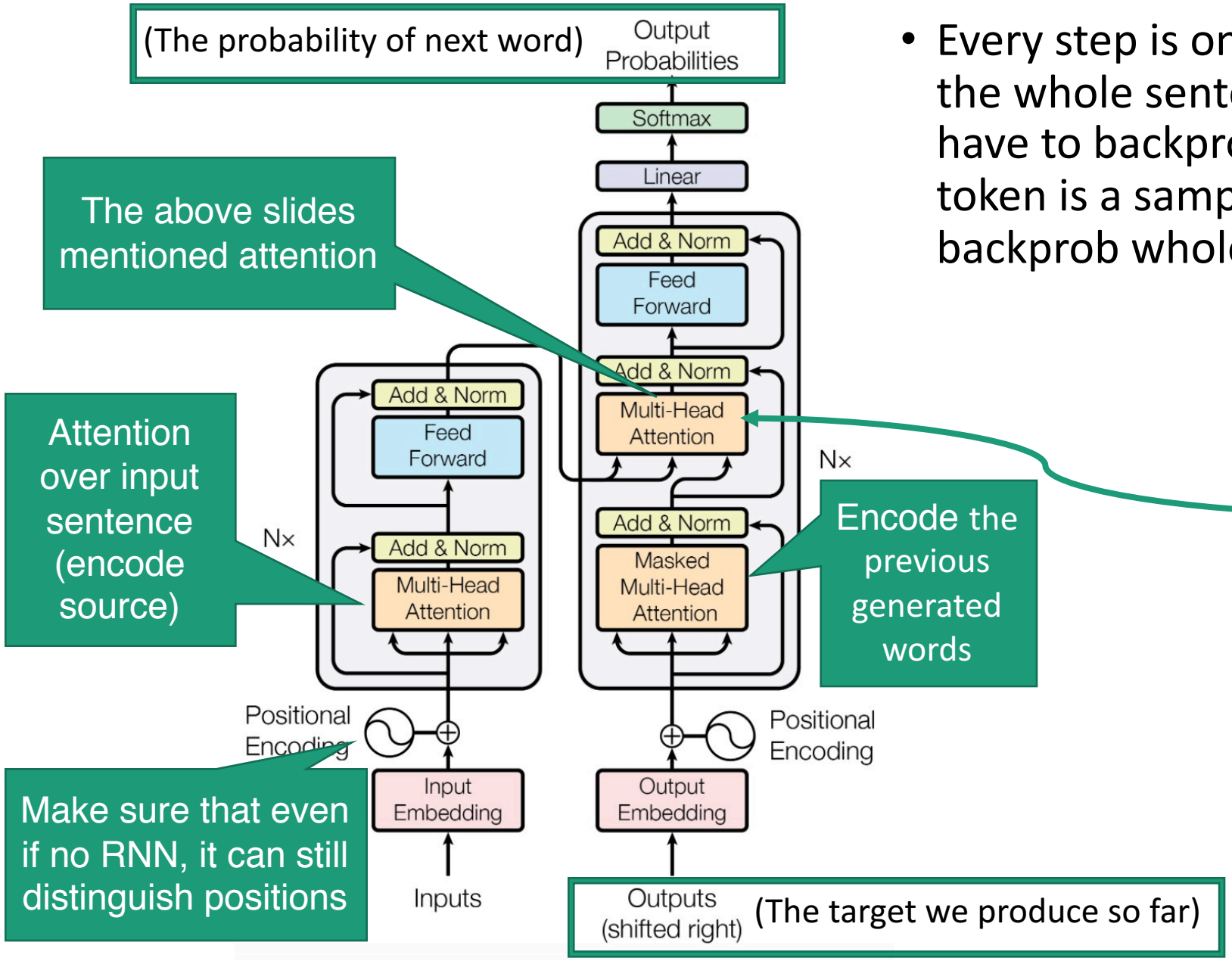
Prediction

Attention

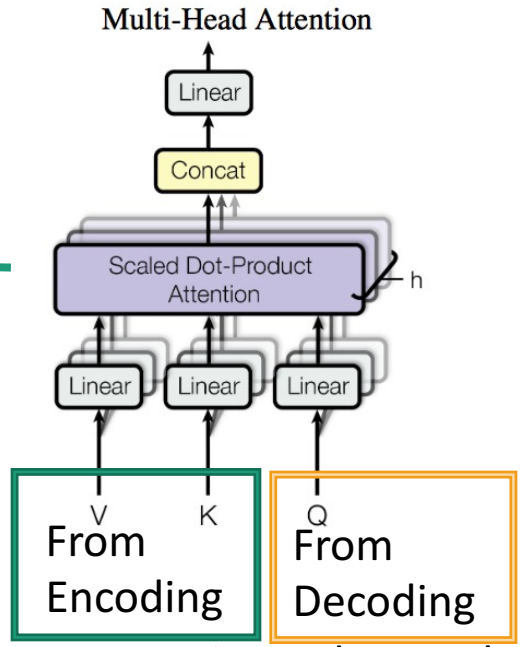
What do we Attend To?

- In language modeling, attend to the previous words (Merity et al. 2016)
- In translation, attend to either input or previous output (Vaswani et al. 2017)
- Images caption attend part of image (Xu et al. 2015)
- Encode with attention over each sentence, then attention over each sentence in the document (Yang et al. 2016)
- ...

Attention Is All You Need (Vaswani et al. 2017)



- Every step is one training sample, in RNN the whole sentence is one sample, you have to backprob all sentence. Here each token is a sample, one don't need to backprob whole sentence.



This structure is used several times. In the arrowhead one the input from different part

Up to Now, all the aforementioned models are non-structure learning. In these models the output are independent.

In NLP or other tasks (i.e., image segmentation) the output is not just a value but have certain structure, or encoding the relationship among outputs can improve performance a lot.

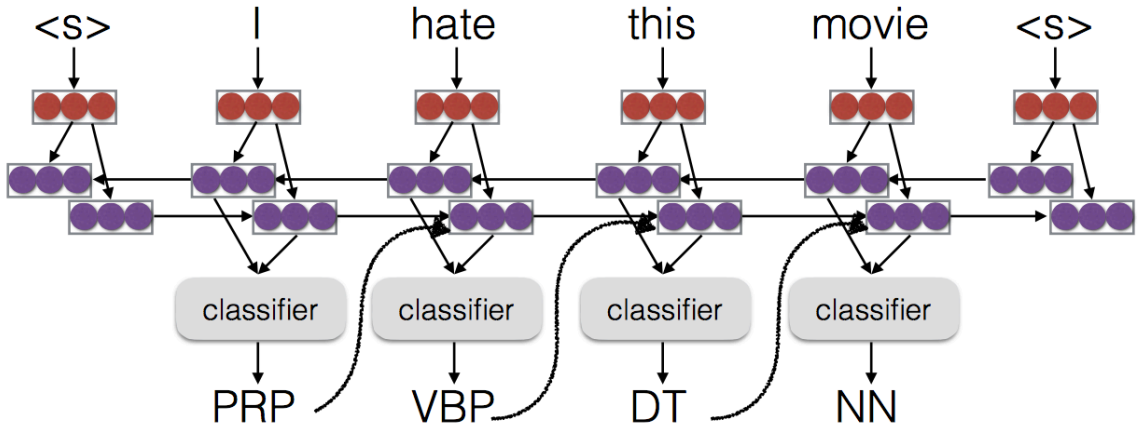
- For example, in POS tagging usually a verb cannot followed by another verb. If this kind of information can be encoded, then the performance will be improved.

Structure learning

- Input and output are both objects with structures, i.e., sequence, list, tree, …
- Example in NLP:
  - Speech recognition
    - X: speech signal (sequence) → Y: text (sequence)
  - Translation
    - X: Chinese (sequence) → Y: English (sequence)
  - Syntactic parsing
    - Sentence → Y: parsing tree (tree structure)
  - Summarization
    - Long document → Y: summary (short paragraph)

Case study (POS tagging)

- A Tagger Considering Output Structure



Teacher forcing assumes feeding correct previous input, but at test time we may make mistakes that propagate. (Exposure bias)

- **Locally normalized models:** each decision made by the model has a probability that adds to one

$$P(Y \mid X) = \prod_{j=1}^{|Y|} \frac{e^{S(y_j|X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{y}_j \in V} e^{S(\tilde{y}_j|X,y_1,\ldots,y_{j-1})}}$$

- **Globally normalized models (a.k.a. energy-based models):** each sentence has a score, which is not normalized over a particular decision

$$P(Y \mid X) = \frac{e^{\sum_{j=1}^{|Y|} S(y_j|X,y_1,\ldots,y_{j-1})}}{\sum_{\tilde{Y} \in V*} e^{\sum_{j=1}^{|\tilde{Y}|} S(\tilde{y}_j|X,\tilde{y}_1,\ldots,\tilde{y}_{j-1})}}$$

Unified Framework

Training

- Find a function F

$$\mathrm{F}: X \times Y \rightarrow \mathrm{R}$$

- F(x,y): evaluate how compatible the objects x and y is

Inference (Testing)

- Given an object x

$$\tilde{y} = \arg\max_{y \in Y} F(x, y)$$

$$f: X \rightarrow Y \Rightarrow f(x) = \tilde{y} = \arg\max_{y \in Y} F(x, y)$$

In some problem F(x, y) can be viewed as joint probability

Three Problems

Problem 1: Evaluation

- What does F(x,y) look like?

Problem 2: Inference

- How to solve the "arg max" problem

$$y = \arg\max_{y \in Y} F(x, y)$$

Problem 3: Training

- Given training data, how to find F(x,y)

- Structured Perceptron

  Aim: the score of true label-data combination is grater than any other combination:

$$\forall n, \forall Y \in \mathbb{Y}, Y \neq \hat{Y}^n: \quad F_\theta\left(\mathrm{X}^\mathrm{n}, \hat{Y}^n\right) \geq F_\theta\left(\mathrm{X}^\mathrm{n}, Y\right)$$

  How to achieve: Find the best prediction $\tilde{Y}^n = \underset{Y \in \mathbb{Y}}{\operatorname{argmax}} F_\theta\left(\mathrm{X}^\mathrm{n}, Y\right)$ under current θ, then update θ accordingly

$$\theta \leftarrow \theta + \alpha\left(\frac{\partial F_\theta\left(\mathrm{X}^\mathrm{n}, \hat{Y}^n\right)}{\partial \theta} - \frac{\partial F_\theta\left(\mathrm{X}^\mathrm{n}, \tilde{Y}^n\right)}{\partial \theta}\right)$$

- Structured SVM: add margin and error in Structure perceptron

  Error function: $\Delta\left(\hat{Y}^n, \mathrm{Y}^\mathrm{n}\right)$ difference between $\hat{Y}^n$ and Y^n

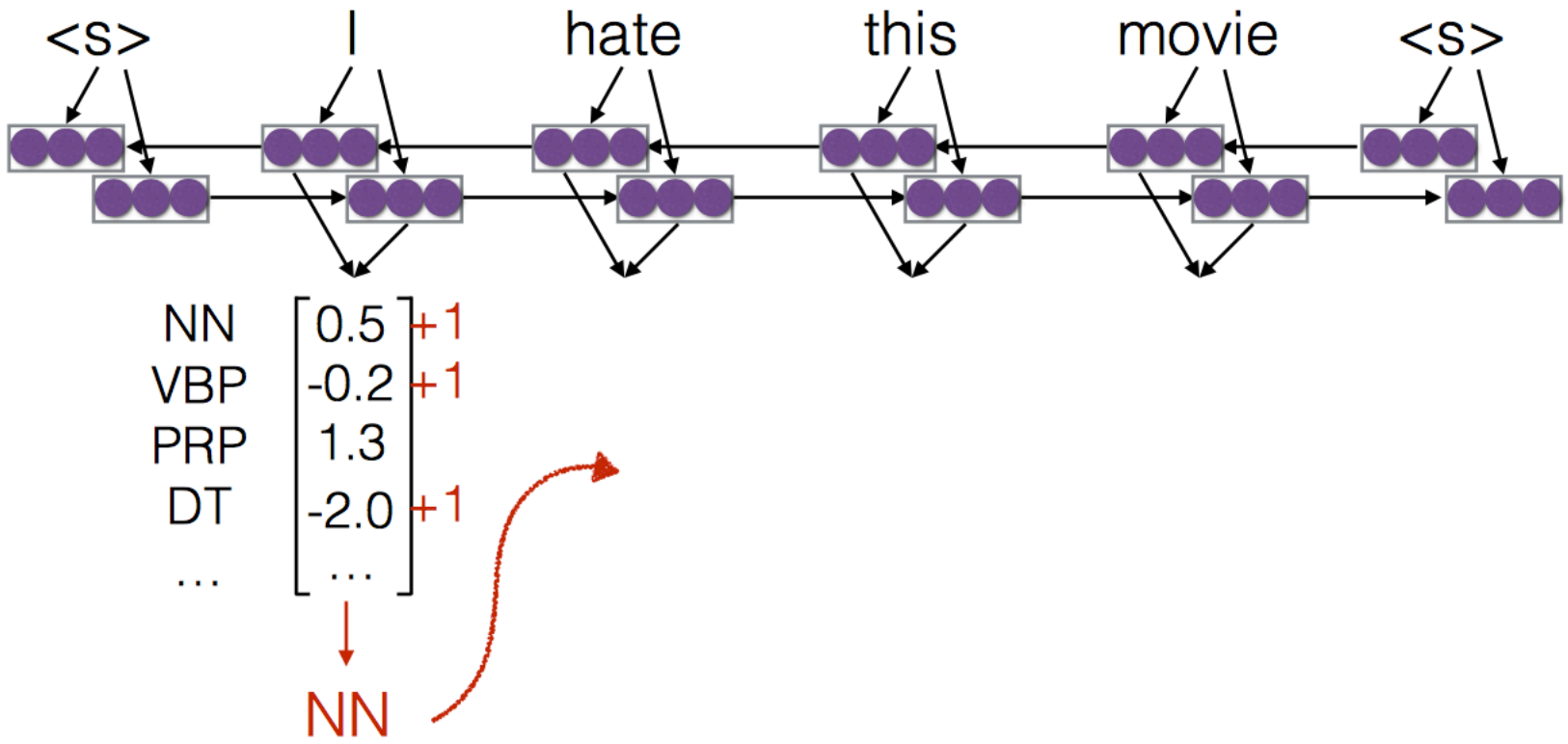
  Find the candidate which is different and have high score:

$$\bar{Y}^n = \underset{Y \in \mathbb{Y}}{\operatorname{argmax}}\left[\Delta\left(\hat{Y}^n, Y\right) + F_\theta\left(\mathrm{X}^\mathrm{n}, Y\right)\right]$$

  then update θ accordingly

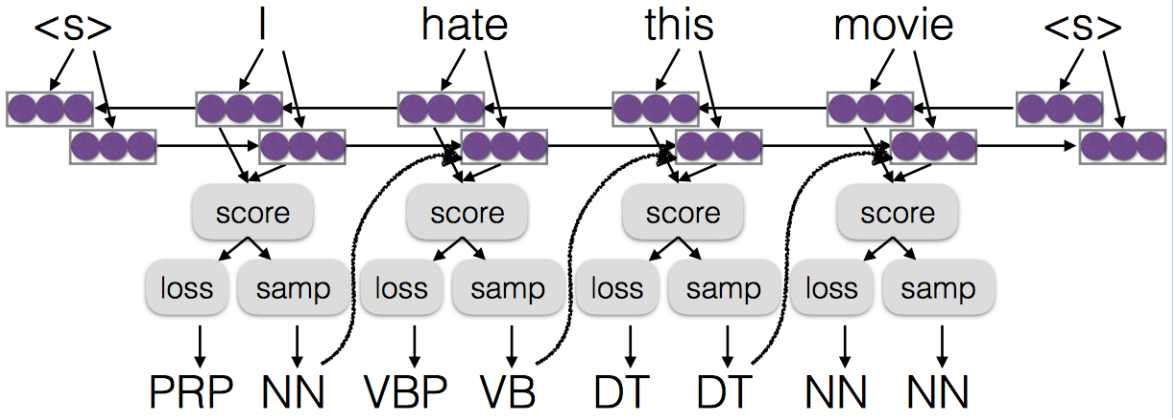
$$\theta \leftarrow \theta + \alpha\left(\frac{\partial F_\theta\left(\mathrm{X}^\mathrm{n}, \hat{Y}^n\right)}{\partial \theta} - \frac{\partial F_\theta\left(\mathrm{X}^\mathrm{n}, \bar{Y}^n\right)}{\partial \theta}\right)$$

Note only a few

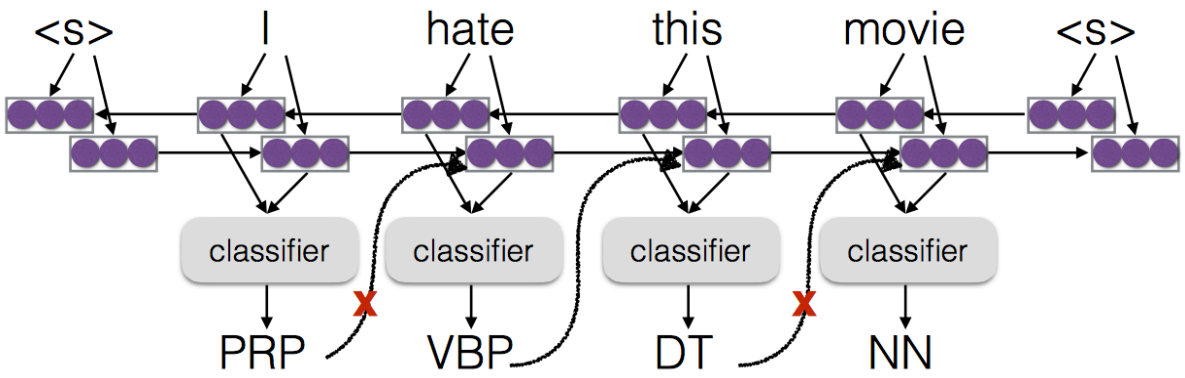


Drawback of above two methods

- Considers fewer samples, so **unstable**
- Requires inference (decoding) while training, so the algorithm takes long time to train.
- Generally must resort to pre-training (and even then, it's not as stable as teacher forcing w/ MLE).



DAgger (Ross et al. 2010), also known a "scheduled sampling", start with no mistakes, and then gradually introduce them using annealing



Basic idea: Simply don't input the previous decision sometimes during training (Gal and Ghahramani 2015). Helps ensure that the model doesn't rely too heavily on predictions, while still using them

- Conditional Random Fields (CRF)

$$F_\theta(\mathrm{X}, Y) = P(Y|X) = \frac{\exp(S_\theta(X,Y))}{\sum_{Y' \in \mathbb{Y}} S_\theta(X,Y')} = \frac{\psi(X,Y)}{\sum_{Y' \in \mathbb{Y}} \psi(X,Y')}$$

An extension of Logistic regression

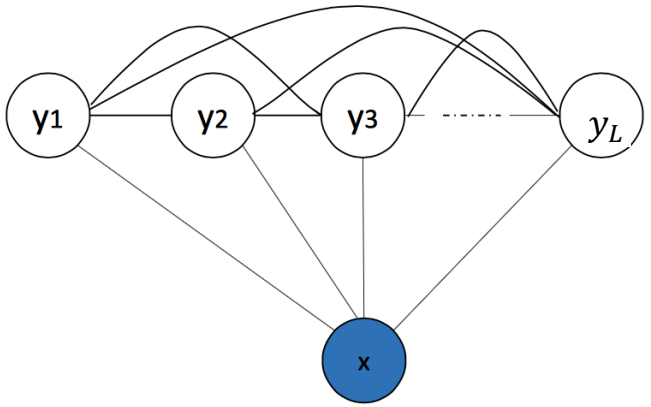
Training:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} -\log[P(Y|X)]$$

An extension of cross entropy

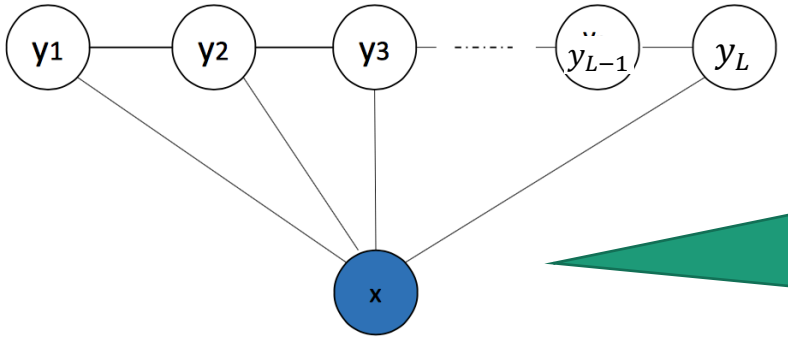
CRF is an extension of logistic regression, in logistic regression each output is a vector, while in CRF each output is a structure with multiple components.

General form of globally normalized model



$$P(Y|X) = \frac{\psi(Y,X)}{\sum_{Y'} \psi(Y',X)}$$

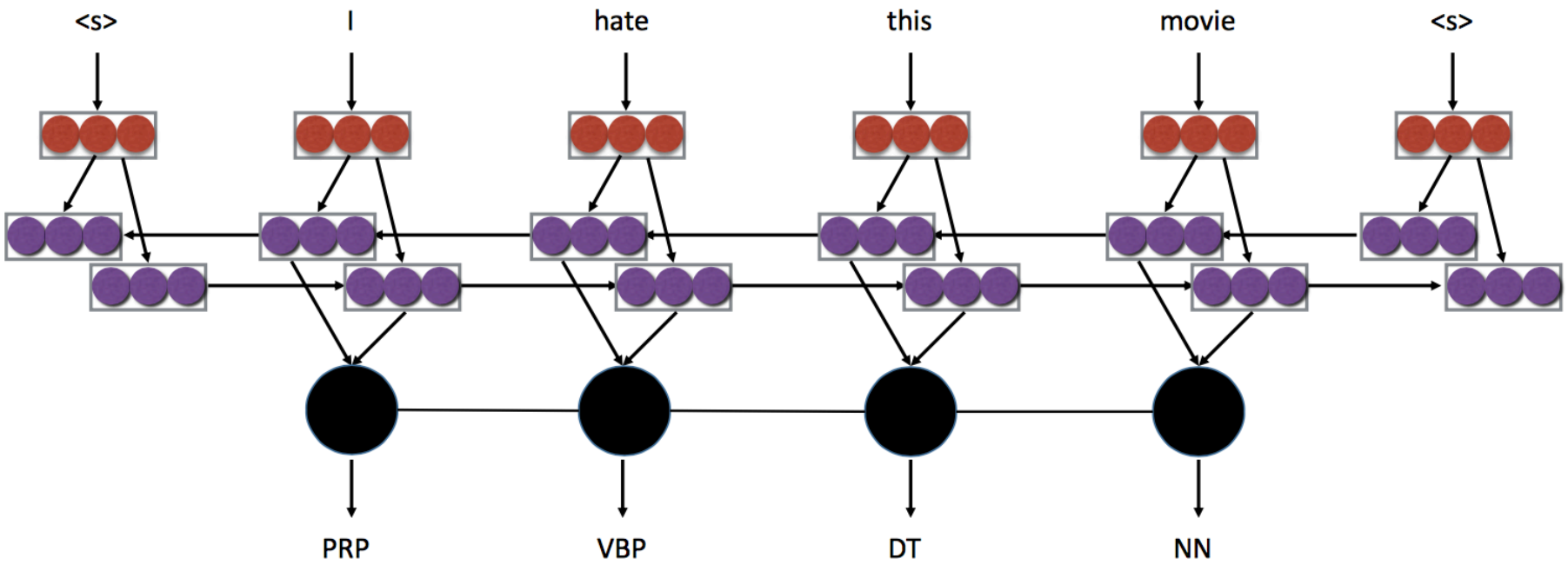
First-order linear CRF



$$P(Y|X) = \frac{\prod_{i=1}^{L} \psi_i(y_{i-1}, y_i, X)}{\sum_{Y'} \prod_{i=1}^{L} \psi_i(y'_{i-1}, y'_i, X)}$$

Add independent assumption
In graphical model to simplify the problem. In the first order linear CRF the Viterbi Algorithm can be used in calculation

BiLSTM-CRF for Sequence Labeling



Compare CRF and Structured Perceptron

- ***Structured Perceptron***

$$\tilde{y}^n = \arg\max_y w \cdot \phi(x^n, y)$$

$$w \rightarrow w + \phi(x^n, \hat{y}^n) - \phi(x^n, \tilde{y}^n)$$

Hard

- ***CRF***

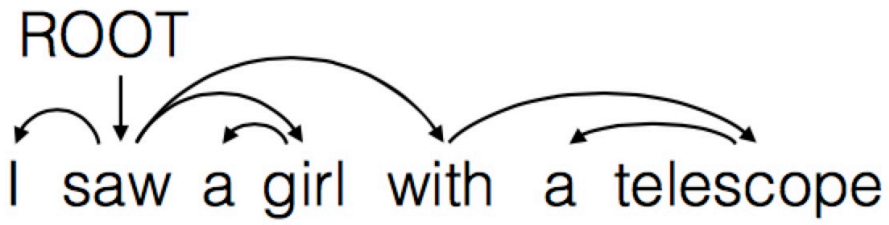
$$\mathrm{w} \rightarrow w + \eta \left(\phi(x^n, \hat{y}^n) - \sum_{y'} P(y'|x^n)\, \phi(x^n, y') \right)$$

Soft

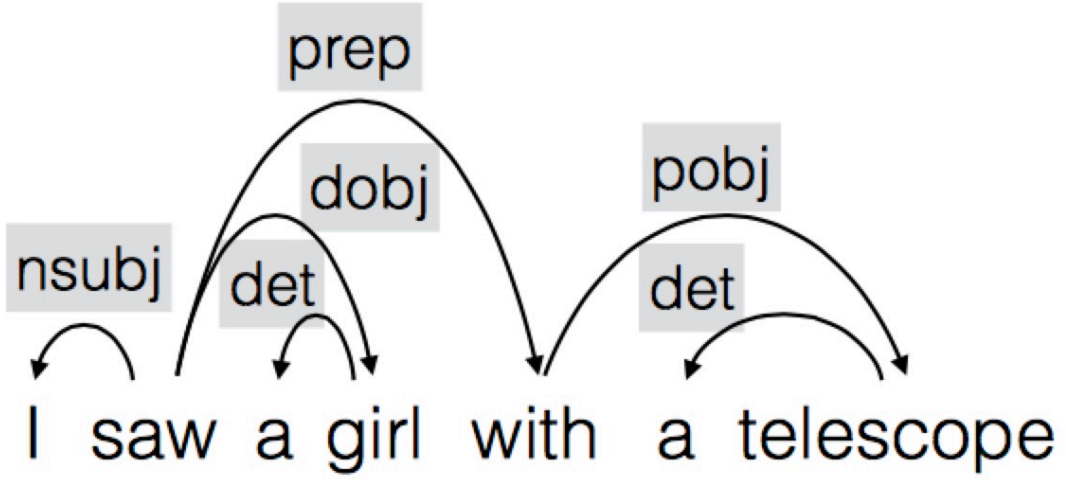
Parsing

Dependency Parsing

- focus on relations between words; Dependencies are often good for semantic tasks, as related words are close in the tree.



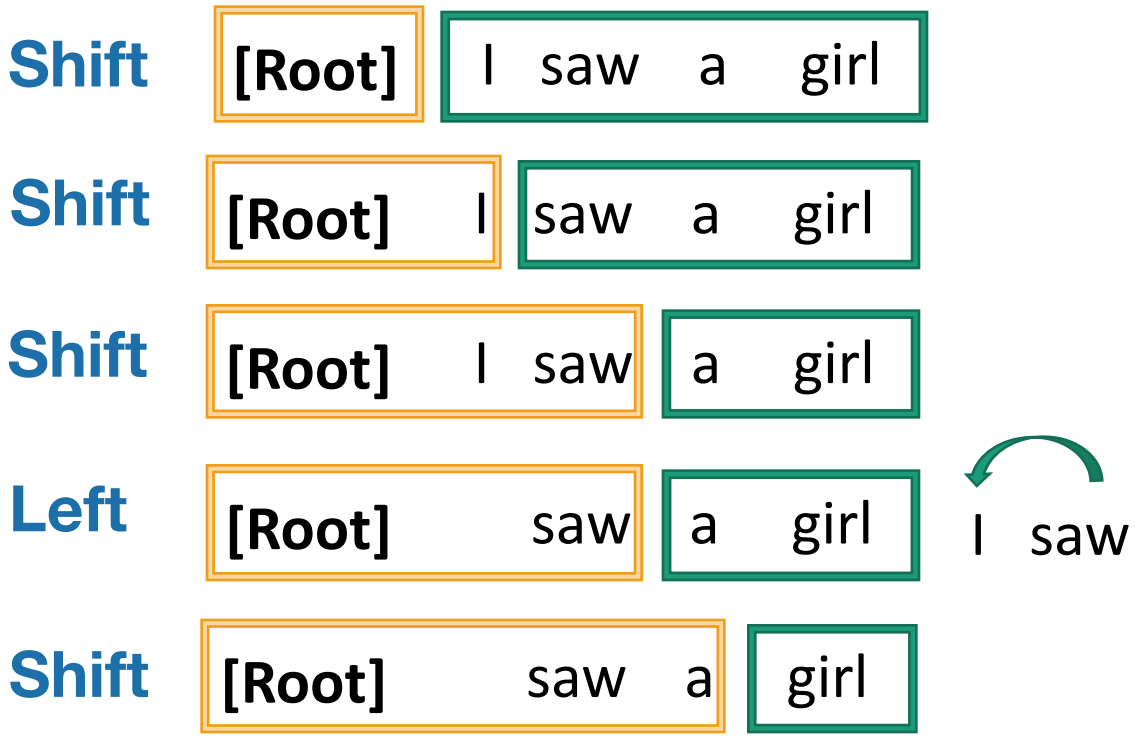
Only one word is dependent of root.
Don’t want cycles

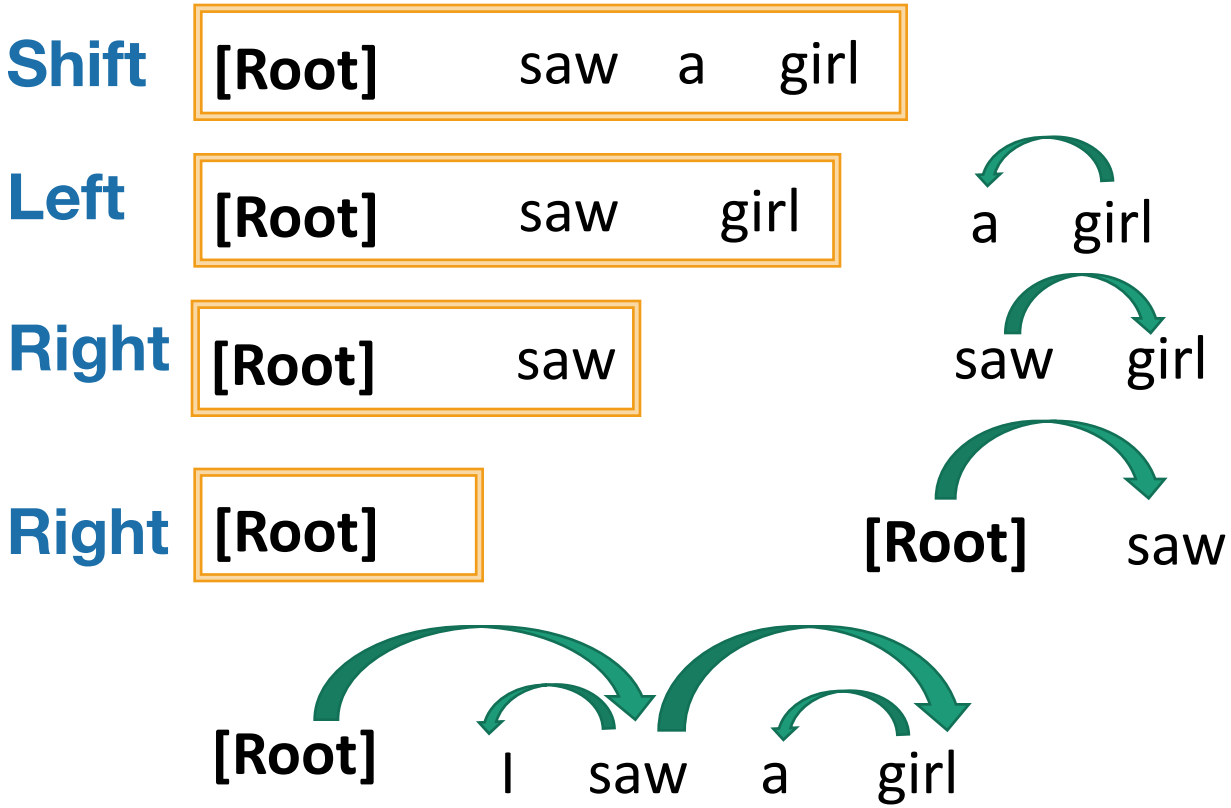


labeled dependencies, that explicitly show the relationship between words

Shift-Reduce Parsing: Process words one-by-one left-to-right

- Two data structures
  Queue: of unprocessed words **Stack:** of partially processed words
- Three actions at each step:
  Shift: move one word from queue to stack
  Reduce left/ Left Arc: top word on stack is head of second, build left arc and remove second word
  Reduce right/ Right Arc: second word on stack is head of top, build right arc and remove first word





Dependency Parsing

- Given a **configuration,** Which **action** do we choose?

  Shift **Reduce left/ Left Arc** **Reduce right/ Right Arc**

- The Dependency Parsing then has been transformed into a serious of 3-class classification problem.

A Feed-forward Neural Model for Shift-reduce Parsing (Chen and Manning 2014)

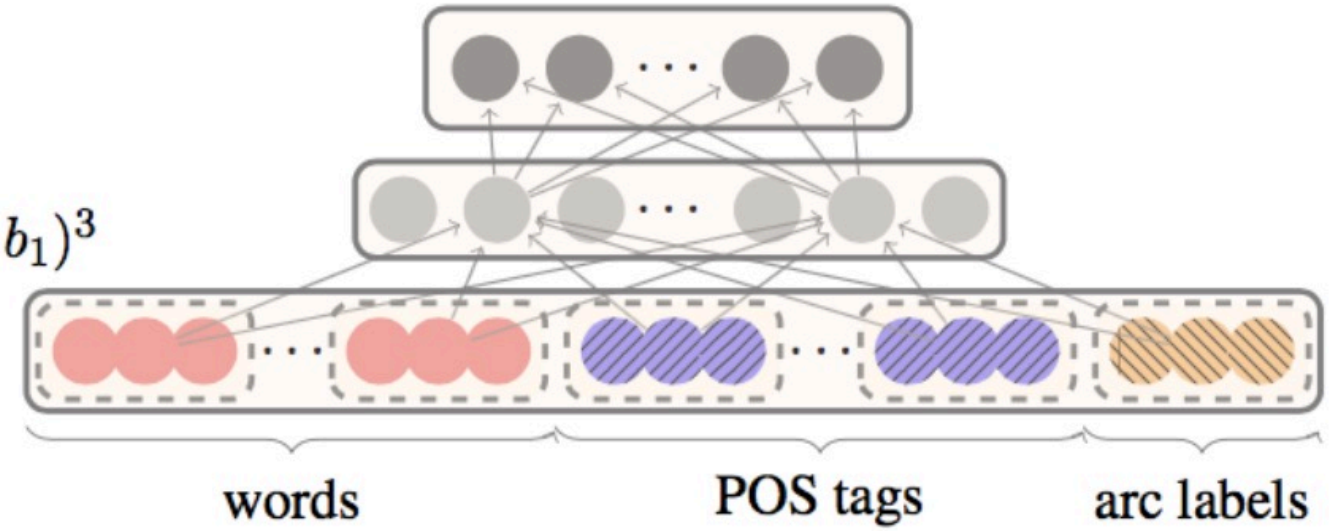
Softmax layer:

$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

Input layer: $[x^w, x^t, x^l]$



Phrase Structure rules for English

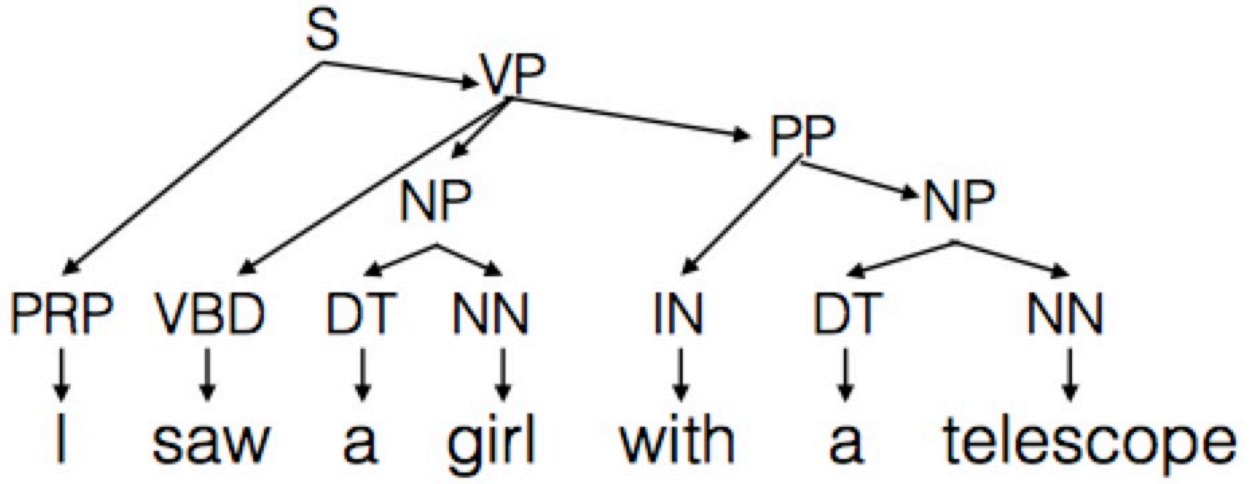
S → NP (Aux) VP
NP→(Det) (Adj) **N** (PP)
PP → P (NP)
VP → V (NP) (PP) (Adj)

S→ sentence
NP → Noun phrase （名词短语）
VP → verb phrase （动词短语）
Aux → auxiliary (助动词)
PP → prepositional Phrase (介词短语)
Det → determiner (限定词)
Adj → adjective (形容词)
Adv -- adverb (副词)

Conjunction： Words and phrases of the same category can be combined using conjunctions (*and, but, or*)
NP→NP conj NP
VP→VP conj VP
S→S conj S

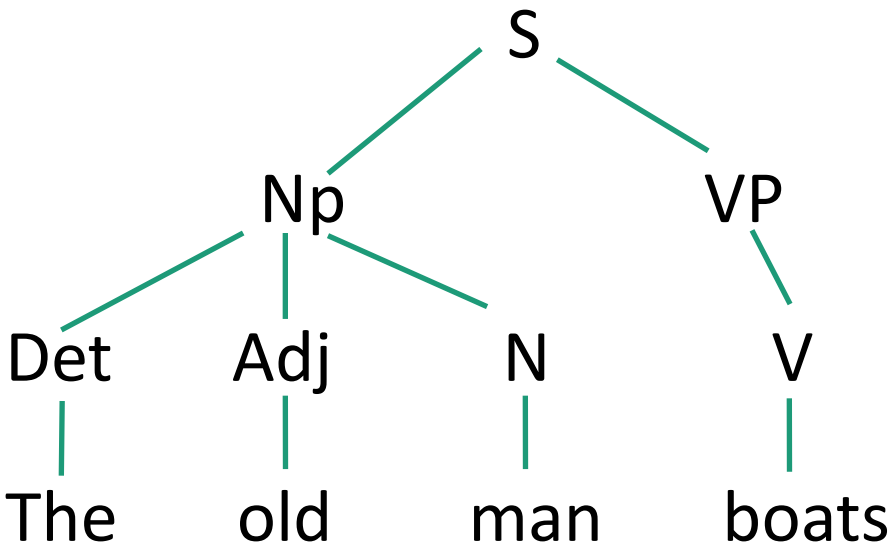
Recursive (Tree structured) Neural Networks

- **Phrase structure:** focus on the structure of the sentence, recursion is natural for describing language

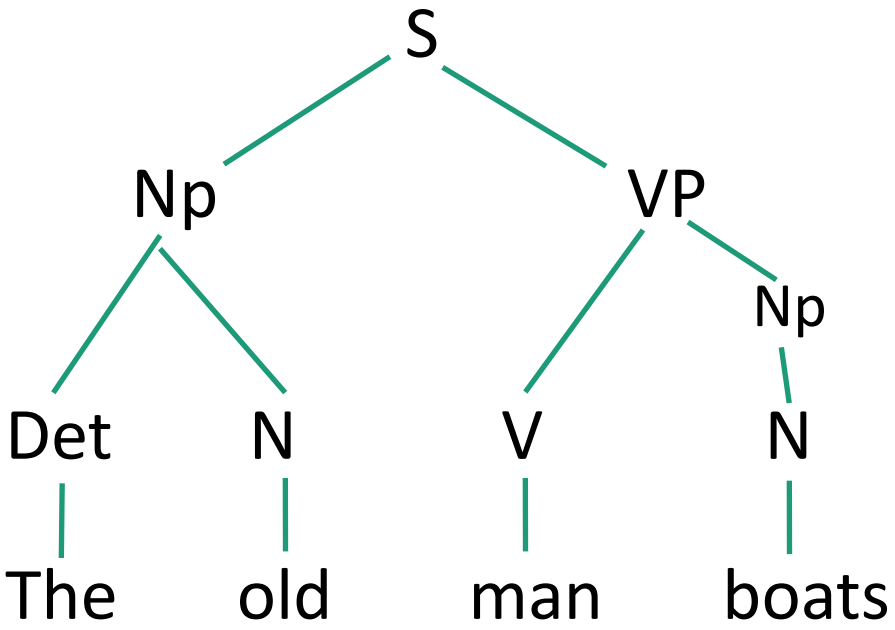


Works better for some tasks to use grammatical tree structure

We can represent the meaning of longer phrases by mapping them into the same vector space!



一个老头在划船



Disambiguate via drawing phrase tree

老年人在操纵船

Probability Context-Free Grammars (PCFGs)

- Add probability to each distinct Phrase Structure rules, the probability of whole sentence is the multiplication of the probability of all selected rules.
- The TreeBanks, e.g., Penn Treeback has the sentence and it’s Phrase Structure

Parsing with a PCFG

- Given a PCFG and a sentence *s*, define T(*s*) to be the set of trees with *s* as the yield
- Assume the PCGG is in Chomsky Normal Form: G=(N, T, R, S)
  - N: a set of non-terminal symbols
  - T: a set of terminal symbols
  - S: start symbol, S$\in N$
  - R: a set of rules have two forms:
    X→ AB, X , $A, and\ B \in N$ X→Y, X$\in N$, Y$\in T$
- CFG can be concert to CNF form easily.

The CKY Parsing algorithm

- Define a dynamic programming table
  - π[I,j,X] = maximum probability of a constituent with non-terminal X spanning words i,…, j inclusive
  - The goal is to calculate $\min_{t \in T(s)} p(t) = \pi$[1, n, S], n id the sentence length
- Base case definition: for all $i = 1 \ldots n$, for X$\in N$

  π[i, i, X]=q(X→ w_i) where w_i $is\ the\ i-th\ token$
- Recursive definition: for all $i = 1 \ldots n-1$, j $= (i+1) \ldots n$, X$\in N$

  π[i, j, X]= $\max_{\substack{(X \to AB) \in R \\ s \in \{i \ldots (j-1)\}}} (q(X \to AB) \cdot \pi$[i, s, A] $\cdot \pi$[s+1, j, B])

The Full Dynamic Programming Algorithm

Input: a sentence $s = x_1 \dots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.

Initialization:

For all $i \in \{1 \dots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \dots (n-1)$ l is the length of segment
  - For $i = 1 \dots (n-l)$ i is the index of start of segment
    - Set $j = i + l$ j is the index of the end of segment
    - For all $X \in N$, calculate s is the index of split

$$\pi(i, j, X) = \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z))$$

and

$$bp(i, j, X) = \arg \max_{\substack{X \rightarrow YZ \in R, \\ s \in \{i \dots (j-1)\}}} (q(X \rightarrow YZ) \times \pi(i, s, Y) \times \pi(s+1, j, Z))$$

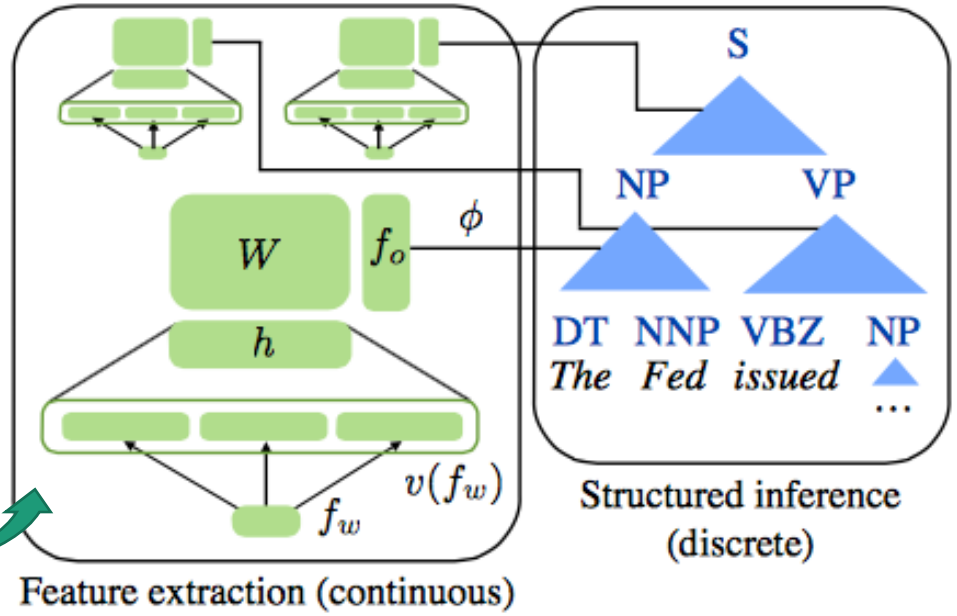
Neural CRF Parsing

- Defined Anchored rule as a tuple (r, s), where r is an indicator of the rule's identity and s = (i,j,k) indicates the span (i,k) and split point j of the rule. $\boldsymbol{w}$ is the word sequence

$$P(T|\boldsymbol{w}) \propto \exp\left(\sum_{(r,s)\in T} \phi(\boldsymbol{w}, r, s)\right)$$

$$\phi_{sparse}(\boldsymbol{w}, r, s; W) = f_s(\boldsymbol{w}, s)^T W f_o(r),$$

$$\phi_{neural}(\boldsymbol{w}, r, s; W) = g(Hv(f_s(\boldsymbol{w}, s)))^T W f_o(r)$$



- Predict score of each span using FFNN
- Do discrete structured inference using CKY

Grammar as a Foreign Language

- Syntactic constituency parsing can be formulated as a sequence-to-sequence problem if we linearize the parse tree, then we can apply standard seq2seq model (maybe with attention) for sentence parsing.

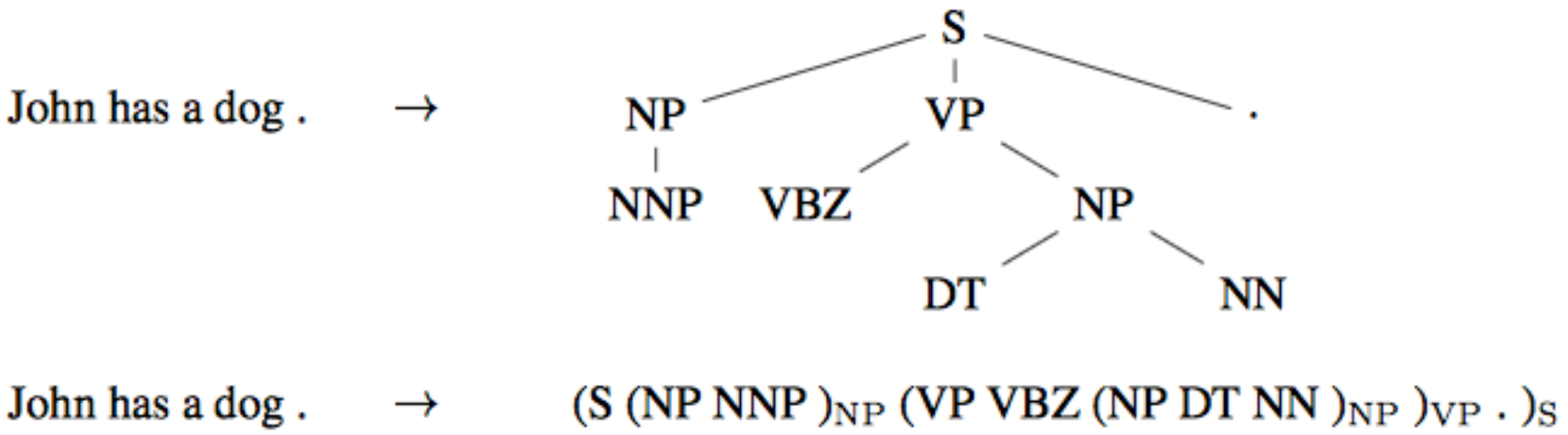


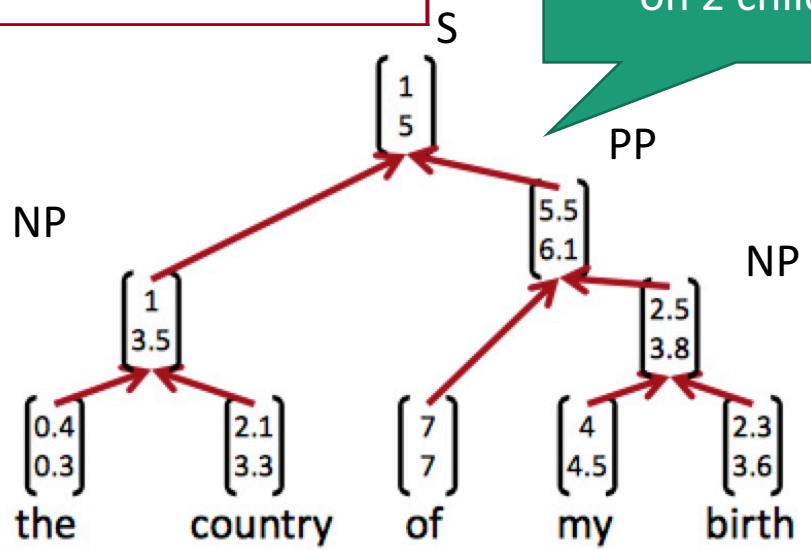
Figure 2: Example parsing task and its linearization.

Map phrases into a vector space:

Compositional Vector Grammar (CVG)

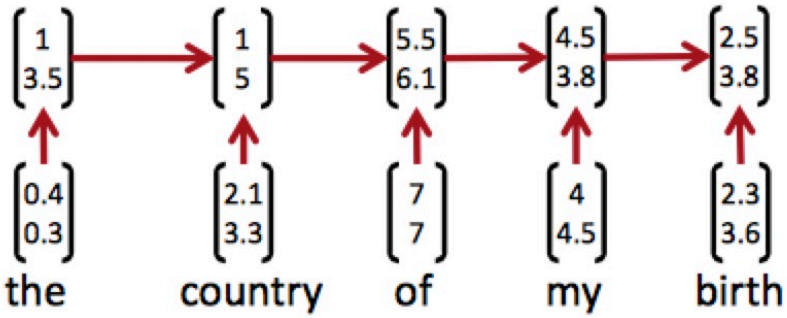
The meaning (vector) of a sentence is determined by
(1) the meanings of its words (2) the rules that combine them.

Calculate via a NN (same for all combinations) based on 2 children.



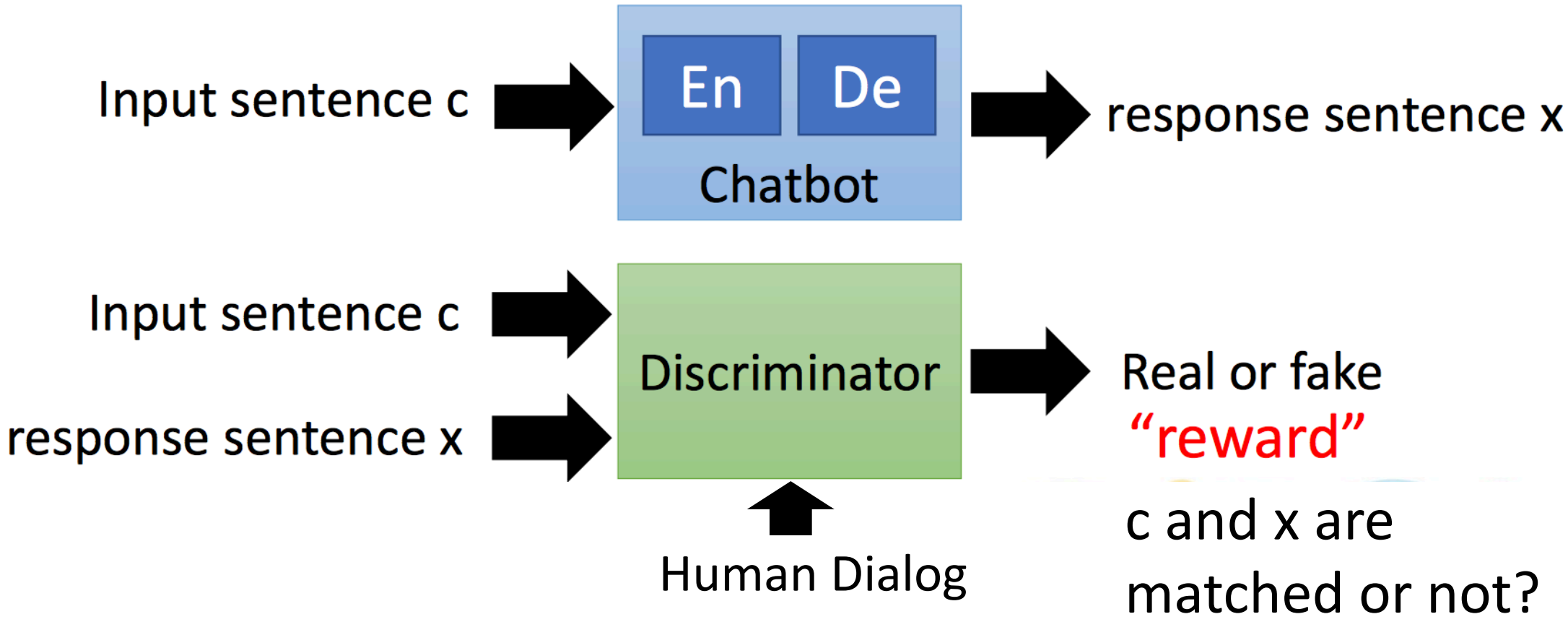
- Recursive neural nets require a tree structure. recursion is natural for describing language. Works better for some tasks to use grammatical tree structure

- Recurrent neural nets cannot capture phrases without prefix context and often capture too much of last words in final vector

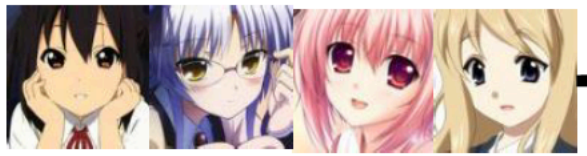


GAN and it’s application in NLP

Conditional GAN



GAN



High score for real,
low score for generated

D

Find a G whose output
obtains large score from D

G

IRL

Expert

$\{\hat{\tau}_1, \hat{\tau}_2, \cdots, \hat{\tau}_N\}$

$\{\tau_1, \tau_2, \cdots, \tau_N\}$

Larger reward for $\hat{\tau}_n$,
Lower reward for τ

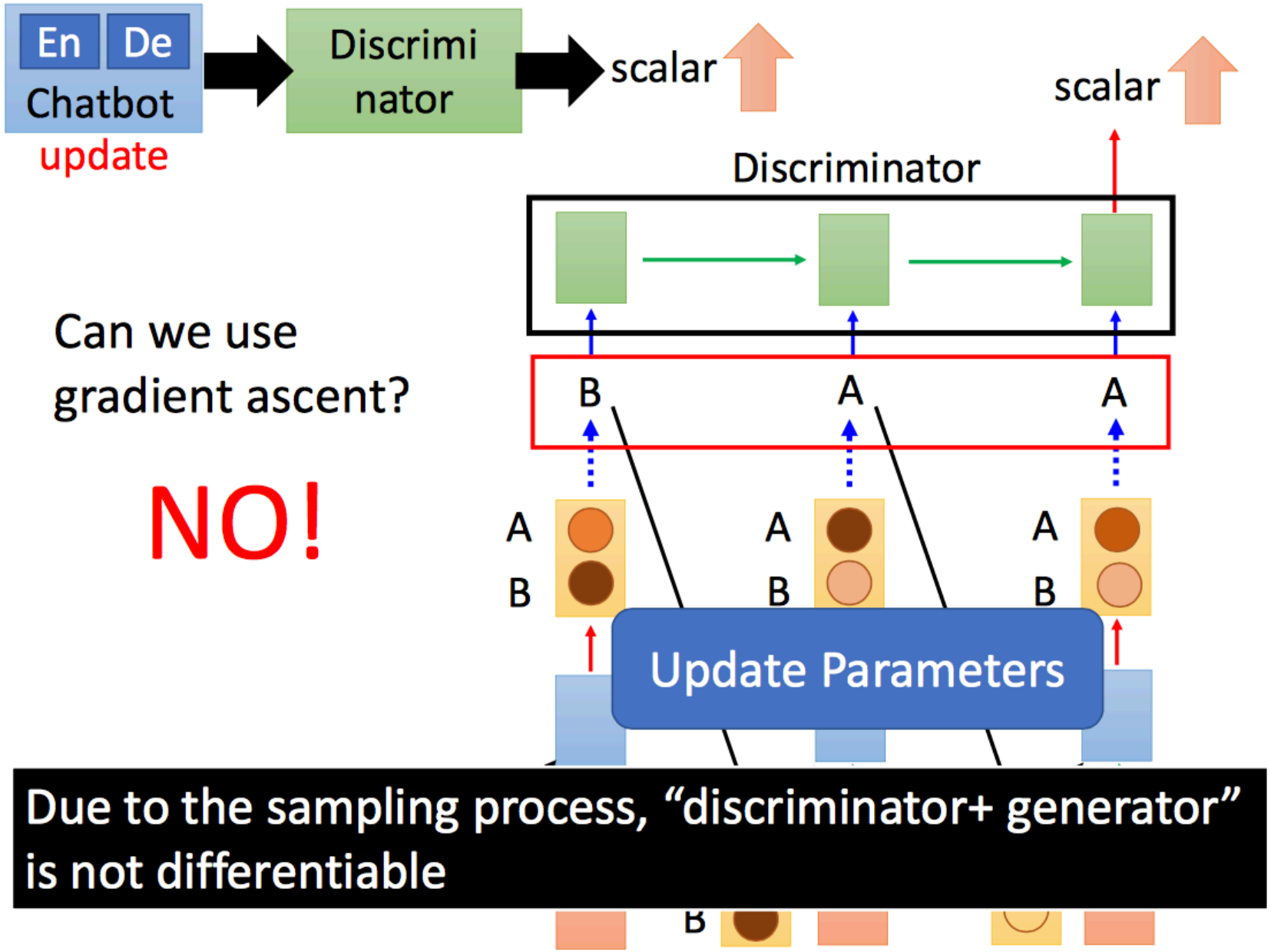
Reward
Function

Find a Actor obtains
large reward

Actor

Inverse reinforcement learning

Using the reward function to find the ***optimal actor***.
Modeling reward can be easier.
Simple reward


En
De
Chatbot
update
Discrimi
nator
scalar
scalar
Discriminator
Can we use
gradient ascent?
NO!
B
A
A
A
B
A
B
A
B
Update Parameters
Due to the sampling process, “discriminator+ generator”
is not differentiable

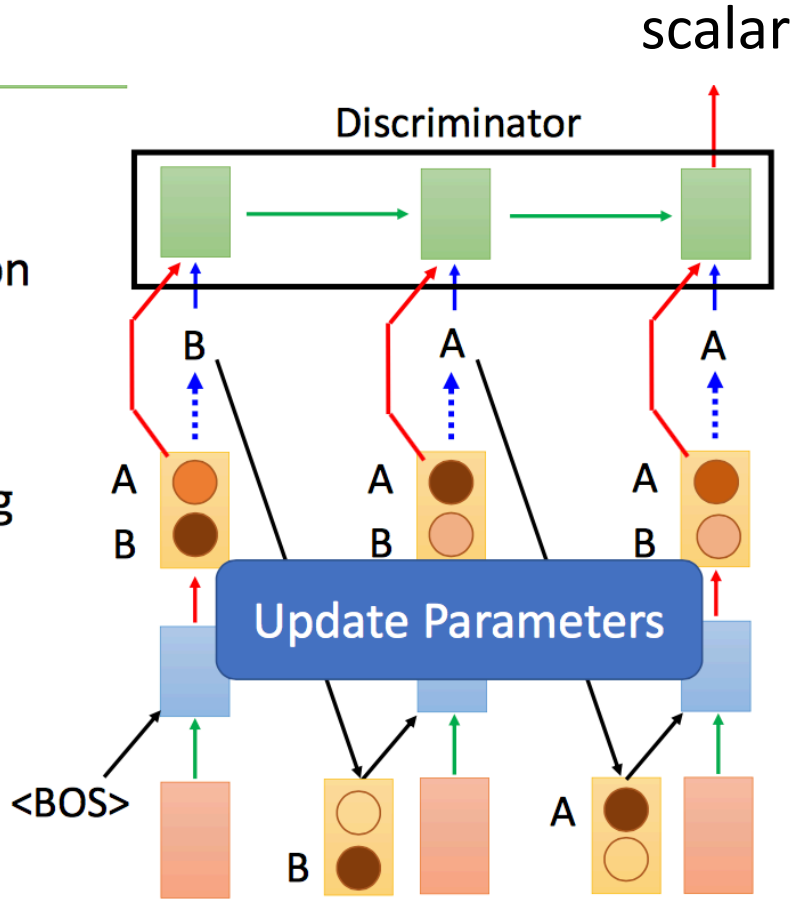
Continuous input for discriminator

update

Use the distribution as the input of discriminator

Avoid the sampling process

We can do backpropagation now.



What is the problem?

- Real sentence

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Discriminator can immediately find the difference.

- Generated

0.9	0.1	0.1	0	0
0.1	0.9	0.1	0	0
0	0	0.7	0.1	0
0	0	0.1	0.8	0.1
0	0	0	0.1	0.9

Can never be 1-of-N

WGAN is helpful

Wasserstein GAN (WGAN)

- Wasserstein distance: the smallest average distance the earth mover has to move the earth.

Average distance of a plan γ:

$$B(\gamma) = \sum_{x_p, x_q} \gamma(x_p, x_q) \|x_p - x_q\|$$

Earth Mover's Distance:

$$W(P, Q) = \min_{\gamma \in \Pi} B(\gamma)$$

The best plan

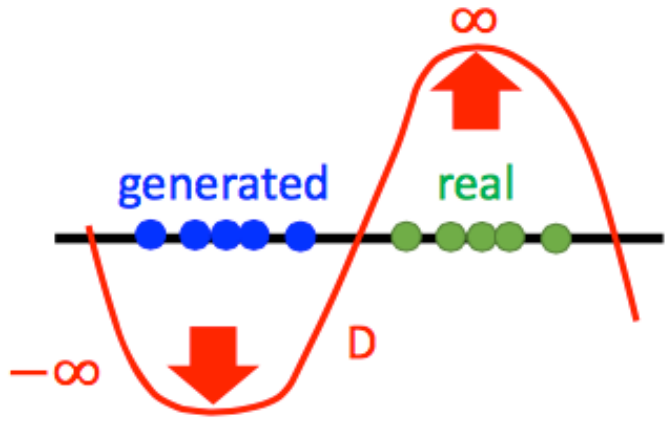
Evaluate wasserstein distance between P_{data} and P_G

$$V(G, D) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

D has to be smooth enough.

Without the constraint, the training of D will not converge.

Keeping the D smooth forces D(x) become ∞ and −∞



Improved WGAN (WGAN-GP)

$$V(G,D) = \max_{D \in 1-Lipschitz} \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)]\}$$

A differentiable function is 1-Lipschitz if and only if it has gradients with norm less than or equal to 1 everywhere.

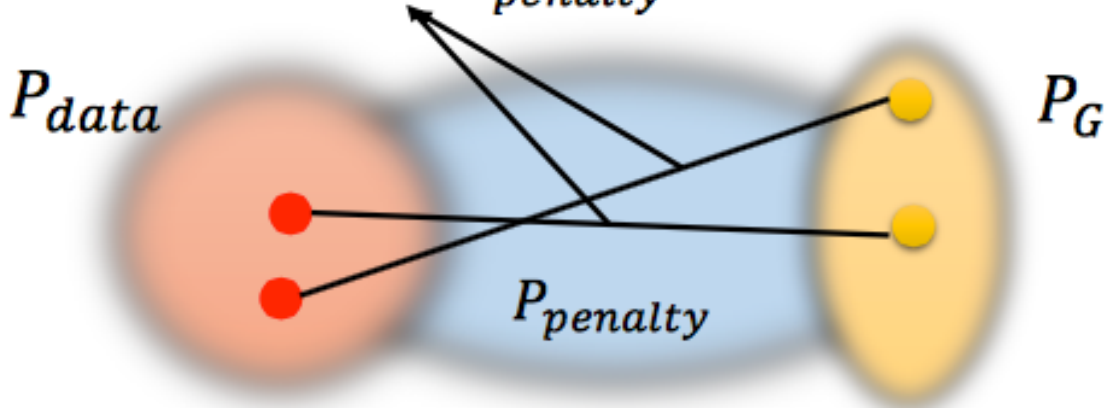
$$D \in 1-Lipschitz \iff \|\nabla_x D(x)\| \leq 1 \text{ for all x}$$

Prefer $\|\nabla_x D(x)\| \leq 1$ for x sampling from $x \sim P_{penalty}$

It is hard to check this on all possible X

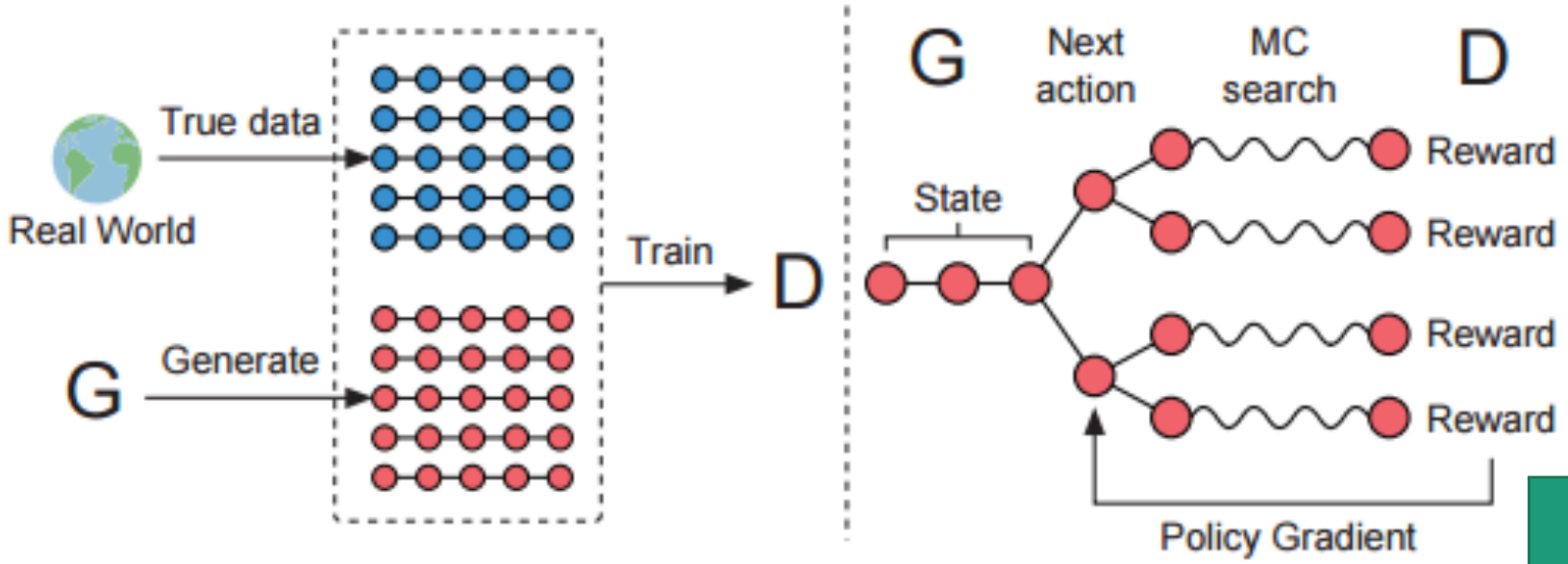
$$V(G,D) \approx \max_D \{E_{x \sim P_{data}}[D(x)] - E_{x \sim P_G}[D(x)] - \lambda E_{x \sim P_{penalty}}[\max(0, \|\nabla_x D(x)\| - 1)]\}$$

$$(\|\nabla_x D(x)\| - 1)^2$$



"Simply penalizing overly large gradients also works in theory, but experimentally we found that this approach converged faster and to better optima."

SeqGan: Sequence Generative Adversarial Nets with Policy Gradient



$$J(\theta) = \mathbb{E}[R_T|s_0, \theta] = \sum_{y_1 \in \mathcal{Y}} G_\theta(y_1|s_0) \cdot Q_{D_\phi}^{G_\theta}(s_0, y_1)$$

$$Q_{D_\phi}^{G_\theta}(s = Y_{1:t-1}, a = y_t) =$$

$$\begin{cases} \frac{1}{N}\sum_{n=1}^{N} D_\phi(Y_{1:T}^n),\ Y_{1:T}^n \in \mathbf{MC}^{G_\beta}(Y_{1:t}; N) & \text{for} \quad t < T \\ D_\phi(Y_{1:t}) & \text{for} \quad t = T, \end{cases}$$

Use Monte Carlo search to sample unknown tokens and get the reward

For the last tokens get direct reward from discriminator

Ongoing work

- Improved sequence generation for semi-supervised text mining.

  GAN-based semi-supervised learning:

$$\max_D \mathbb{E}_{x,y\sim\mathcal{L}} \log P_D(y|x, y \le K) + \mathbb{E}_{x\sim p} \log P_D(y \le K|x) + \mathbb{E}_{x\sim p_G} \log P_D(K+1|x)$$

Labeled data
The cross entropy

Unlabeled data
The unlabeled data must belong to one of the k-th class

Fake data
The fake data belongs to the (k+1)-th class

Can be generated via Seqgan

Salimans, Tim, et al. "Improved techniques for training gans." *Advances in Neural Information Processing Systems*. 2016.

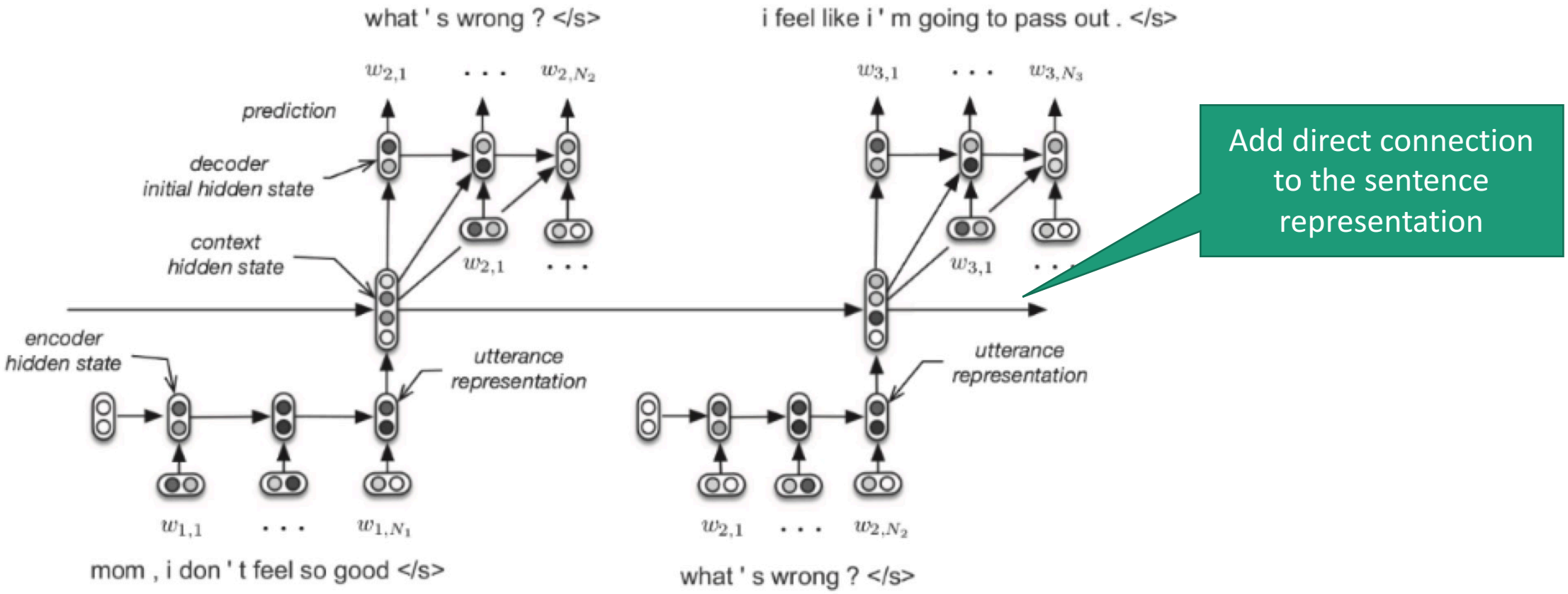
Models of Dialog

Models of Dialog

- Generation-based Models: Train phrase-based machine translation (encoder-decoders), but more dependent on global coherence, have much more varied responses.
- Retrieval-based Chat : given an utterance, find the most similar in the database and return it. The response has particular format. Much more safe than Generation-based Models.

Generation-based Models

- Hierarchical Encoder-decoder Model (serban et al. 2016): Also have utterance-level RNN track overall dialog state



Retrieval-based Chat

- Basic idea: given an utterance, find the most similar in the database and return it