Lecture 8: Learning to Search – Imitation Learning in NLP

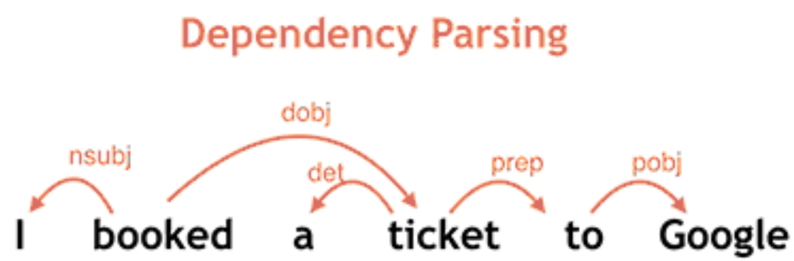
Kai-Wei Chang
CS @ UCLA
kw@kwchang.net

Couse webpage: https://uclanlp.github.io/CS269-17/

UCLA ENGINEERING
Computer Science

Learning to search approaches
Shift-Reduce parser

- Maintain a buffer and a stack
- Make predictions from left to right
- Three (four) types of actions:
  Shift, Reduce, Left, Right



Credit: Google research blog

Structured Prediction as a Search problem

- Decomposition of y often implies an ordering
  $\Rightarrow$ a sequential decision making process

I	can	can	a	can
Pro	Md	Vb	Dt	Nn

Notations

- Input: $x \in X$
- Truth: $\mathrm{y}^* \in Y(x)$
- Predicted: $h(x) \in Y(x)$
- Loss: $loss(y, y^*)$

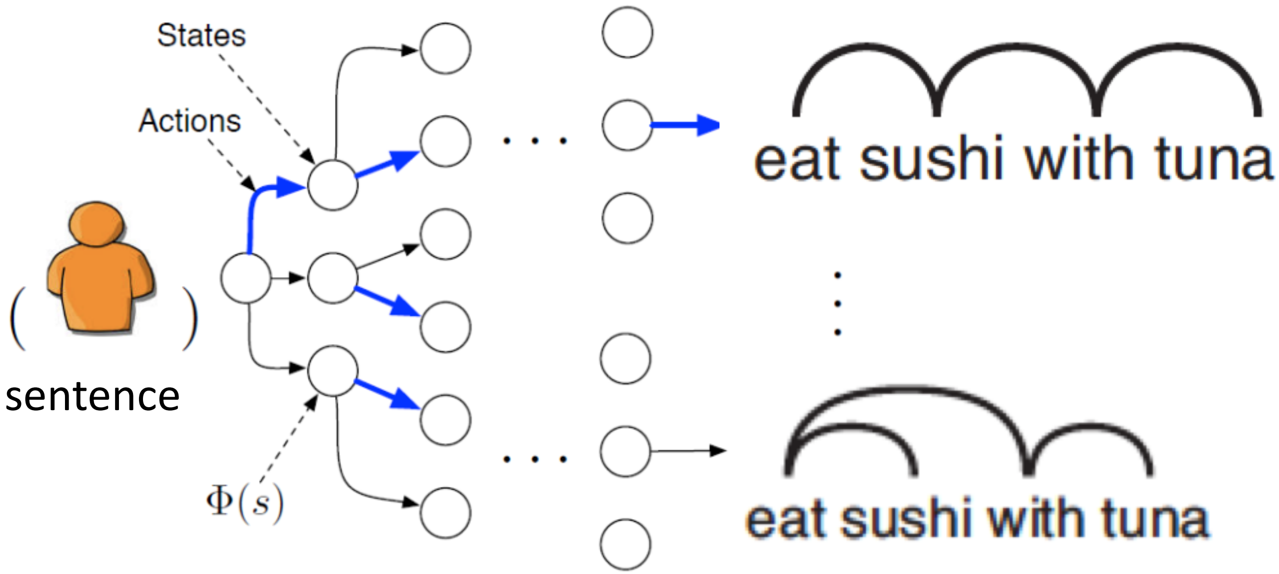
I	can	can	a	can
Pro	Md	Vb	Dt	Nn
Pro	Md	Md	Dt	Vb
Pro	Md	Md	Dt	Nn
Pro	Md	Nn	Dt	Md
Pro	Md	Nn	Dt	Vb

Goal: make joint prediction to minimize a joint loss

find $h \in H$ such that $h(\mathrm{x}) \in Y(X)$
minimizing $E_{(x,y)\sim D}\left[loss\left(y, h(x)\right)\right]$ based on N
samples $(x_n, y_n) \sim D$

Credit Assignment Problem

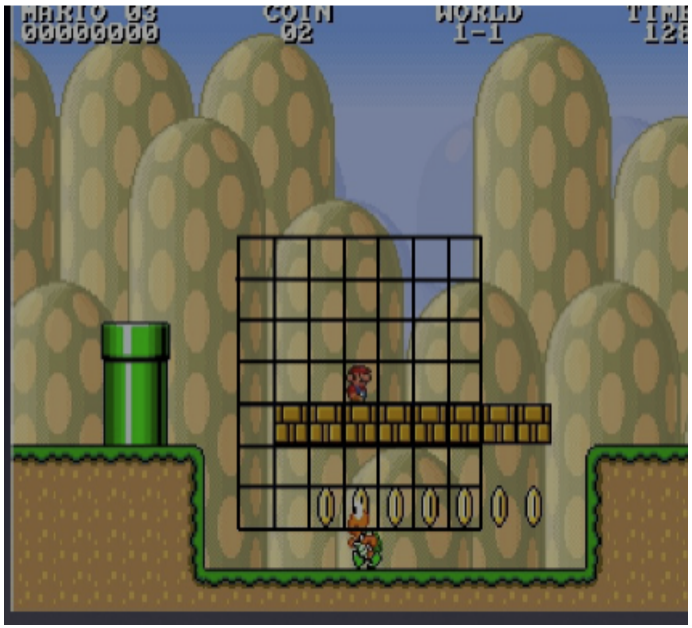
When making a mistake, which local decision should be blamed?



An Analogy from Playing Mario

From Mario AI competition 2009

Input:



Output:

Jump in {0,1}
Right in {0,1}
Left in {0,1}
Speed in {0,1}

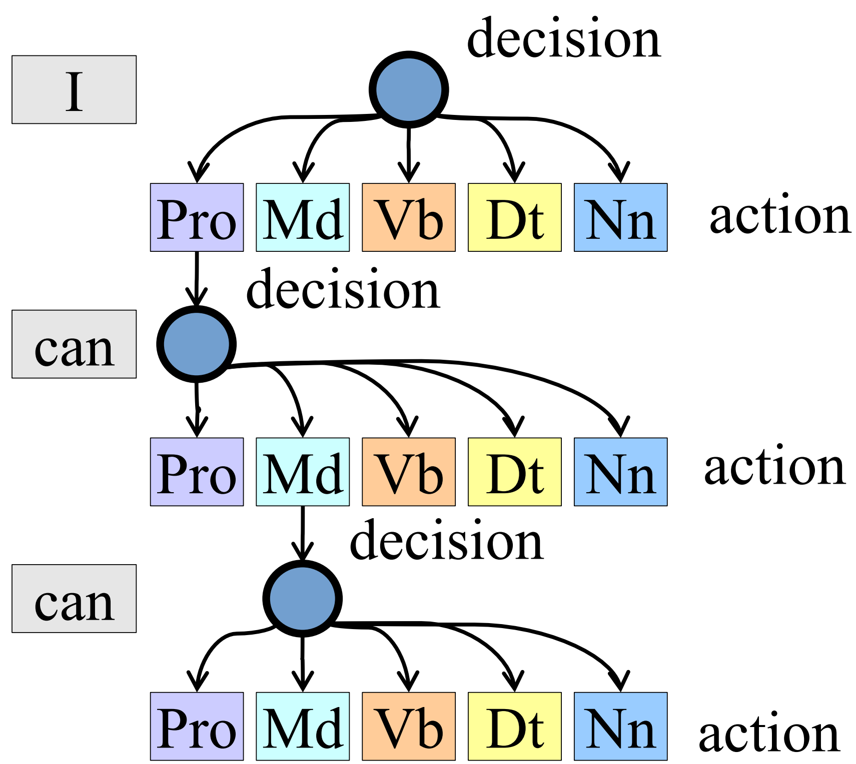
Extrac
from l
(14 bi

High level goal:
Watch an expert play and
learn to mimic her behavior

Video credit: Stéphane Ross, Geoff Gordon and Drew Bagnell

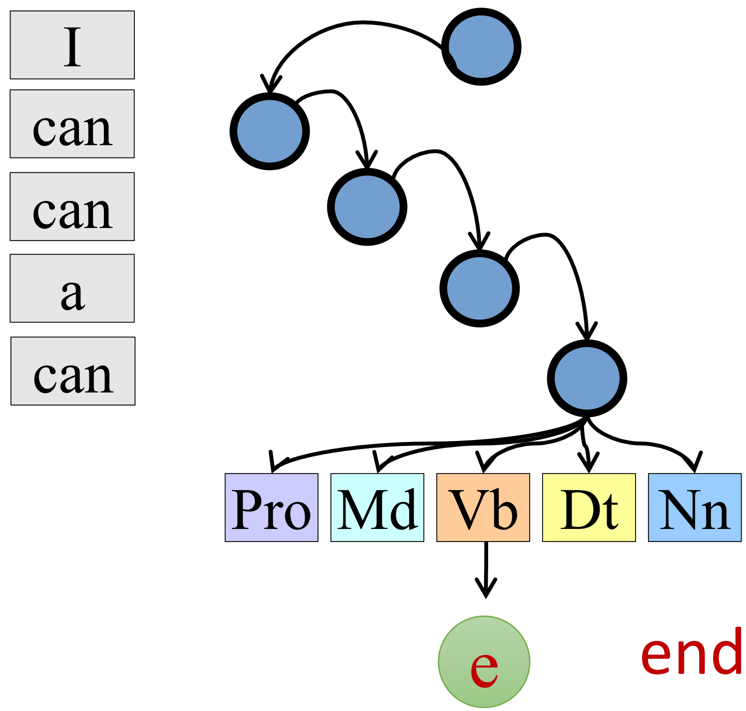
Example of Search Space

I	can	can	a	can
Pro	Md			



Example of Search Space

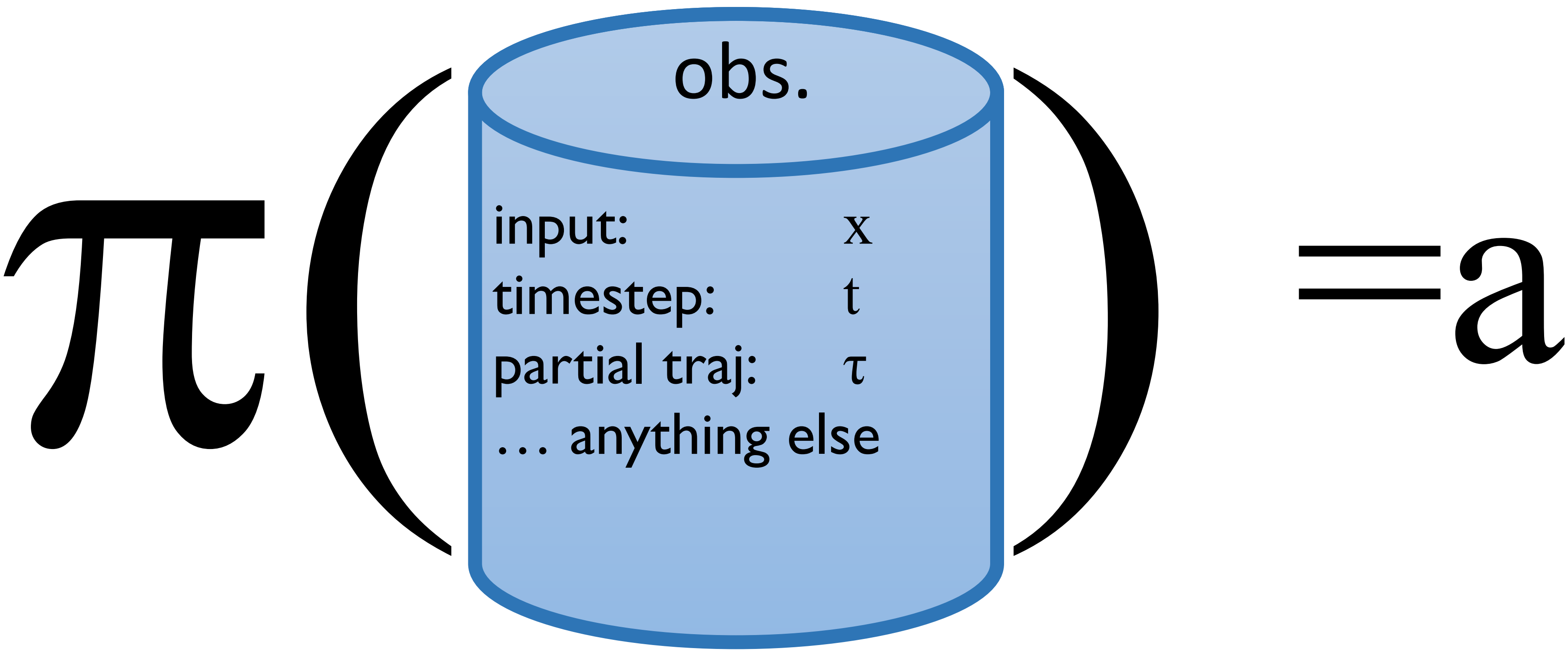
I	can	can	a	can
Pro	Md	Vb	Dt	Vb



Encodes an output
$\hat{y} = \hat{y}(e)$
from which
$loss(y, \hat{y})$
can be computed
(at training time)

Policies

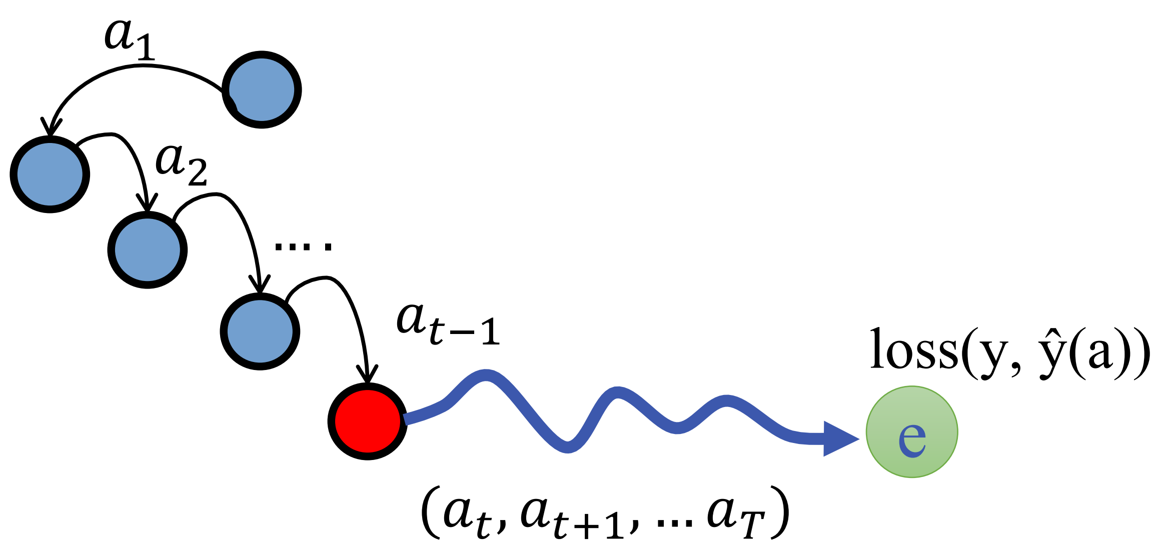
- A policy maps observations to actions



Labeled data → Reference policy

- Given partial traj. $a_1, a_2, \ldots a_{t-1}$ and true label y, the minimum achievable loss is

$$(a_t^*, a_{t+1}^*, \ldots a_T^*) = \arg \min_{(a_t, a_{t+1}, \ldots a_T)} \text{loss}(y, \hat{y}(a))$$



Labeled data → Reference policy

- Given partial traj. $a_1, a_2, \ldots a_{t-1}$ and true label y, the minimum achievable loss is

$$(a_t^*, a_{t+1}^*, \ldots a_T^*) = \arg \min_{(a_t, a_{t+1}, \ldots a_T)} \text{loss}(y, \hat{y}(a))$$

- The optimal action is the corresponding a_t^*
- The optimal policy is the policy that always selects the optimal action
- Reference policy can be constructed by the gold label in the training phase

Ingredients for learning to search

Structured Learning

Input: $x \in X$

Truth: $y \in Y(x)$

Outputs: $Y(x)$

Loss: $loss(y, \hat{y})$

Learning to Search

Search Space:

- state: $s \in S$
- action: $a \in A(a)$
- end state $e \in S$

Policies: $\pi(s) \rightarrow a$

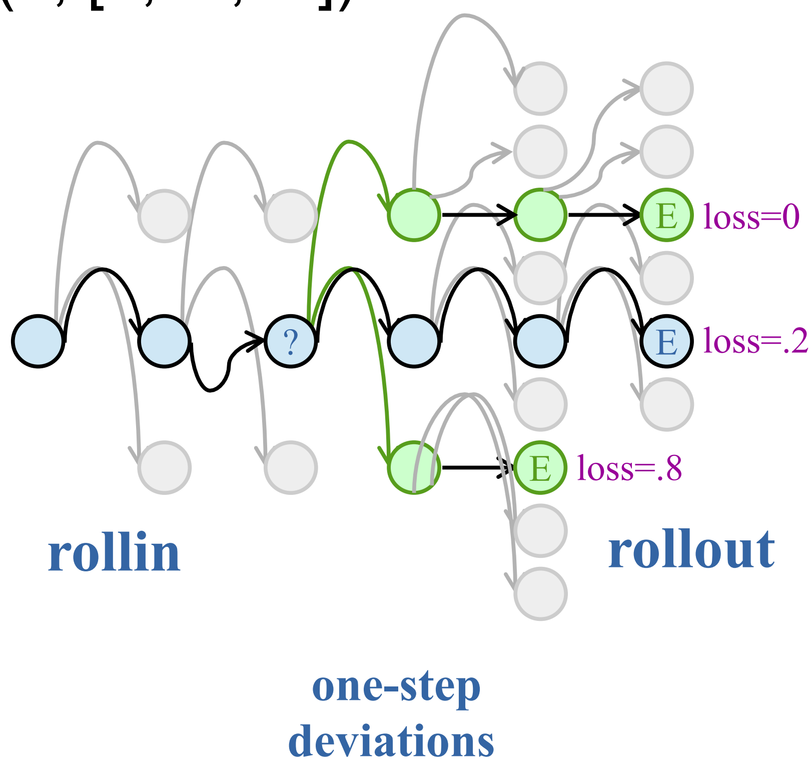
Reference policy: π^{ref}

A Simple Approach

- Collect trajectories from expert π^{ref}
- Store as dataset $D = \{(o, \pi^{ref}(o)) | o \sim \pi^{ref}\}$
- Train classifier π on D
- Let π play the game!

Learning a Policy[Chang+ 15, Ross+15]

- At “?” state, we construct a cost-sensitive multi-class example (?, [0, .2, .8])



Example: Sequence Labeling

Receive input:

$x =$ the monster ate the sandwich
$y =$ Dt Nn Vb Dt Nn

Make a sequence of predictions:

$x =$ the monster ate the sandwich
$\hat{y} =$ Dt Dt Dt Dt Dt

Pick a timestep and try all perturbations there:

$x =$ the monster ate the sandwich
$\hat{y}_{Dt} =$ Dt Dt Vb Dt Nn $l=1$
$\hat{y}_{Nn} =$ Dt Nn Vb Dt Nn $l=0$
$\hat{y}_{Vb} =$ Dt Vb Vb Dt Nn $l=1$

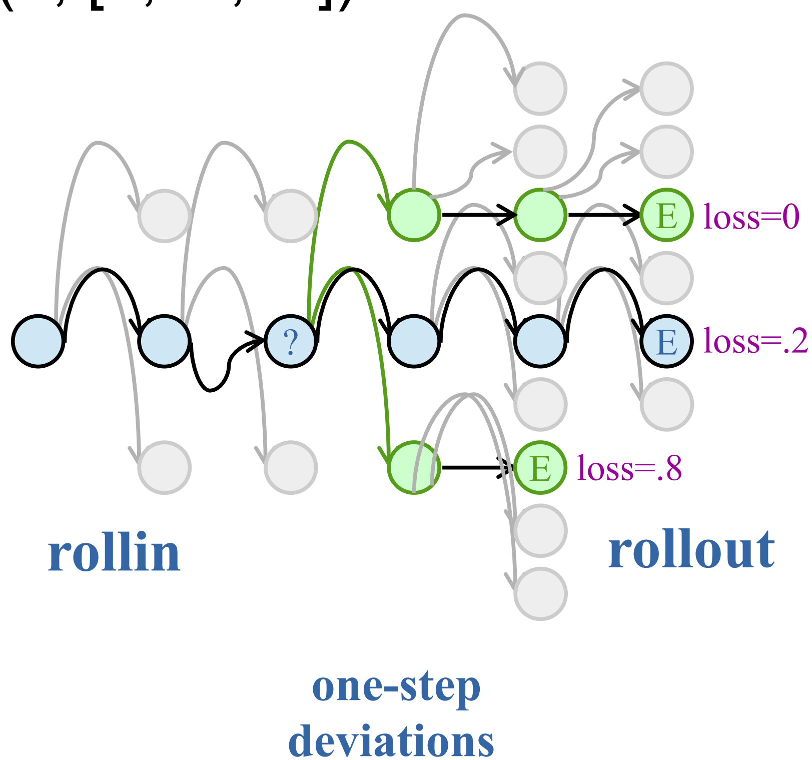
Compute losses and construct example:

({ w=monster, p=Dt, …}, [1,0,1])

UCLA ENGINEERING
Computer Science

Learning a Policy[Chang+ 15, Ross+15]

❖ At "?" state, we construct a cost-sensitive multi-class example (?, [0, .2, .8])



Analysis

roll-out → ↓ roll-in	Reference	Mixture	Learned
Reference	Inconsistent		
Learned	No local opt	Good	RL

Mixture: w.p. β use Ref, else use Learned.

[ICML 15]: Learning to search better than your teacher

UCLA ENGINEERING
Computer Science

Analysis

roll-out → / ↓ roll-in	Reference	Mixture	Learned
Reference	Inconsistent		
Learned	No local opt	Good	RL

Mixture: w.p. β use Ref, else use Learned.

Roll-in with Ref:

unbounded structured regret

UCLA ENGINEERING
Computer Science

Analysis

roll-out → ↓ roll-in	Reference	Mixture	Learned
Reference	Inconsistent		
Learned	$\boxed{\text{No local opt}}$	Good	RL

Mixture: w.p. β use Ref, else use Learned.

Roll-out with Ref:

no local optimal if reference is sub-optimal

Analysis

roll-out → ↓ roll-in	Reference	Mixture	Learned
Reference	Inconsistent		
Learned	No local opt	Good	$\boxed{\text{RL}}$

Mixture: w.p. β use Ref, else use Learned.

Roll-in & Roll-out with current policy
ignore Ref ⇒ reinforcement learning

UCLA ENGINEERING Computer Science

Analysis

roll-out → ↓ roll-in	Reference	Mixture	Learned
Reference	Inconsistent		
Learned	No local opt	Good	RL

Mixture: w.p. β use Ref, else use Learned.

Minimizes a combination of regret to Ref and regret to its own one-step deviations.

- Competes with Ref when Ref is good.
- Competes with local deviations to improve on suboptimal Ref

UCLA ENGINEERING
Computer Science

How to Program?

Sequential_RUN($examples$)

1: **for** $i = 1$ **to** len($examples$) **do**
2: $\quad prediction \leftarrow$ **predict**($examples[i]$, $examples[i].label$)
3: $\quad$ **loss**($prediction \neq examples[i].label$)
4: **end for**

Decoder + loss + reference advice

- Sample codes are available in VW