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) (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.

Predicate: INFORM

type = ”hotel”
count = ”182”
dogs_allowed = dont_care

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.

Predicate: INFORM

type = "Sanjalisco"
good_for_meal = breakfast
near = mission

__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:

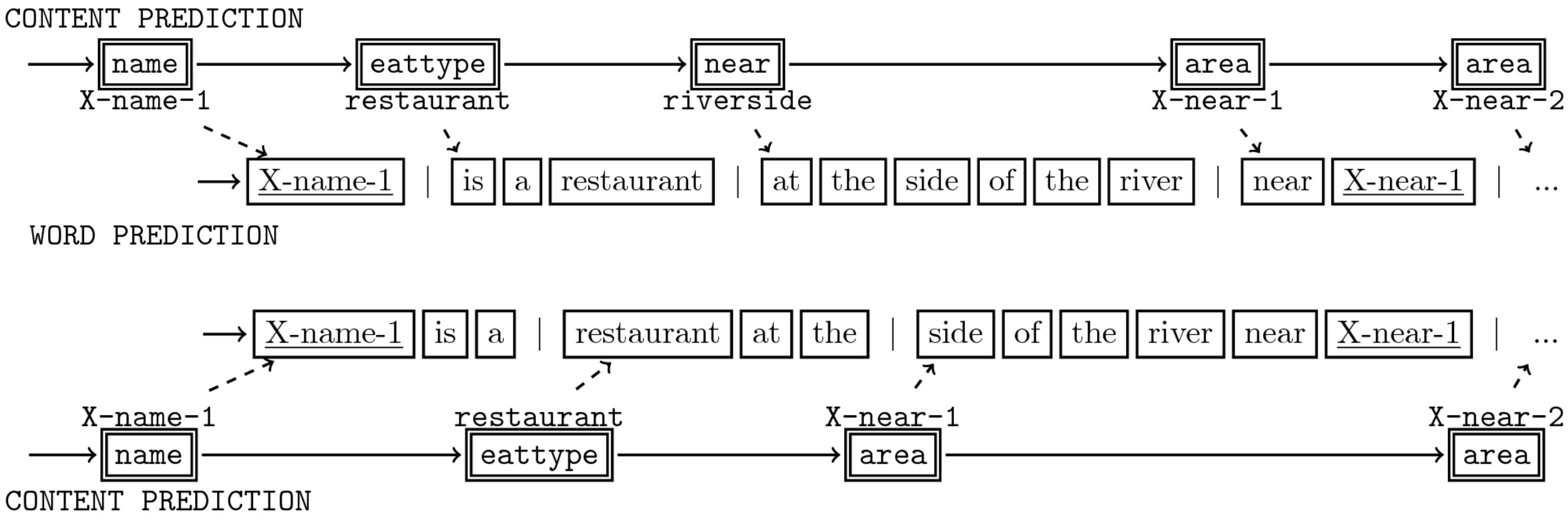
- content prediction actions a_c, and
- word prediction actions a_w.

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!

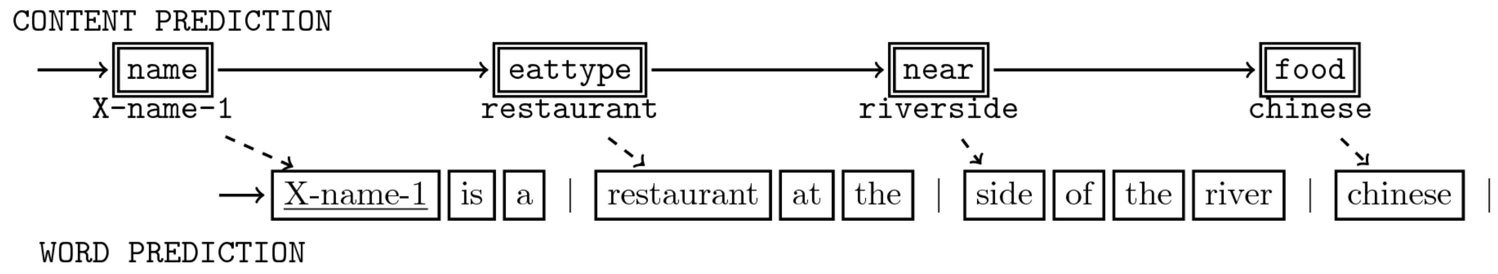
If no alignments are available, they could be automatically calculated ([Liang et al. 2009 (http://www.aclweb.org/anthology/P09-1011)](http://www.aclweb.org/anthology/P09-1011)).

- 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.

INFORM (name = X-name-1, eattype = restaurant, near = riverside, food = chinese)



Suboptimal expert policy

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

Other potential causes of suboptimal experts are computational restraints.

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

Locally Optimal Learning to Search

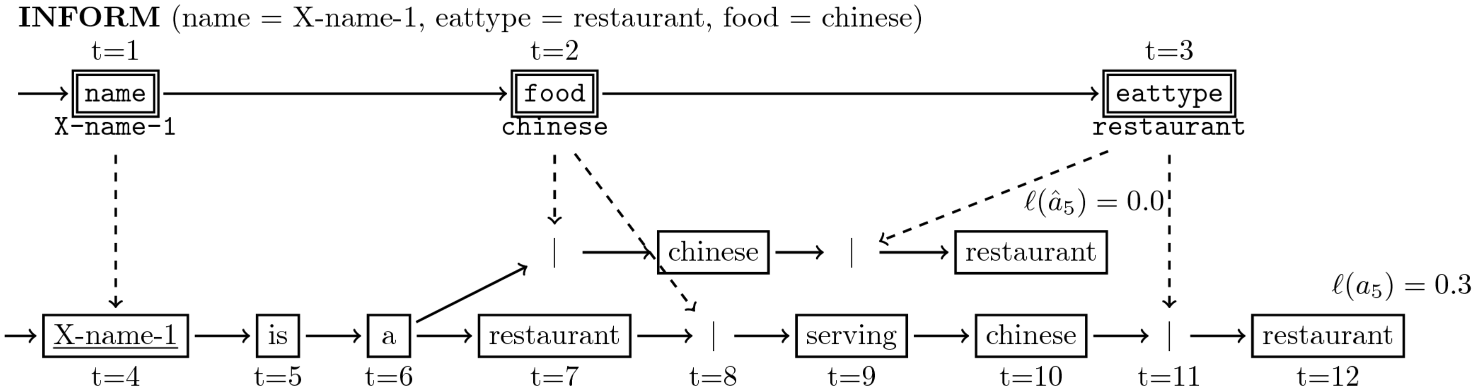
LOLS can learn from suboptimal π_{ref}

- 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!



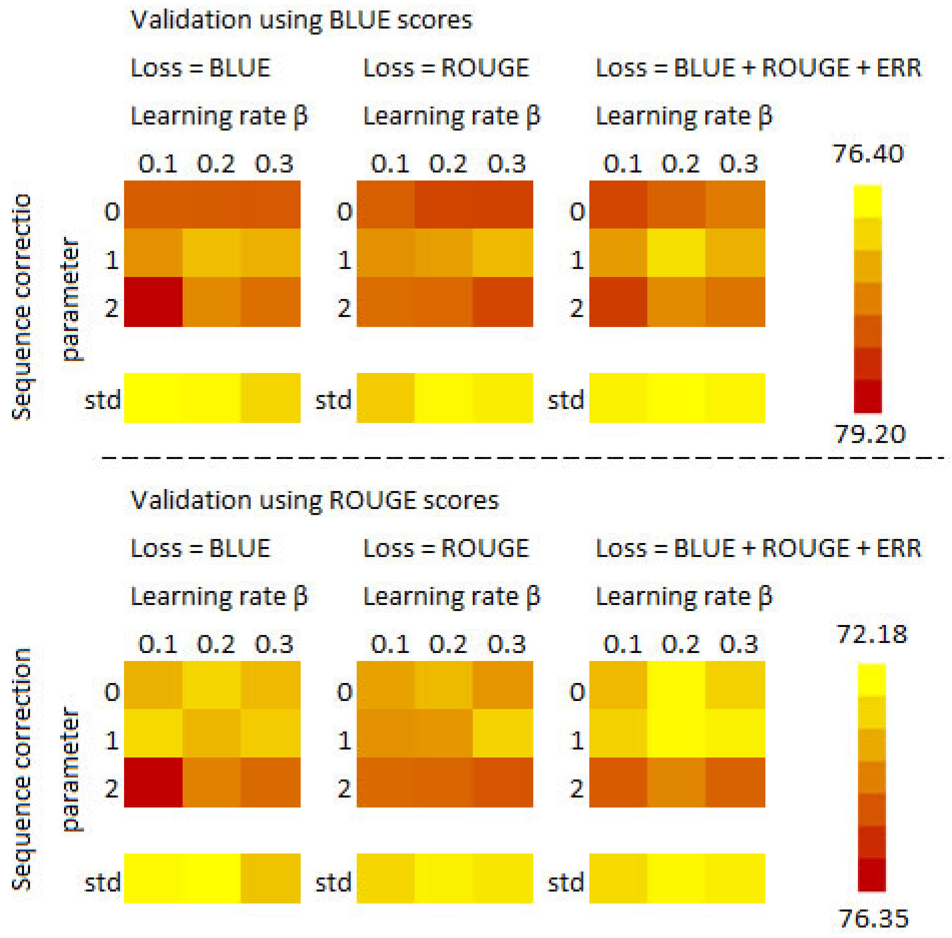
Sequence correction

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

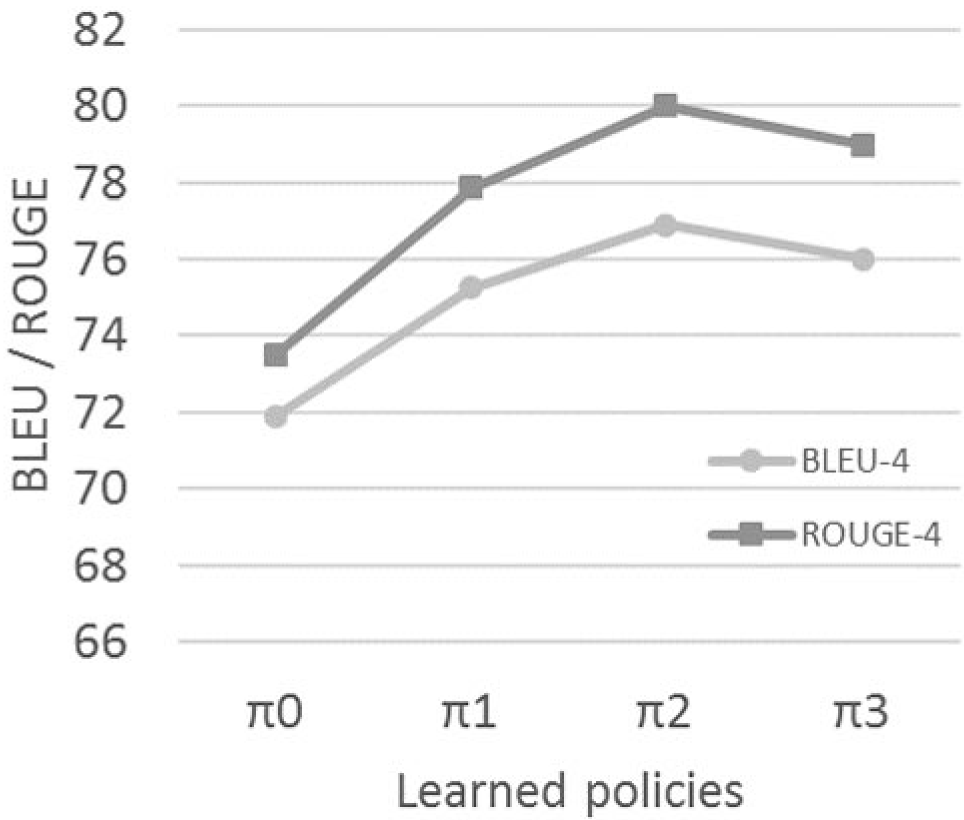
- We correct all the already examined actions using π_{ref}.
- And re-predict the rest of the sequence using π_i.

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.