Imitation learning for structured prediction in natural language processing

Andreas Vlachos (http://andreasvlachos.github.io), Gerasimos Lampouras (http://glampouras.github.io)
{a.vlachos,g.lampouras}@sheffield.ac.uk
Department of Computer Science
University of Sheffield

Sebastian Riedel (http://www.riedelcastro.org/)
s.riedel@ucl.ac.uk
Department of Computer Science
University College London

sheffieldnlp.github.io/ImitationLearningTutorialEACL2017/ (http://sheffieldnlp.github.io/ImitationLearningTutorialEACL2017/)

Your name sounds familiar

Imitation is an advanced behavior whereby an individual observes and replicates another's behavior. Imitation is also a form of social learning that leads to the development of traditions, and ultimately our culture.

(https://en.wikipedia.org/wiki/Imitation (https://en.wikipedia.org/wiki/Imitation))

Robotics

Legged locomotion
(Rattlif et al., 2006 (https://papers.nips.cc/paper/3154-boosting-structured-prediction-for-imitation-learning.pdf))

(http://www.bostondynamics.com/robot_littledog.html)

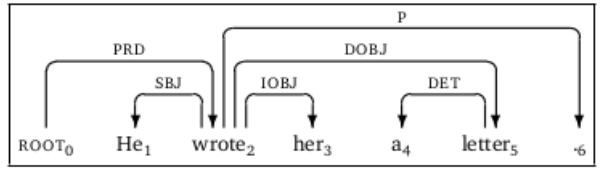
Autonomous helicopter flight
(Coates et al., 2008 (http://heli.stanford.edu/papers/coatesabbeelng_icml2008.pdf))

(http://www.theverge.com/2016/1/6/10721654/electric-self-flying-quadcopter-ehang-184-ces-2016)

And more: outdoor navigation (Silver et al., 2008 (http://www.roboticsproceedings.org/rss04/p34.pdf)), Super-Mario (Ross et al., 2011 (https://www.cs.cmu.edu/~sross1/publications/Ross-AIStats11-NoRegret.pdf)), autonomous driving (Zhang and Cho, 2017 (https://arxiv.org/abs/1605.06450)) ...

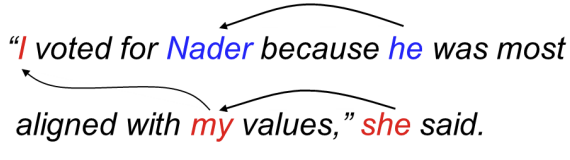
Your name sounds more(!) familiar

Dynamic oracles for parsing
(Goldberg and Nivre, 2012 (http://www.aclweb.org/anthology/C12-1059),
Ballesteros et al., 2016 (https://arxiv.org/pdf/1603.03793.pdf))

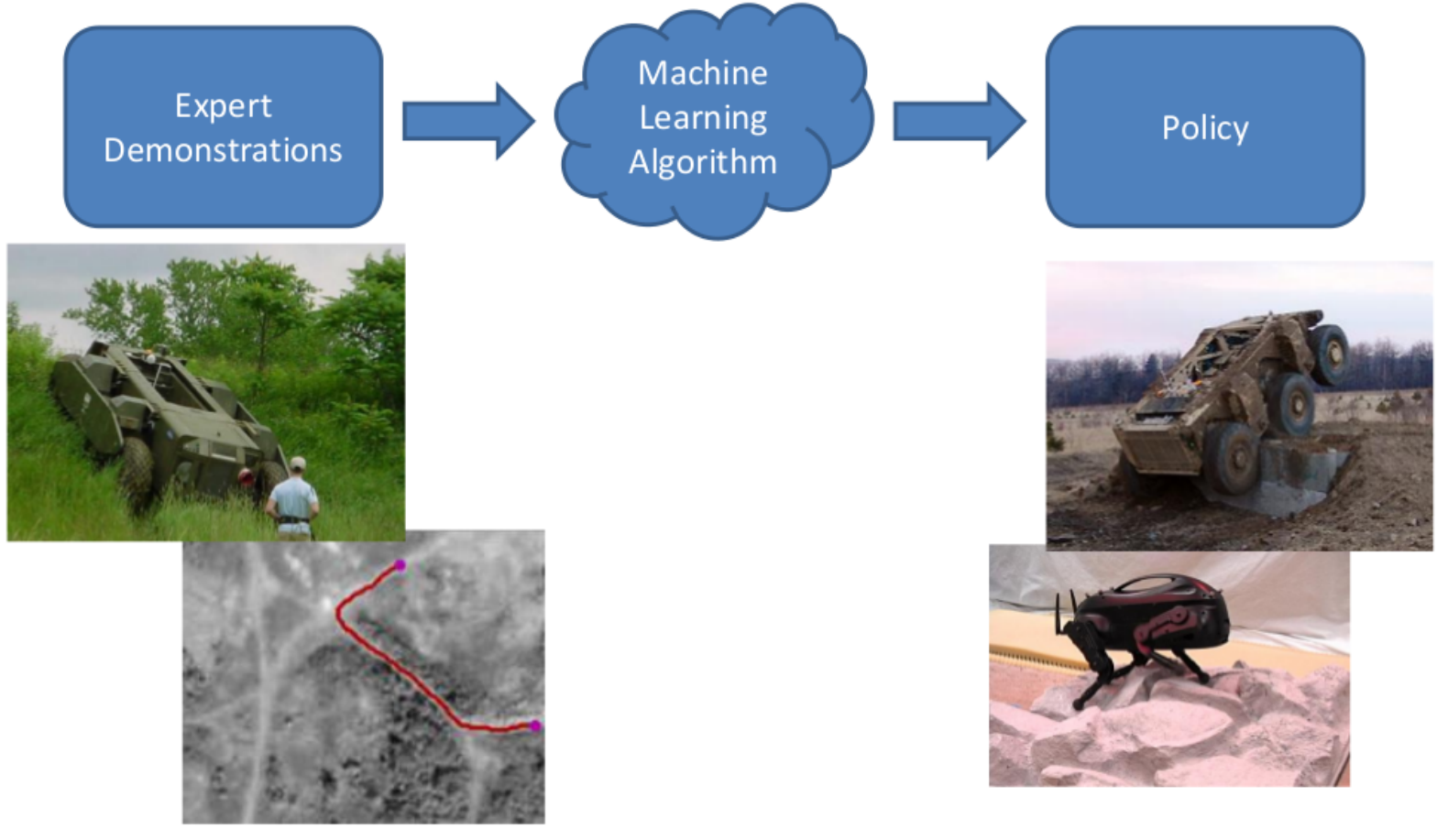


Incremental coreference resolution
(Clark and Manning, 2015 (http://cs.stanford.edu/people/kevclark/resources/clark-manning-acl15-entity.pdf))

(http://nlp.stanford.edu/projects/coref.shtml)



Imitation Learning in a nutshell



(http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats11-Slides.pdf)

Meta-learning: better model (≈policy) by generating better training data from demonstrations.

Part 1: Imitation Learning for Structured Prediction

- Basics:
  - structured prediction
  - different types of supervision and loss functions
  - cost-sensitive classification
- Imitation learning algorithms:
  - Dataset Aggregation (DAgger)
  - Learning to Locally Optimal Learning to Search (LOLS)
- Interpretations and connections
  - Reinforcement Learning
  - Recurrent Neural Network training
  - Adversarial training

Part 2: NLP Applications and practical advice

- Applications:
  - Dependency parsing
  - Semantic parsing:
  - Natural language generation
- Practical advice
  - expert policy definition
  - accelerating cost estimation
  - trouble-shooting

Outcomes

- Understanding of how IL works via unified algorithmic presentations
- Clarification of the connections to other learning fr ameworks
- Familiarization with representativ e NLP applications
- Recognizing when and how to apply IL

Preliminaries

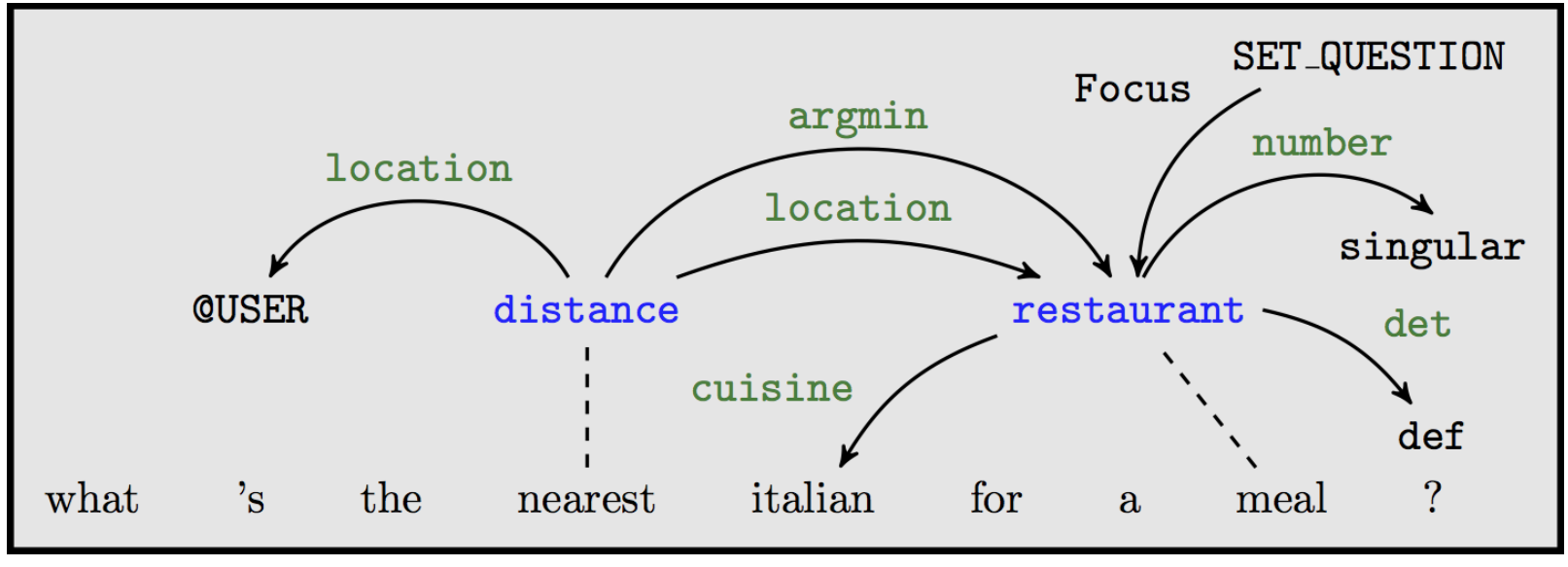
Structured prediction

I	studied	in	London	with	Sebastian	Riedel
PRP	VBD	IN	NNP	IN	NNP	NNP
O	O	O	B-LOC	O	B-PER	I-PER

- part of speech (PoS) tagging
- named entity recognition (NER)

Input: a sentence $\mathbf{x} = [x_1 \dots x_N]$
Output: a sequence of labels $\mathbf{y} = [y_1 \dots y_N] \in \mathcal{Y}^N$

Structured prediction



Semantic parsing, but also syntactic parsing, semantic role labeling, question answering over knowledge bases, etc.)

Input: a sentence $\mathbf{x} = [x_1 \ldots x_N]$
Output: a meaning representation graph $\mathbf{G} = (V, E) \in \mathcal{G}_{\mathbf{x}}$

Structured prediction

INPUT:

```
predicate= INFORM
name = "The Saffron Brasserie"
type = placetoeat
eattype = restaurant
area = riverside, "addenbrookes"
near = "The Cambridge Squash", "The Mill"
```

OUTPUT:

The Saffron Brasserie is a restaurant at the side of the river near the Cambridge Squash and the Mill in the area of Addenbrookes

Natural language generation (NLG), but also summarization, decoding in machine translation, etc.

Input: a meaning representation
Output: a sentence $\mathbf{w} = [w_1 \ldots w_N], w \in \mathcal{V} \cup END, w_N = END$

Two main paradigms

Joint modeling, a.k.a:

- global inference
- structured prediction

Incremental modeling, a.k.a:

- local
- greedy
- pipeline
- transition-based
- history-based

Joint modeling

A model (e.g. conditional random field) that scores complete outputs (e.g. label sequences):

$$\hat{\mathbf{y}} = \hat{y}_1 \ldots \hat{y}_N = \underset{Y \in \mathcal{Y}^N}{\arg\max} f(y_1 \ldots y_N, \mathbf{x})$$

- no error propagation
- exhaustive exploration of the search space
- large/complex search spaces are challenging
- efficient dynamic programming restricts modelling flexibility (i.e. Markov assumptions)

Incremental modeling

A classifier that predicts one label at a time giv en the previous predictions:

$$\hat{\mathbf{y}} = \begin{aligned} \hat{y}_1 &= \arg\max_{y \in \mathcal{Y}} f(y, \mathbf{x}), \\ \hat{y}_2 &= \arg\max_{y \in \mathcal{Y}} f(y, \mathbf{x}, \hat{y}_1), \cdots \\ \hat{y}_N &= \arg\max_{y \in \mathcal{Y}} f(y, \mathbf{x}, \hat{y}_1 \ldots \hat{y}_{N-1}) \end{aligned}$$

- use our favourite classifier
- no restrictions on features
- prone to error propagation (i.i.d. assumption brok en)
- local model not trained wrt the task-level loss

Imitation learning

Improve incremental modeling to:

- address error-propagation
- train wrt the task-level loss function

Meta-learning: use our favourite classifier and features, but generate better (non-i.i.d.) training data

But let's see some basic concepts first

Transition system

The **actions** $\mathcal{A}$ the classifier f can predict and their effect on the **state** which keeps track of the prediction: $S_{t+1} = (S_1, \alpha_1 \ldots \alpha_t)$

Input: $\mathbf{x}$
$state\ S_1 = initialize(\mathbf{x})$; $timestep\ t = 1$
while S_t not final **do**
  $action\ \alpha_t = \arg\max_{\alpha \in \mathcal{A}} f(\alpha, \mathbf{x})$
  $S_{c+1} = update(\alpha_t, S_t)$; $t = t + 1$
Output: $S_{final} = S_t$

- **PoS/NER tagging?** for each word in the sentence, left-to-right, predict a P oS tag which is added to the output
- **NLG?** predict a word from the vocabulary that is added to the output until the `END`

Task loss

Given S_{final}, how does it compare to the gold standard $\mathbf{y}$?

$$loss = L(S_{final}, \mathbf{y}) \geq 0$$

- **PoS tagging?** Hamming loss: number of incorrect tags
- **NER?** number of false positives and false negatives
- **NLG?** BLEU: % of n-grams predicted present in the gold reference(s) ($L = 1 - BLEU(s_{final}, \mathbf{y})$)

Goal: models minimizing the loss on unseen test data

Decomposable losses

A loss is **decomposable** if it is the sum of the losses for each action α_t independently of the future actions $[\alpha_{t+1} \dots \alpha_T]$:

$$L(S, \mathbf{y}) = \sum_{t=1}^{T} \ell(\alpha_t, \mathbf{y}, [\alpha_1 \dots \alpha_{t-1}])$$

I	studied	in	London	with	Sebastian	Riedel
PRP	VBD	IN	NNP	IN	NNP	NNP
O	O	O	B-LOC	O	B-PER	I-PER

Can we tell $\ell(\alpha_6, \cdot)$ for

- PoS tagging? **Yes!** $\ell(\alpha_6, \cdot) = 0$ no matter α_7
- NER? **No!** If α_7 is
  - I-PER: $\ell(\alpha_6, \cdot) = 0$ (correct)
  - O: $\ell(\alpha_6, \cdot) = 2$ (1 FP and 1 FN)
  - B-*: $\ell(\alpha_6, \cdot) = 3$ (2 FP and 1 FN)

Non-decomposable loss

Is BLEU score decomposable?

$$BLEU([\alpha_1 \dots \alpha_T], \mathbf{y}) = \prod_{n=1}^{N} \frac{\#\text{n-grams} \in ([\alpha_1 \dots \alpha_T] \cap \mathbf{y})}{\#\text{n-grams} \in [\alpha_1 \dots \alpha_T]}$$

No! Assuming N>1 and a word-by-word predictor.

Non-decomposability affects joint models: loss does not always decompose over the graphical model (Tarlow and Zemel, 2012 (http://www.cs.toronto.edu/~dtarlow/tarlow_zemel_aistats12.pdf))

Even F-score for binary classification is non-decomposable (Narasimhan et al., 2015 (http://jmlr.org/proceedings/papers/v37/narasimhana15.pdf))!

Expert policy

Returns the best action at the current state b y looking at the gold standard assuming future actions are also optimal:

$$\alpha^\star = \pi^\star(S_t, \mathbf{y}) = \arg\min_{\alpha \in \mathcal{A}} L(S_t(\alpha, \pi^\star), \mathbf{y})$$

I	studied	in	London	with	Sebastian	Riedel
PRP	VBD	IN	NNP	IN	NNP	NNP

PoS tagging: $\pi^\star(S_t, \mathbf{y}) = \pi^\star(S_t, [y_1 \ldots y_T]) = y_t$

Only available for the training data
(it cheats!): a human teacher
demonstrating how to perform the task

(http://www.salon.com/2016/10/06/what-makes-a-good-teacher-why-certifications-and-standards-dont-guarantee-quality-educators_partner/)

Expert policy

What action should $\pi^\star$ return?

I	studied	in	London	with	Sebastian	Riedel
O	O	O	B-LOC	O	B-PER	I-PER
O	O	O	B-LOC	O	O	O

Takes previous actions into account to be optimal (dynamic)

Finding the optimal action can be expensive; can be sub-optimal

Cost-sensitive classification

Classification: single correct label per-instance, e.g. B-PER

Multi-label classification: multiple correct labels per-instance, e.g. B-OR G, B-LOC

Cost-sensitive classification: each label has a cost, e.g.

O	B-PER	I-PER	B-LOC	I-LOC	B-ORG	I-LOC
1	2	2	0	1	0	1

Cost-sensitive classification

Imitation learning: learn a better model, i.e. action-predicting classifier b y imitating expert demonstrations

- learn that same mistakes are worse than others
- multiple actions can be optimal

Learning with costs:

- sample instances according to their cost to tr ain a classifier (Abe et al., 2004 (http://www.hunch.net/~jl/projects/reductions/mc2/p542-Abe.pdf))
- adjust the updates according the cost in error-driv en learning (Crammer et al. 2006 (http://jmlr.csail.mit.edu/papers/volume7/crammer06a/crammer06a.pdf))

Imitation learning

Imitation learning for part-of-speech tagging

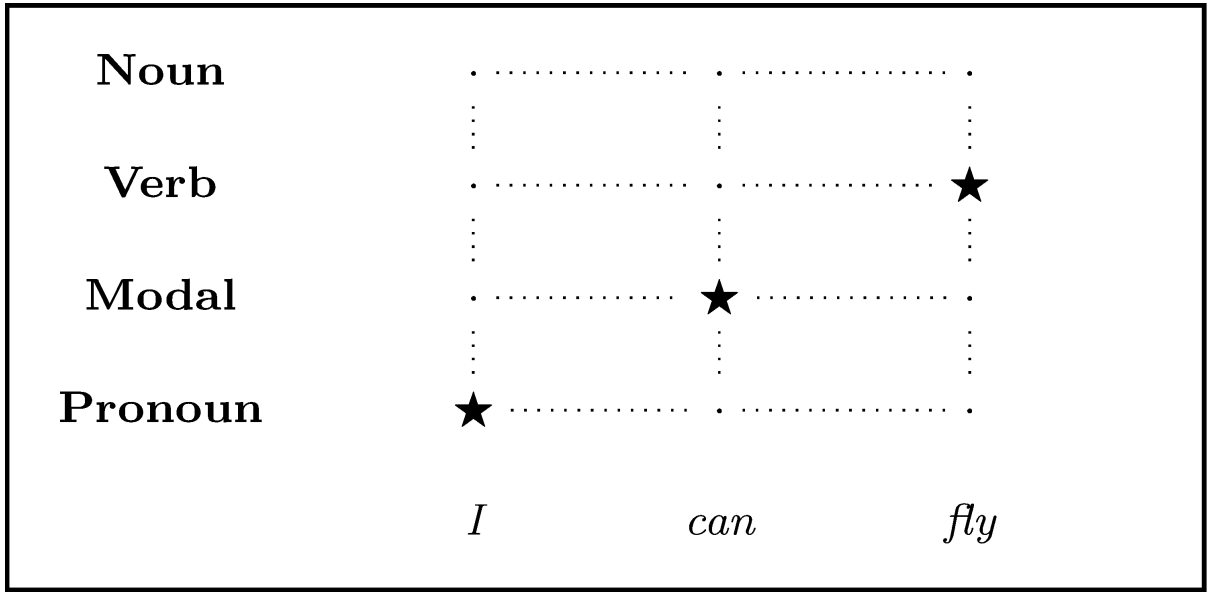
I	**can**	**fly**
Pronoun	Modal	Verb

Task loss: Hamming loss: number of incorrectly predicted tags

Transition system: Tag each token left-to-right

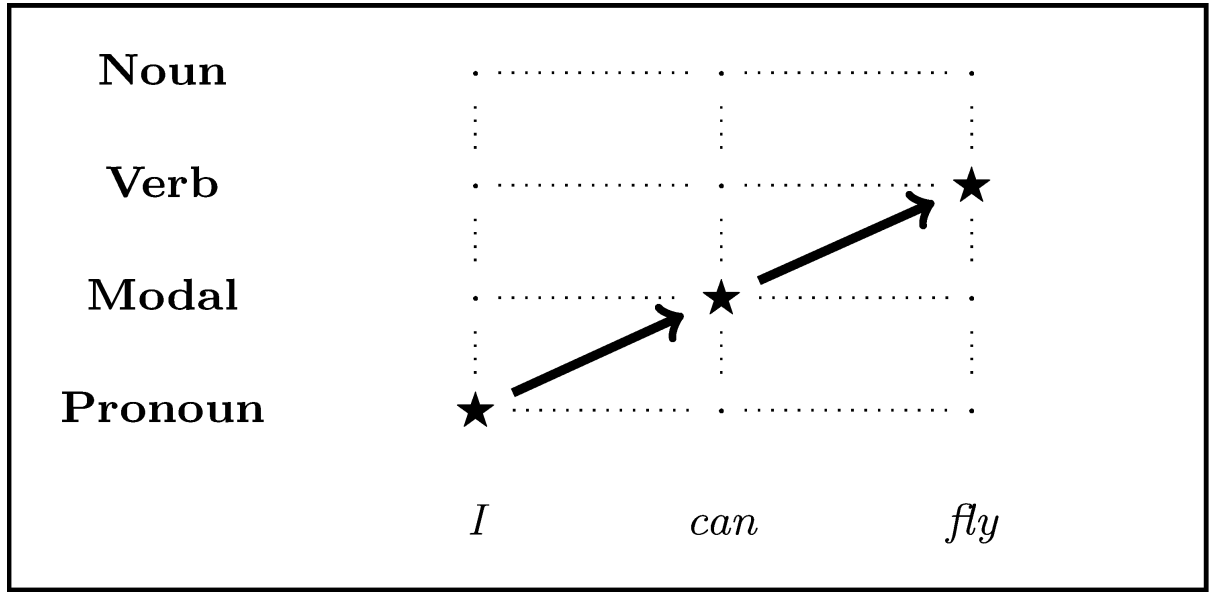
Expert policy: Return the next tag from the gold standard

Gold standard in search space



- Three actions to complete the output
- Expert policy replicates the gold standard

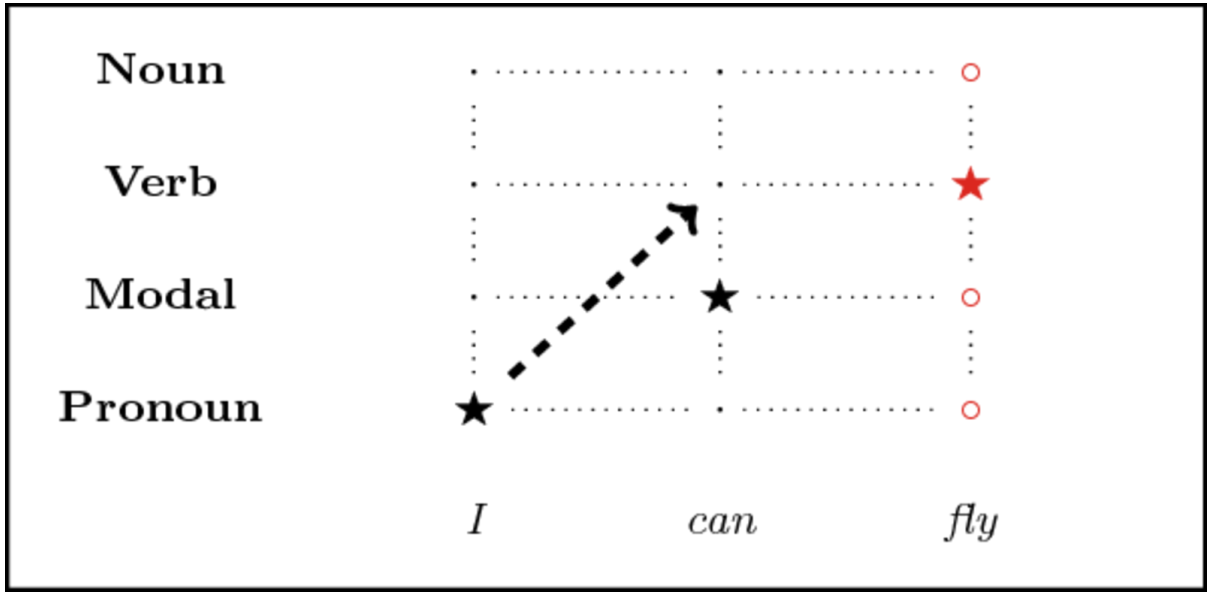
Training a classifier with structure features



label	features
Pronoun	token=I, ..., prev=**NULL**
Modal	token=can, ..., prev=**Pronoun**
Verb	token=fly, ..., prev=**Modal**

With logistic regression and k previous tags: training a kth-order Maximum Entropy Markov Model (McCallum et al., 2000 (http://people.csail.mit.edu/mcollins/6864/slides/memm.pdf))

Exposure bias



We had seen:

label	features
Verb	token=fly,..., prev=**Modal**

but not:

label	features
Verb	token=fly,..., prev=**Verb**

Addressing exposure

Allow the classifier to guide the learning

(https://www.pinterest.com/explore/affordable-driving-school/)

- 1st iteration: **roll-in** through the data with the expert
- 2nd onwards: mix expert and classifier to expose the classifier to its own actions

DAgger algorithm

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$, *expert probability* $\beta = 1$
while termination condition not reached **do**
  set *rollin policy* $\pi^{in} = \beta H + (1 - \beta)\pi^\star$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      ask expert for best action $\alpha^\star = \pi^\star(\mathbf{x}, S_{t-1})$
      extract $features = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (features, \alpha^\star)$
  learn H from $\mathcal{E}$
  decrease β

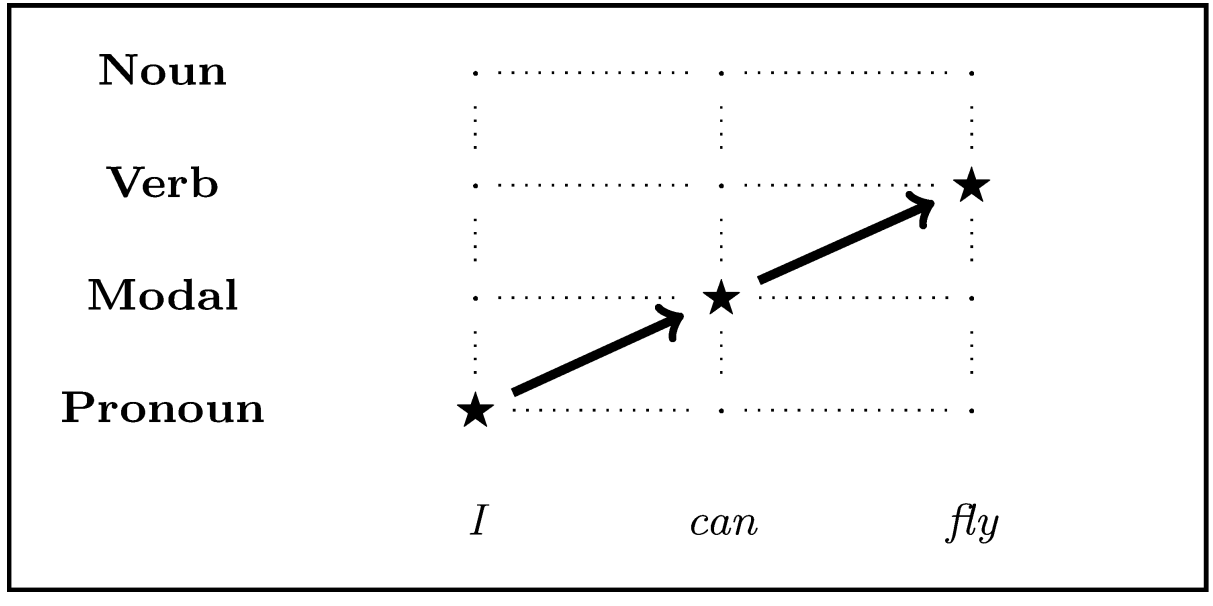
DAgger algorithm

Proposed by Ross et al. (2011) (http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats11-NoRegret.pdf) motivated by robotics

- first iteration is standard classification training
- task loss and gold standard are implicitly considered via the expert
- DAgger: the Datasets in each iteration are Aggregated

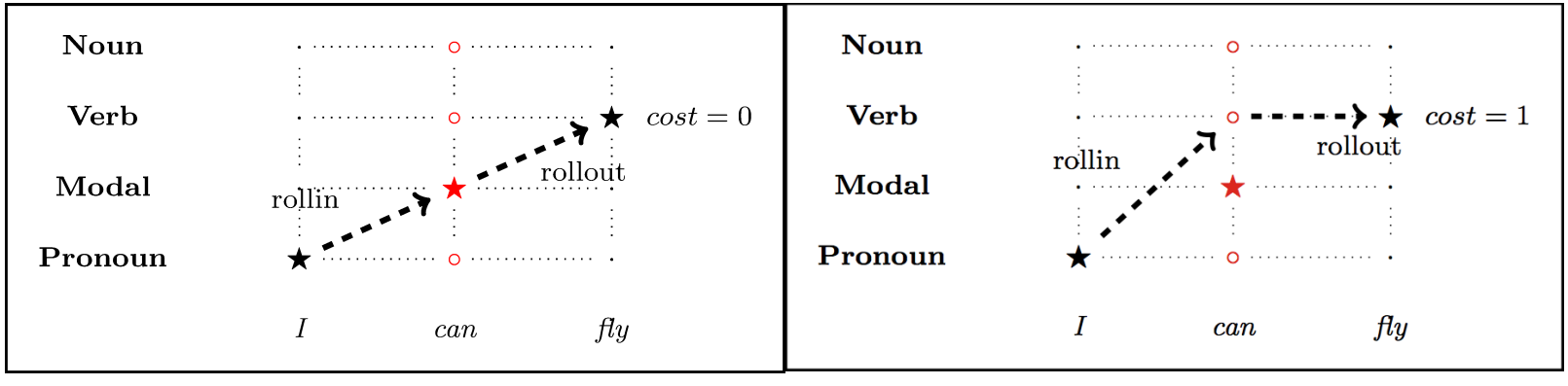
So far we looked at how to overcome previous errors. What about anticipating (and avoiding) the future ones?

Training labels as costs



Pronoun	Modal	Verb	Noun	features
0	1	1	1	token=I, prev=**NULL**...
1	0	1	1	token=can, prev=**Pronoun**...
1	1	0	1	token=fly, prev=**Modal**...

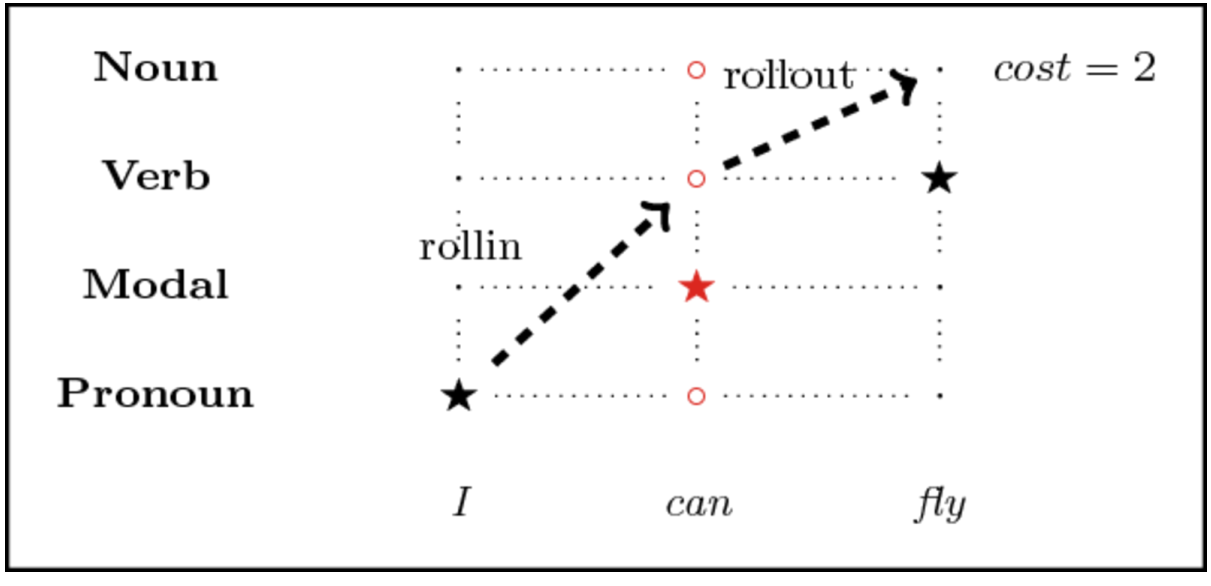
Cost break down



- **roll-in** to a point in the sentence
- try each possible label and **rollout** till the end
- evaluate the complete output with the task loss
- If **roll-out** with expert only, correct action has 0 cost, incorrect 1.

Mixed roll-outs

Rolling out with the classifier allows us to see future mistak es



Pronoun	Modal	Verb	Noun	features
1	0	2	1	token=can, prev=**Pronoun**...

DAgger with roll-outs

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$, *expert probability* $\beta = 1$
while termination condition not reached **do**
  set *rollin/out policy* $\pi^{in/out} = \beta H + (1 - \beta)\pi^\star$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in/out}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      for $\alpha \in \mathcal{A}$ **do**
        rollout $S_{final} = \pi^{in/out}(S_{t-1}, \alpha, \mathbf{x})$
        cost $c_\alpha = L(S_{final}, \mathbf{y})$
      extract $features = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (features, \mathbf{c})$
  learn H from $\mathcal{E}$
  decrease β

Roll-outs

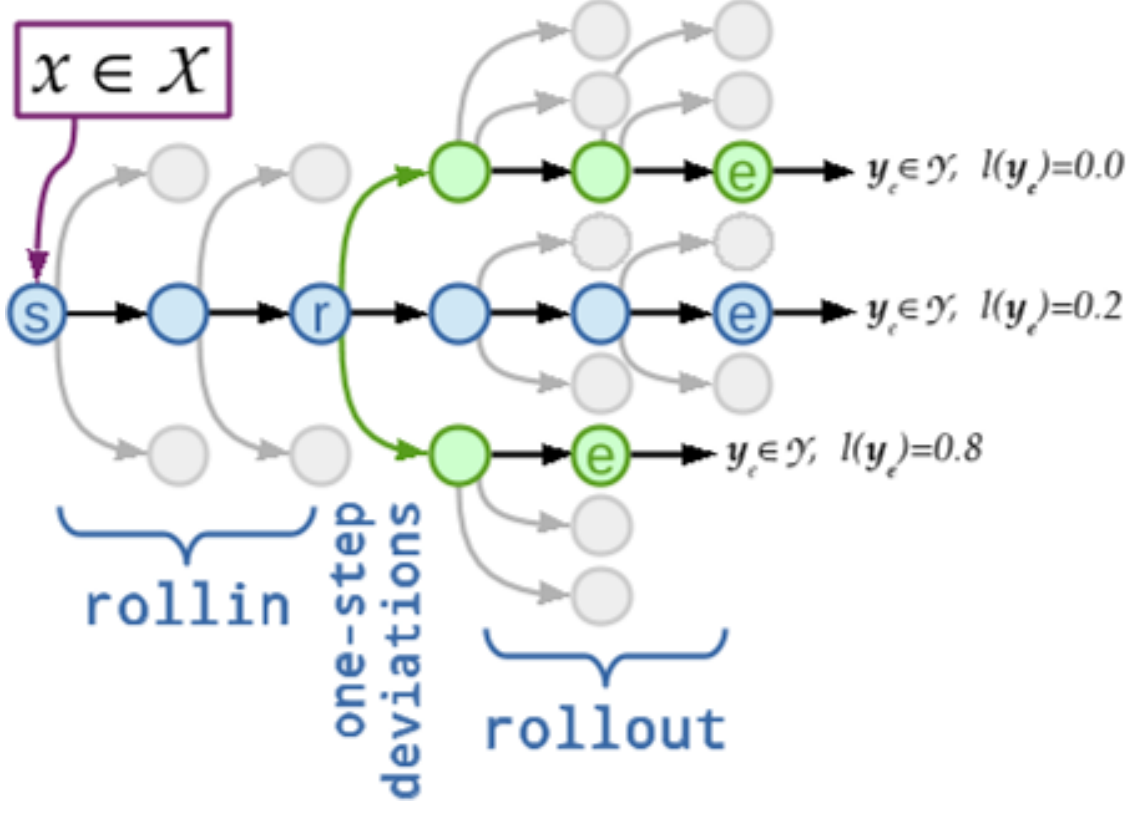
- can learn with non-decomposable losses
- can learn with sub-optimal experts
- expensive when there are many actions and long sequences to complete outputs

Some history:

- first proposed in SEARN (Daumé III et al., 2009 (http://hunch.net/~jl/projects/reductions/searn/searn.pdf))
- used to hybridise DAgger by Vlachos and Clark (http://www.aclweb.org/anthology/Q14-1042), referred to later as V-DAgger (Goodman et al. 2016 (http://aclweb.org/anthology/P16-1001))
- also proposed as look-aheads (Tsuruoka et al. 2011 (http://www.anthology.aclweb.org/W/W11/W11-0328.pdf))

LoLS

Locally Optimal Learning to Search (Chang et al., 2015 (https://arxiv.org/pdf/1502.02206.pdf))



- rollin always with the classifier
- each rollout uses either the expert or the classifier exclusively

Generic imitation learning

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$
while termination condition not reached **do**
  set *rollin policy* $\pi^{in} = mix(H, \pi^\star)$
  set *rollout policy* $\pi^{out} = mix(H, \pi^\star)$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      rollout to obtain costs c for all possible actions using L
      extract features $f = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (f, c)$
  learn H from $\mathcal{E}$

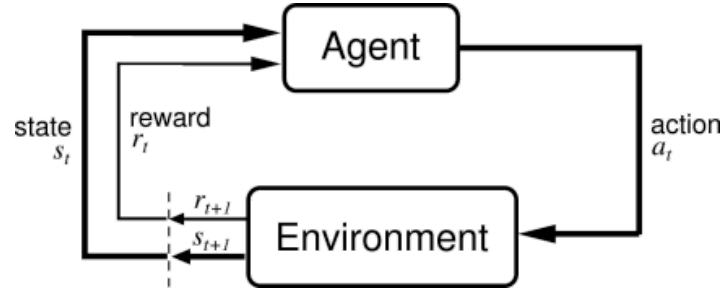
Summary so far

- basic intuition behind IL
- rollin and the DAgger algorithm
- rollouts and V-DAgger and LoLS
- generic imitation learning recipe

Interpretation and connections

Reinforcement learning

- states defined via features
- the agent is a classifier
- rewards?



(https://webdocs.cs.ualberta.ca/~sutton/book/ebook/node28.html)

Inverse reinforcement learning

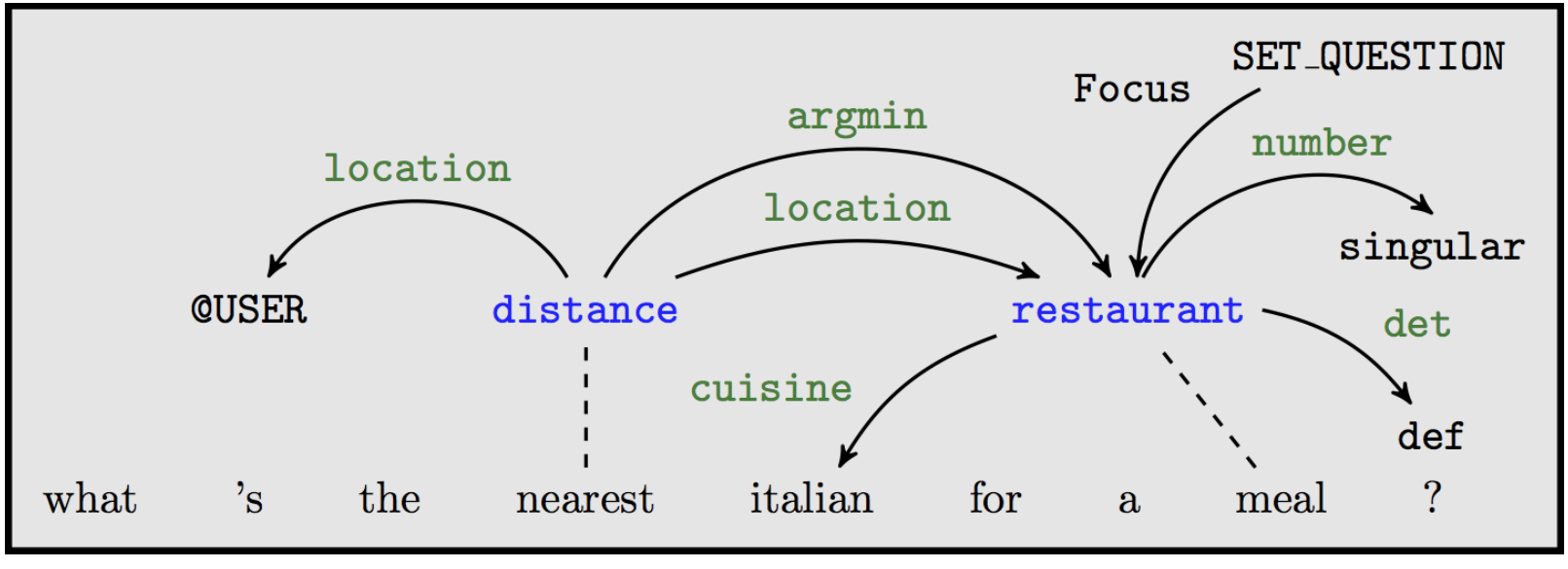
- we have the expert policy (inferred from the gold standard in the training data)
- we infer the per-action reward function (rollin/ out)

Replacing the expert policy in LoLS with a random (sub-optimal) one is RL (Chang et al., 2015 (https://arxiv.org/pdf/1502.02206.pdf))

Semi/Unsupervised learning

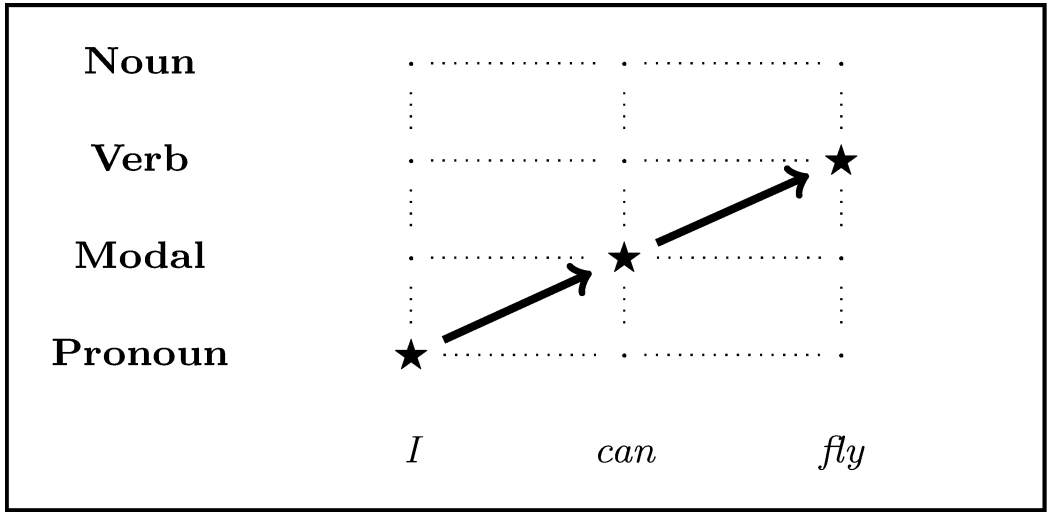
Learning with non-decompos able loss functions means

- no need to know the correct actions
- learn to predict them in order to minimize the loss



UNSEARN (Daumé III, 2009 (http://www.umiacs.umd.edu/~hal/docs/daume09unsearn.pdf)): Predict the structured output so that you can predict the input from it (auto-encoder!)

Negative data sampling



- Expert action sequence → positive example
- All other action sequences → negative examples
- Using all negative examples inefficient

Imitation learning: generate negative samples around the expert

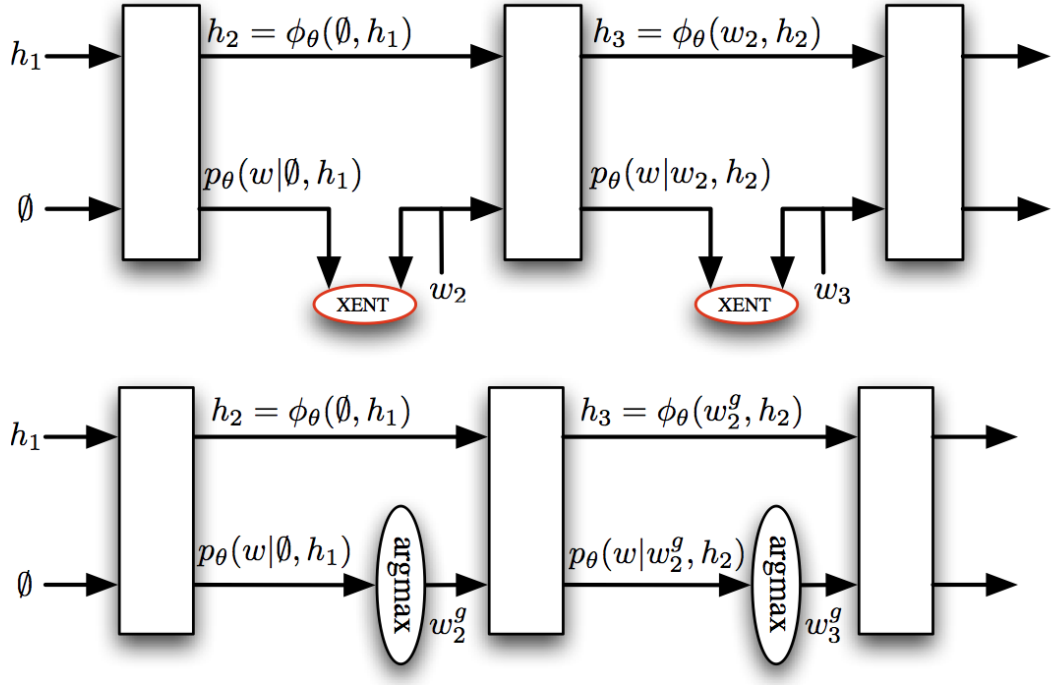
A form of **Adversarial training** (Ho and Ermon, 2016 (https://arxiv.org/abs/1606.03476))

Coaching

If the optimal action is difficult
to predict, the coach teaches
a good one that is easier
(He et al., 2012 (https://papers.nips.cc/paper/4545-imitation-learning-by-coaching.pdf))

(https://commons.wikimedia.org/wiki/File:US_Navy_091206-N-2013O-023_Sam_Givens,_a_player_for_the_Harlem_Ambassadors_basketball_team,_demonstrate

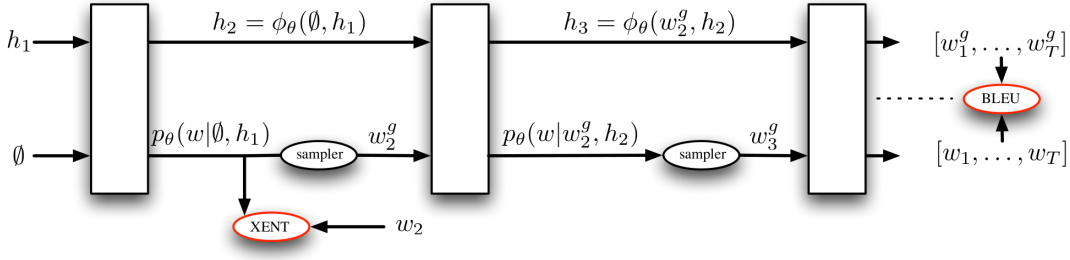
What about Recurrent Neural Networks?



They face similar problems:

- trained at the word rather than sentence level
- assume previous predictions are correct

Imitation learning and RNNs



- DAgger mixed rollins, similar to scheduled sampling (Bengio et al., 2015 (http://arxiv.org/abs/1506.03099))
- no rollouts, learn a regressor to estimate action costs
- end-to-end back propagation through the sequence
- MIXER (Ranzato et al., 2016 (https://arxiv.org/abs/1511.06732)): Mix REINFORCE-ment learning with imitation: we have the expert policy!

Summary so far

- basic concepts
  - loss function decomposability
  - expert policy
- imitation learning
  - rollin/outs
  - DAgger algorithm
  - DAgger with rollouts and LoLS
- connections and interpretations

After the break

- Applications:
  - dependency parsing
  - natural language generation
  - semantic parsing
- Practical advice
  - making things faster
  - debugging

Break!

First part recap

Imitation Learning

- Meta-learning framework, over an existing classifier(s).
- In practice, generates more (and better) training data, to improve the existing classifier(s).

For applied Imitation Learning we need to define:

- Transition system
- Task loss
- Expert policy

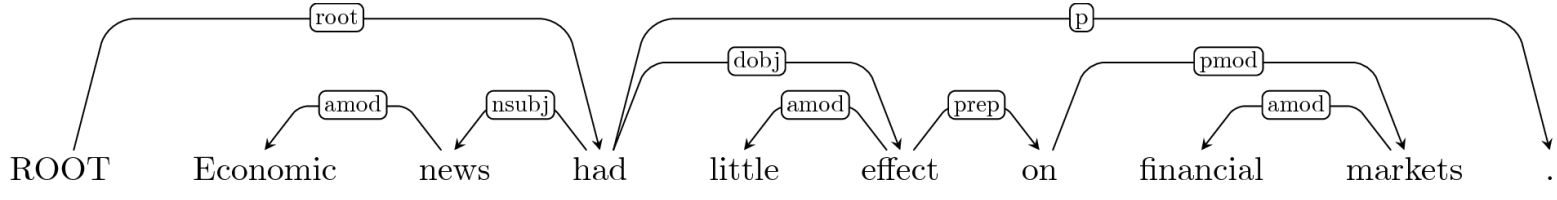
Part 2: NLP Applications and practical advice

- Applications:
  - Dependency parsing
  - Semantic parsing
  - Natural language generation
- Practical advice
  - Expert policy definition
  - Accelerating cost estimation
  - Trouble-shooting

Applying Imitation Learning on Dependenc y Parsing

Dependency parsing

(Goldberg and Nivre 2012 (http://www.aclweb.org/anthology/C12-1059), Goldberg and Nivre 2013 (https://www.aclweb.org/anthology/Q/Q13/Q13-1033.pdf))



To represent the syntax of a sentence as directed labeled edges between words.

- Where labels represent dependencies between words.

Error propagation

Due to greedy decoding, where the parser builds the parse while maintaining only the best hypothesis at each step.

- The first error encountered will confuse the classifier , since it moves the sequence to space not explored by the gold sequence of actions.
- More errors will likely follow, as the transition increasingly moves into more foreign states.

How can Imitation Learning help with that?

Imitation Learning addresses error propagation, by considering the interaction among the transition being considered and transitions to be predicted later in the sentence.

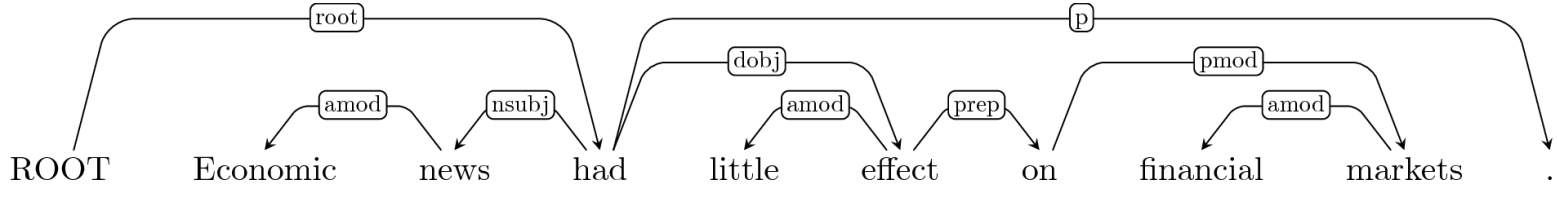
- Explores the search space, but avoids enumerating all possible outputs.
- Also learns how to recover from errors.

Transition system?

We can assume any transition-based system (Arc-Eager, Arc-Standard, Easy-First, etc.).

- Each action, transforms the current state/graph until a terminal state/graph is reached.
- In essence, which arc and label should we add ne xt?

Transition-based dependency parsing in action!



Task loss?

Hamming loss: the number of incorrectly predicted labeled or unlabeled dependency arcs.

- Directly related to the attachment score metrics used to e valuate dependency parsers.

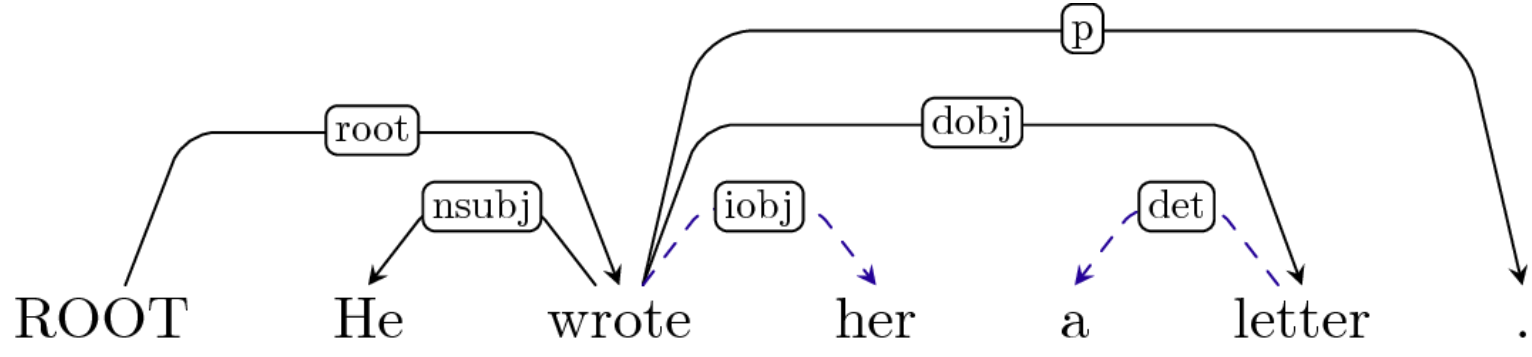
Expert policy?

A single static canonical sequence of actions from the initial to the terminal state.

- Derived from the reference graph.

Single static policies worked well for the Part-of-Speech tagging task.

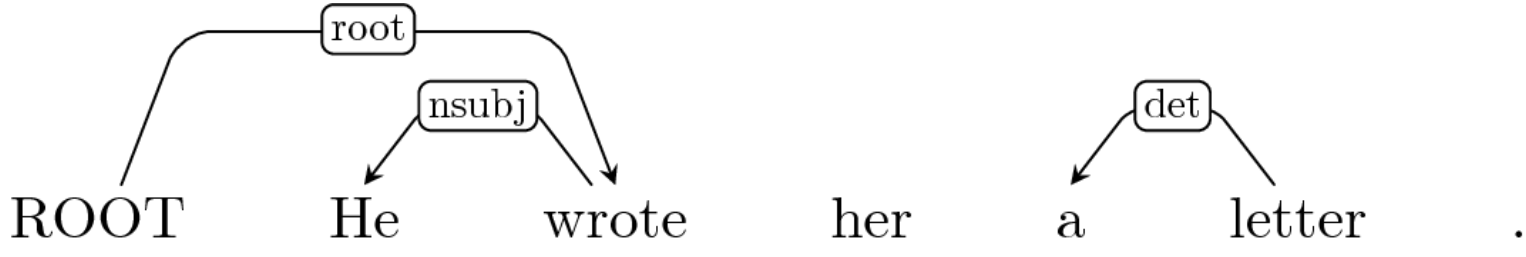
But what if there are multiple correct transitions?



A static policy could arbitarilly chose a tr ansition (e.g. prioritize shifts over other actions).

- But this indirectly labels the alternativ e transition as false!

And what if a mistake happens?

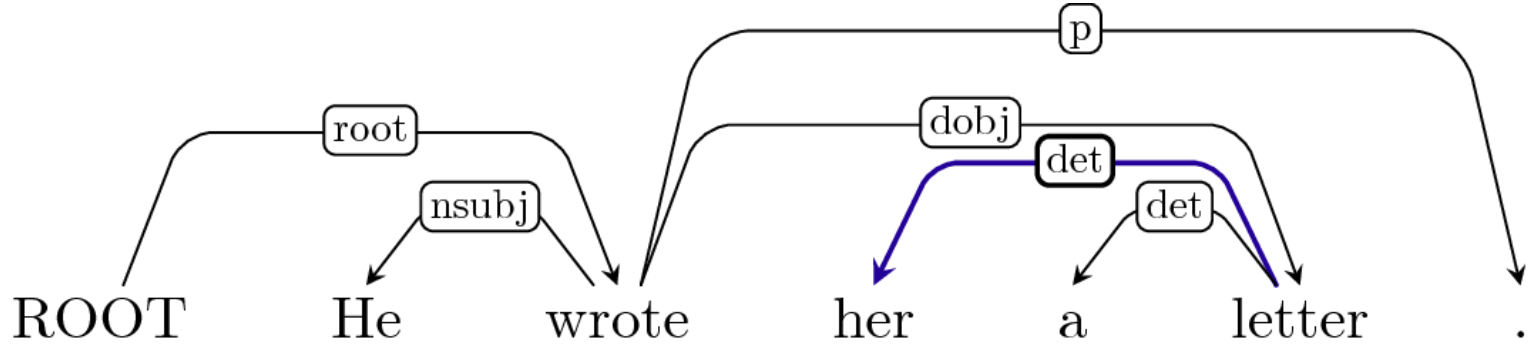


A static policy is not well defined in states that are not part of the gold tr ansition.

Dynamic policy

Non-deterministic and complete policy

- Allows ambiguous transitions.
- Defined for all states.
- Recovers from errors.

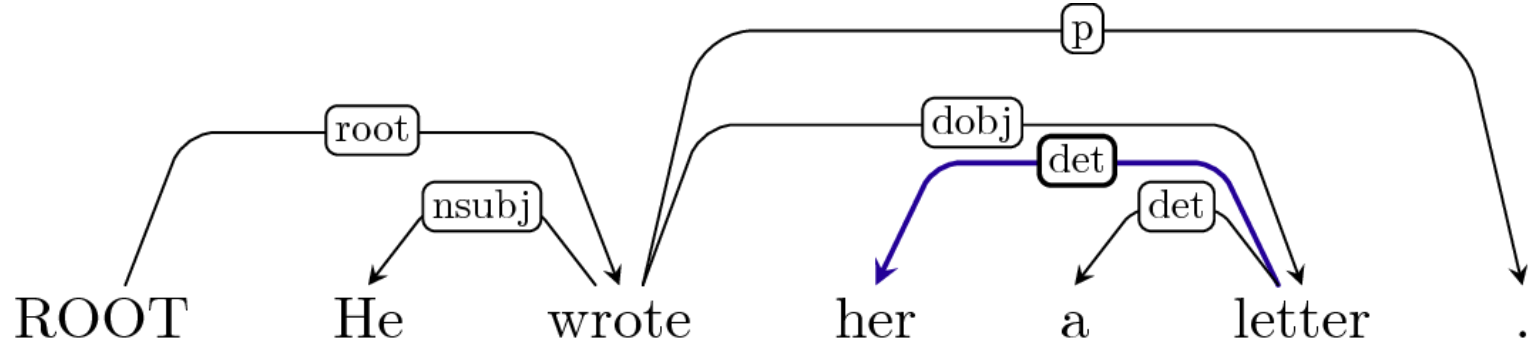


What is the expert policy then?

Given a particular state, where an error may or may not have already occured:

- We need to determine the best reachable terminal state
  - Quite possibly not an optimal terminal state, if we have made an error before.
  - "Best" according to some loss function in relation to the gold terminal state.
- Return the set of transition actions that lead to that state.

Expert policy in action!



Cost of 1, if we use the labeled attachment score as a loss function.

Is that DAgger?

Goldberg and Nivre (2013) (https://www.aclweb.org/anthology/Q/Q13/Q13-1033.pdf) proposed a system that employed dynamic expert policies for dependency parsing.

- As well as an algorithm to learn parameters by exploration.
- Very similar to DAgger.
  - Roll-in is a mix of the learned and expert policies, at the time-step level.
  - There may be multiple correct actions at each time-step.

Results

system / language	hungarian	chinese	greek	czech	basque	catalan	english	turkish	arabic	italian
					UAS					
eager:static	76.42	85.01	79.53	78.70	75.14	91.30	86.10	77.38	81.59	84.40
eager:dynamic	77.48	85.89	80.98	80.25	75.97	92.02	88.69	77.39	83.62	84.30
hybrid:static	76.39	84.96	79.40	79.71	73.18	91.30	86.43	75.91	83.43	83.43
hybrid:dynamic	77.54	85.10	80.49	80.07	73.70	91.06	87.62	76.90	84.04	83.83
easyfirst:static	81.27	87.01	81.28	82.00	75.01	92.50	88.57	78.92	82.73	85.31
easyfirst:dynamic	81.52	87.48	82.25	82.39	75.87	92.85	89.41	79.29	83.70	85.11
					LAS					
eager:static	66.72	81.24	72.44	71.08	65.34	86.02	84.93	66.59	72.10	80.17
eager:dynamic	68.41	82.23	73.81	72.99	66.63	86.93	87.69	67.05	73.92	80.43
hybrid:static	66.54	80.17	70.99	71.88	62.84	85.57	84.96	64.80	73.16	78.78
hybrid:dynamic	68.05	80.59	72.07	72.15	63.52	85.47	86.28	66.12	74.10	79.25

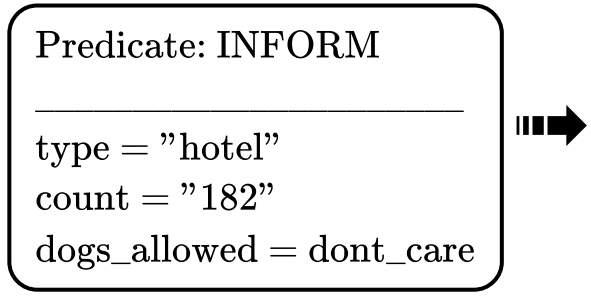
Applying Imitation Learning on Natural Language Generation

Natural Language Generation (concept-to-text)

(Lampouras and Vlachos 2016 (https://aclweb.org/anthology/C/C16/C16-1105.pdf))

...is the natural language processing task of generating text from a non-linguistic form...

- e.g. a meaning representation, database records.

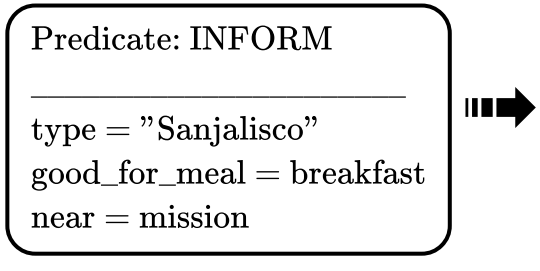


There are _182_ _hotels_ if you _do not care_ whether dogs are allowed.

NLG Motivation

Statistical methods (mostly) rely on human-annotated data for tr aining.

- Especially on alignments between the meaning representation and reference texts.
- Time-consuming and costly to construct.



__Sanjalisco__ is good for __breakfast__ and is near the __mission__ district.

How can Imitation Learning help with that?

Imitation Learning can be used to learn from unaligned data.

- Why unaligned data? To limit the cost of dataset construction!
- Why Imitation Learning? It can learn from non-decomposable loss functions, and suboptimal training data!

Transition system?

NLG is a complex task due to large output space.

- The set of possible words limited to those observ ed from the references of the training data.

We formulate NLG as a sequence A of two types of actions:

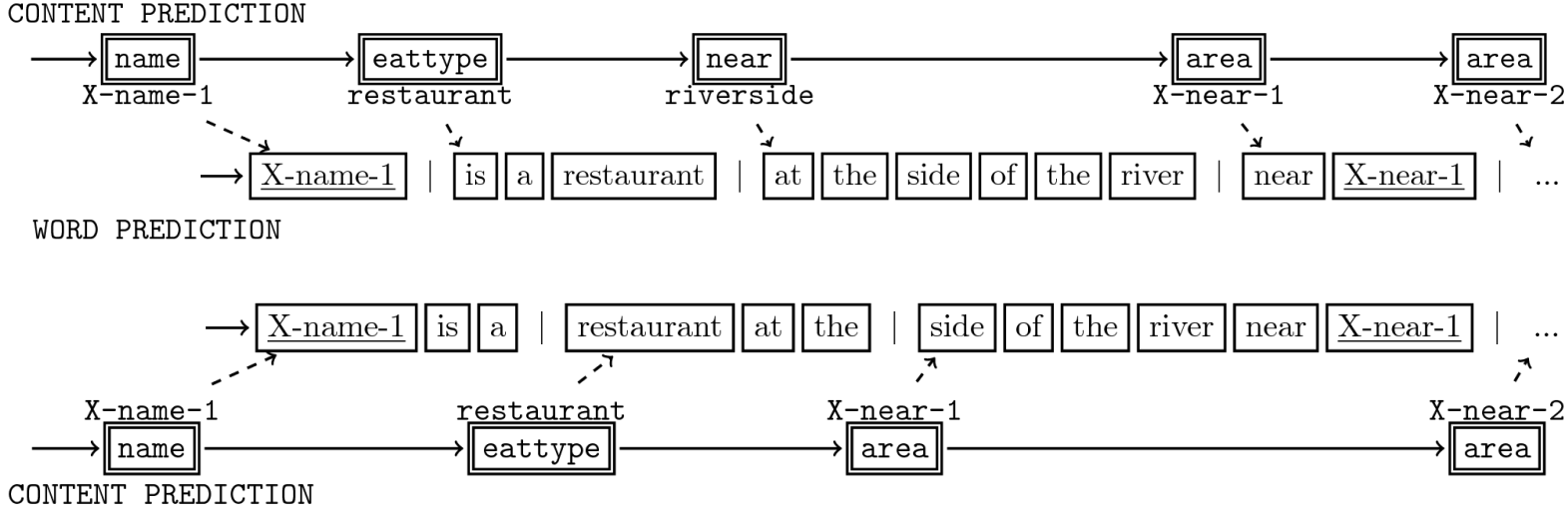
- content prediction actions a_c, and
- word prediction actions a_w.

NLG formulation

Input: meaning representation MR with set of attributes C, attribute dictionaries D_c, $\forall c \in C$
Output: action sequence A

do
  predict attribute $c \in C \cup END_{attr}$
  append a_c to A_c
  remove c from C
while $ac \neq END_{attr}$
for a_c *in* A_c **do**
  do
    predict word $w \in D_c \cup END_{word}$
    append a_w to A_w
  while $a_w \neq END_{word}$
$A = (A_c, A_w)$

NLG transition in action!



Task loss?

We can use various loss functions (e.g. BLEU, ROUGE).

- Content actions are ignored by the loss function, but they are indirectly evaluated on their impact on the word predictions that follow them.
- The loss function also penalizes undesirable behaviour, e.g. repeating the same word, predicting attributes not in the MR.

Expert policy?

The expert policy π_{ref} is based on:

- the NL references of the MR,
- and the alignments..?

Alignments

Training these models (independently or jointly) would be possible if we extracted data from manually aligned training references.

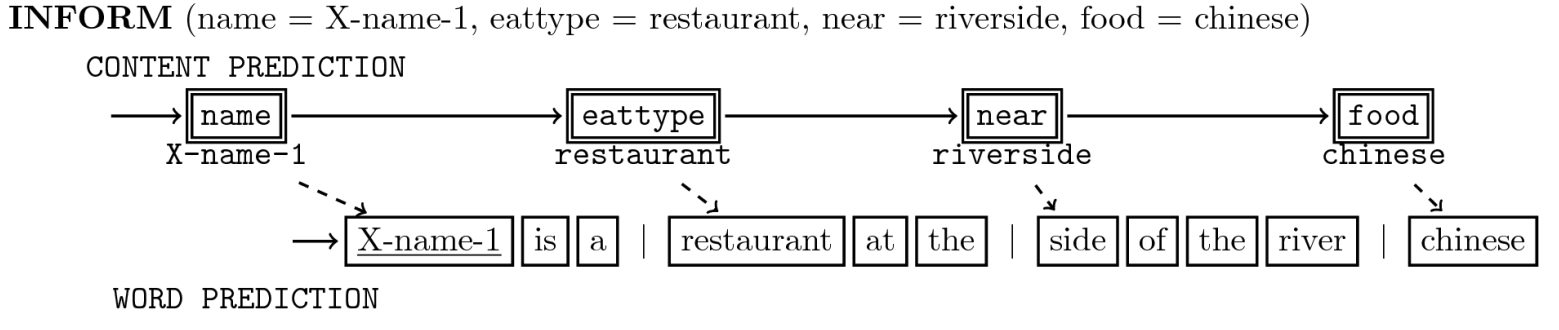
- However, we do not assume access to such information!

If no alignments are available, they could be automatically calculated (Liang et al. 2009 (http://www.aclweb.org/anthology/P09-1011)).

- But Liang et al.'s model was trained on the datasets considered, and does not generalize well.
- We will assume no access to that either.

Using naive alignments

References:
X-name-1 is a	restaurant at the	side of the river.
X-name-1 is a	restaurant at the	riverside.
X-name-1 is a	restaurant by the	river that serves	Chinese.
X-name-1 is a	riverside	restaurant that serves	Chinese.
For a Chinese	restaurant,	go to X-name-1 near the	riverside.



Suboptimal expert policy

Since our gold standard is naiv ely constucted, the resulting expert policy is suboptimal.

Other potential causes of suboptimal e xperts are computational restraints.

- For large action sequences we ma y need to limit our estimations on a subsequence.

Locally Optimal Learning to Search

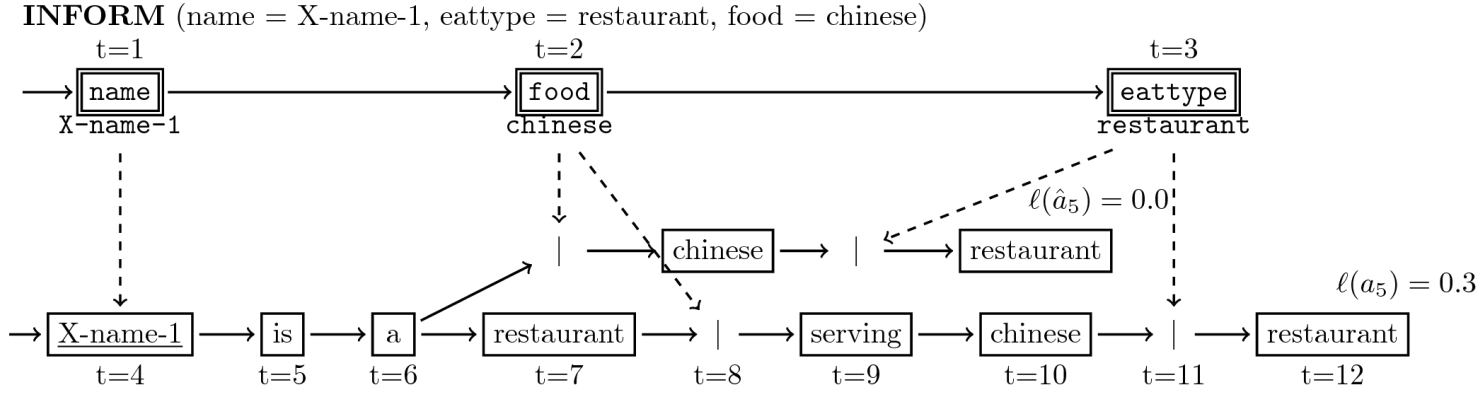
LOLS can learn from suboptimal π_{ref}

- Because it potentially performs roll-outs with π_i.

LOLS can learn from non-decomposable loss functions (e.g. BLEU, ROUGE).

- Because it only needs to evaluate complete output predictions, not individual actions.
- For NLG, this means we do not require explicit supervision on how each action is aligned, or which predictor should generate each word; we just need a way to evaluate how good the complete final sentence is.

LOLS in action!



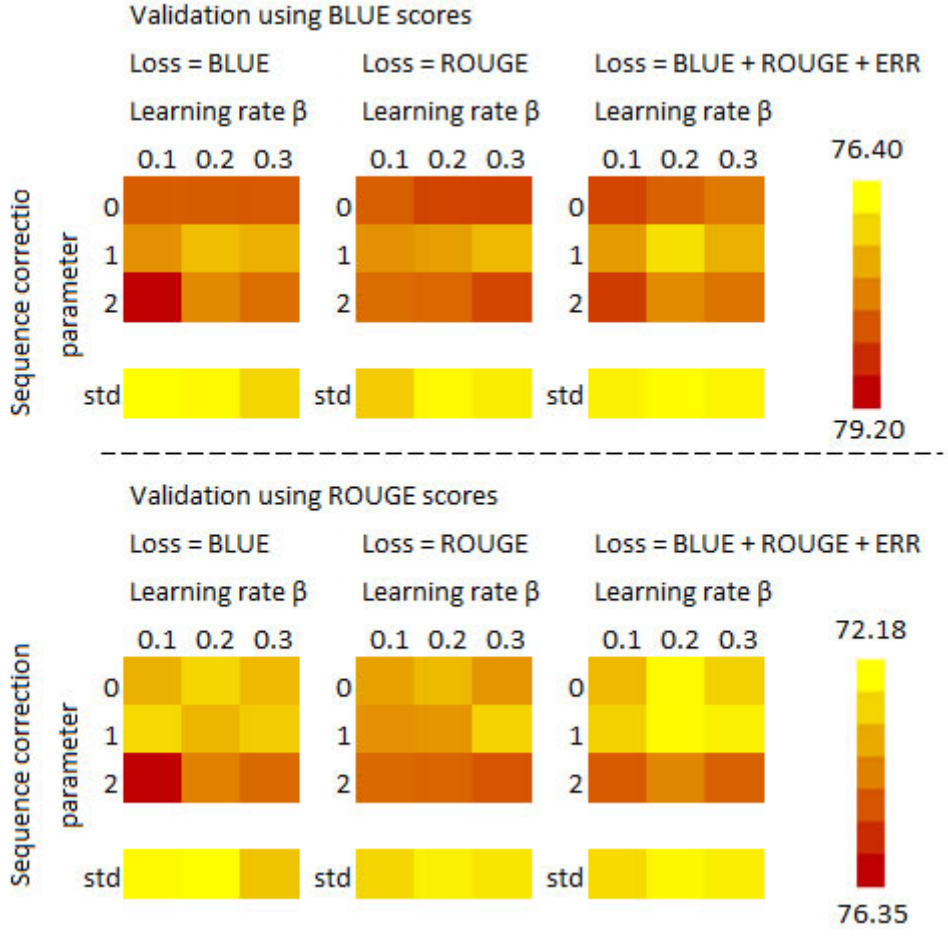
Sequence correction

Imitation Learning on NLG can generate very noisy training instances. To address this, we apply sequence correction before mo ving to the next timestep:

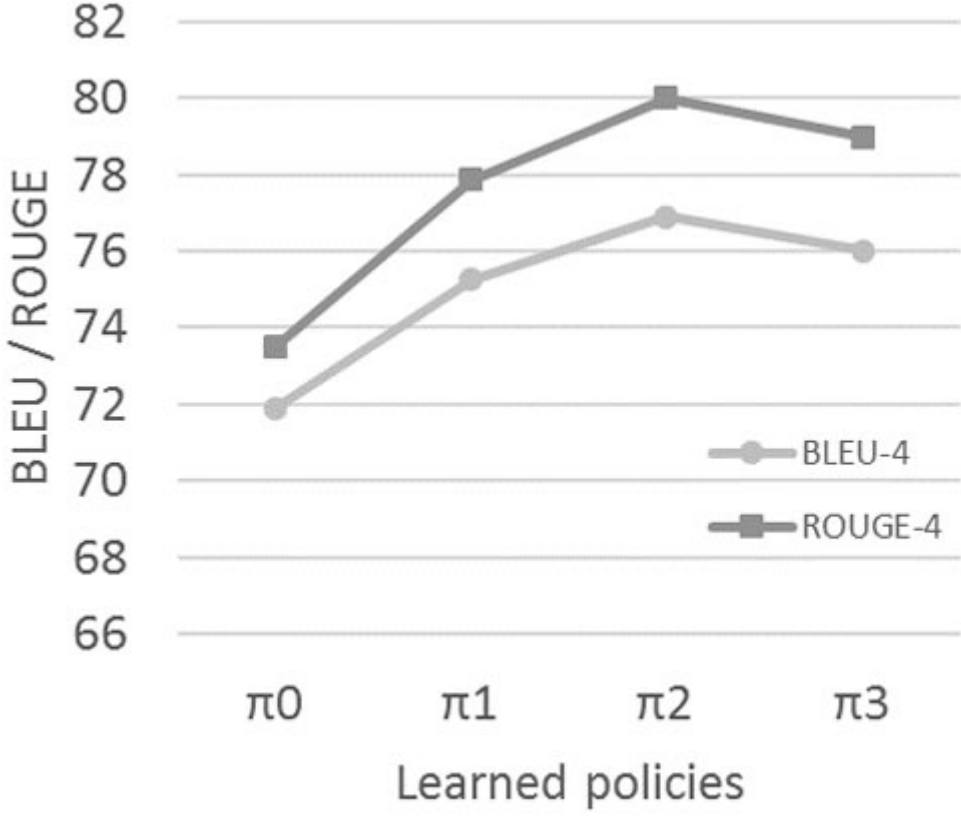
- We correct all the already examined actions using π_{ref}.
- And re-predict the rest of the sequence using π_i.

If suboptimal actions are encountered further in the new sequence, sequence correction may again be performed. Before SC, we may allow the examination of at most E actions after the first suboptimal one; to allow the predictors to learn how to recover from the mistake.

Sequence Correction results



Results per LOLS epoch



Automatic evaluation for NLG

	SF Restaurant			SF Hotel		
	BLEU	**ROUGE**	**ERR(%)**	**BLEU**	**ROUGE**	**ERR(%)**
LSTM	52.97	43.52	6.29	66.37	56.19	3.99
LOLS	49.44	38.52	0.58	68.65	68.37	0.52

Human evaluation for NLG

	SF Restaurant		SF Hotel	
	Fluency	**Informativeness**	**Fluency**	**Informativeness**
LSTM	4.49	5.29	4.41	5.36
LOLS	4.23	5.36	4.68	5.19

We performed Analysis of Variance (ANOVA) and post-hoc Tukey tests (a = 0.05); there is no statistically significant difference.

Applying Imitation Learning on Semantic P arsing

Semantic parsing

(Goodman et al. 2016 (http://aclweb.org/anthology/P16-1001))

Semantic parsers map natural language to meaning representations

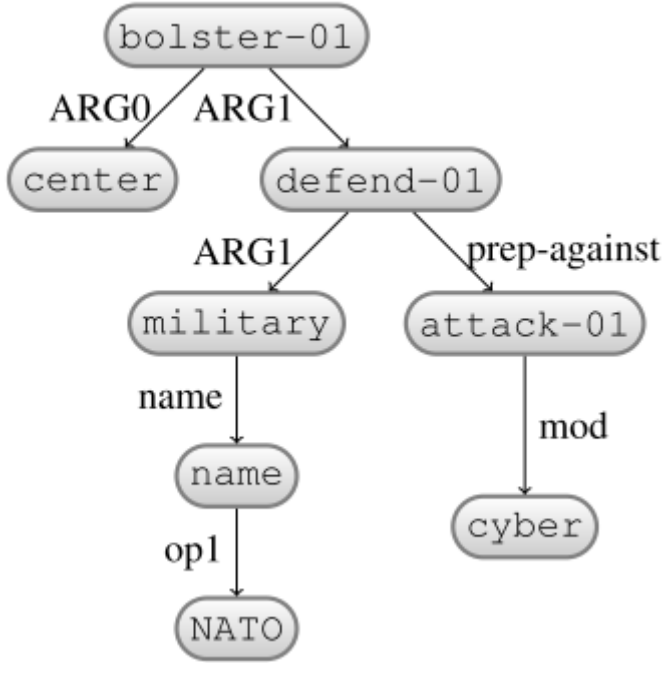
- Need to abstract over syntactic phenomena, resolve anaphora, eliminate ambiguousness in word senses
- Essentially the inverted task of natural language generation

Also known as:

- natural language understanding, natural language database interfaces, semantic role labeling, question answering on databases

Abstract meaning representation

(Banarescu et al. 2013 (http://www.aclweb.org/anthology/W13-2322))



A meaning representation formalism that utilizes a gr aph to represent relationships between concepts.

- Structure similar to dependency parses.
- But abstracts away from function words, and inflection details of words.
- Due to its structure, tr ansition-based approaches are common.

How can Imitation Learning help with that?

Similarly to dependency parsing, greedy encoding suffers from error propagation.

Imitation Learning addresses error propagation, by considering the interaction among the transition being considered and transitions to be predicted later in the sentence.

- Explores the search space, but avoids enumerating all possible outputs.
- Also learns how to recover from errors.

Transition system?

We consider a dependency graph (tree) as input.

- Dependency graphs are derived from the sentences.
- There is a lot more training data avalaible for dependecy parsing, than exists for AMR parsing.

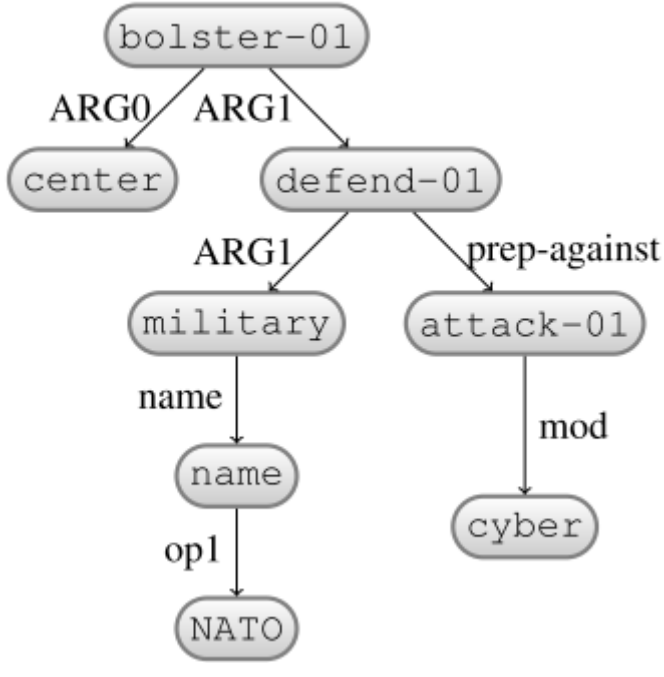
Transition actions transform the dependency graph into an AMR graph.

- In intermediate stages, some nodes are labeled with words from the sentence, and others with AMR concepts.

Actions

Action Name	Param.	Pre-conditions	Outcome of action
NextEdge	l_r	β non-empty	Set label of edge (σ_0, β_0) to l_r. Pop β_0.
NextNode	l_c	β empty	Set concept of node σ_0 to l_c. Pop σ_0, and initialise β.
Swap		β non-empty	Make β_0 parent of σ_0 (reverse edge) and its sub-graph. Pop β_0 and insert β_0 as σ_1.
ReplaceHead		β non-empty	Pop σ_0 and delete it from the graph. Parents of σ_0 become parents of β_0. Other children of σ_0 become children of β_0. Insert β_0 at the head of σ and re-initialise β.
Reattach	κ	β non-empty	Pop β_0 and delete edge (σ_0, β_0). Attach β_0 as a child of κ. If κ has already been popped from σ then re-insert it as σ_1.
DeleteNode		β empty; leaf σ_0	Pop σ_0 and delete it from the graph.
Insert	l_c		Insert a new node δ with AMR concept l_c as the parent of σ_0, and insert δ into σ.
InsertBelow			Insert a new node δ with AMR concept l_c as a child of σ_0.

Transition-based AMR parsing in action!



Task loss?

Smatch (Cai and Knight, 2013 (http://amr.isi.edu/smatch-13.pdf))

- F_1-Score between predicted and gold-target AMR gr aphs.
- Computationally expensive for every rollout.

Expert policy?

Best reachable state is explored via roll-outs, with naive Smatch used as a loss function.

- Skips combinatorial mapping of nodes between predicted and target gr aphs.
- Also, to encourage short trajectories, a length penalty is applied.

V-DAgger

Variant of DAgger proposed by Vlachos and Clark (2014) (http://www.aclweb.org/anthology/Q14-1042)

- Employs roll-outs, with the same policy used for both roll-ins and roll-outs.

Imitation Learning challenges

Incredible number of possible actions at each time-step.

- In the order of 10^3 to 10^4.
- Exploring all alternative actions at each time-step can be very time-consuming.

Incredible length of the action sequences.

- In the range of 50-200 actions.
- Especially challenging when combined with the large number of possible actions.

Targeted exploration

There is no reason to explore alternative actions when:

- Expert and learned policy agree on the correct action, **and**
- no alternative action is scored highly.

The algorithm limits the exploration to the expert action and learned policy actions whose scores is within a threashold τ from the best scored one.

- In first epoch, where there is no learned policy , we randomly explore a number of actions.

Other cases of partial exploration

SCB-LOLS and AggreVaTe both use partial exploration.

- They select which time-step they apply it at random.
- They select which actions they explore at random.

Targeted exploration focuses on the actions for which the leaned policy is least certain, or disagrees with the expert.

Issues of step-level stochasticity

v-DAgger and SEARN employ step-level stochasticity during their roll-outs.

- i.e. each step during roll-out can be performed by either the learned or expert policy.
- In other words, the same training example may have very different roll-outs when reexamined.
- This results in high variance in the reward signal, and hinders effective learning.

Noise reduction

a-bound by Khardon and Wachman (2007) (http://www.jmlr.org/papers/volume8/khardon07a/khardon07a.pdf)

- Exclude a training example from subsequent training if it has been already misclassified a times during training.

Alternatively, we could use LOLS

- Rollouts are performed consistently with the same policy .
- Can hurt training times when moving from exclusive expert to exclusive learned policy, due to large length of action sequences.

Focused costing

Introduced by Vlachos and Craven (2011) (http://www.aclweb.org/anthology/W/W11/W11-0307.pdf)

- Instead of using learned policy for β% of the rollout steps,
- use it for the first β steps and reverty to the expert policy for the rest.

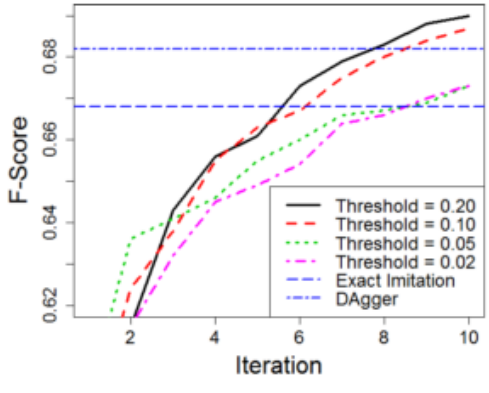
This keeps roughly the same computational cost, while focusing the effect of the explored action to the immediate actions that follow.

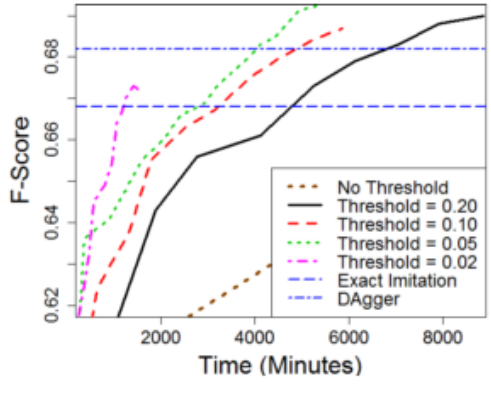
- Reduces noise, the mistakes the learned policy may make on distant actions are considered irrelevant.
- We can increase β with each epoch, to move away from the expert.

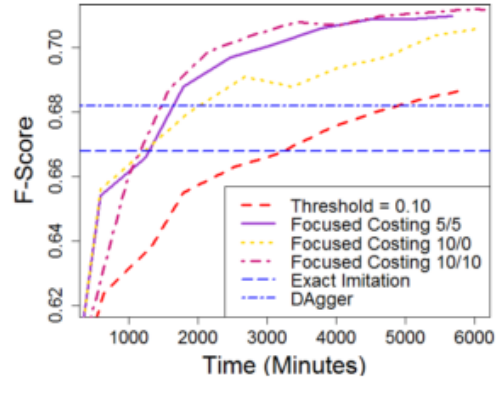
DAgger with a-bound

	Exact Imitation			Imitation Learning				
Experiment	No α	α=1	α-Gain	No α	α=1	IL Gain (α)	IL Gain (No α)	Total Gain
AROW, C=10	65.5	66.8	1.3	65.5	67.4	0.6	0.0	1.9
AROW, C=100	66.4	66.6	0.2	66.4	67.7	1.1	0.0	1.3
AROW, C=1000	66.4	67.0	0.6	66.5	68.2	1.2	0.1	1.8
PA, C=100	66.7	66.5	-0.2	67.2	68.7	2.2	0.5	2.0
Perceptron	65.5	65.3	-0.2	66.6	68.6	3.3	1.1	3.1

Targeted exploration and focused costing results







Comparison with previous work

Authors	Algorithmic Approach	R	P	F
Flanigan et al. (2014)	Concept identification with semi-markov model followed by optimisation of constrained graph that contains all of these.	0.52	0.66	0.58
Werling et al. (2015)	As Flanigan et al. (2014), with enhanced concept identification	0.59	0.66	0.62
Wang et al. (2015b)	Single stage using transition-based parsing algorithm	0.62	0.64	0.63
Pust et al. (2015)	Single stage System-Based Machine Translation	-	-	0.66
Peng et al. (2015)	Hyperedge replacement grammar	0.57	0.59	0.58
Artzi et al. (2015)	Combinatory Categorial Grammar induction	0.66	0.67	0.66
Wang et al. (2015a)	Extensions to action space and features in Wang et al. (2015b)	0.69	0.71	0.70
This work	Imitation Learning with transition-based parsing	0.68	0.73	0.70

Comparison with previous work

	Validation F-Score			Test F-Score	
Dataset	EI	D	V-D	V-D	Rao et al
proxy	0.670	0.686	0.704	0.70	0.61
dfa	0.495	0.532	0.546	0.50	0.44
bolt	0.456	0.468	0.524	0.52	0.46
xinhua	0.598	0.623	0.683	0.62	0.52
lpp	0.540	0.546	0.564	0.55	0.52

Lighning round

brief presentation of other imitation learning related work

Work using imitation learning in EACL 2017

Tackling Error Propagation t hrough Reinforcement Learning: A Case of Greedy Dependency Parsing
Minh Lê and Antske Fokkens

Summary

Discussed expert policy definitions:

- Static vs. dynamic vs. suboptimal policies.

Examined variations to exploration:

- Early termination, and targetted and partial exploration.

Showed how we can accelerate cost estimation.

- Noise reduction and focused costing.

Applied Imitation Learning on various NLP tasks.

- Transitions, loss functions, expert policies.
- And improve on the results!