Automatic Response Generation for Conversational e-Commerce Agents

A Reinforcement Learning based approach to entertainment in NLG

Snehasish Mukherjee (mukherji@stanford.edu)

Objective

A response is said to be **entertaining** when it follows the users way of speaking. Entertaining responses are desirable but many conversational e-commerce systems use hard-coded templates to generate response and hence cannot entertain.

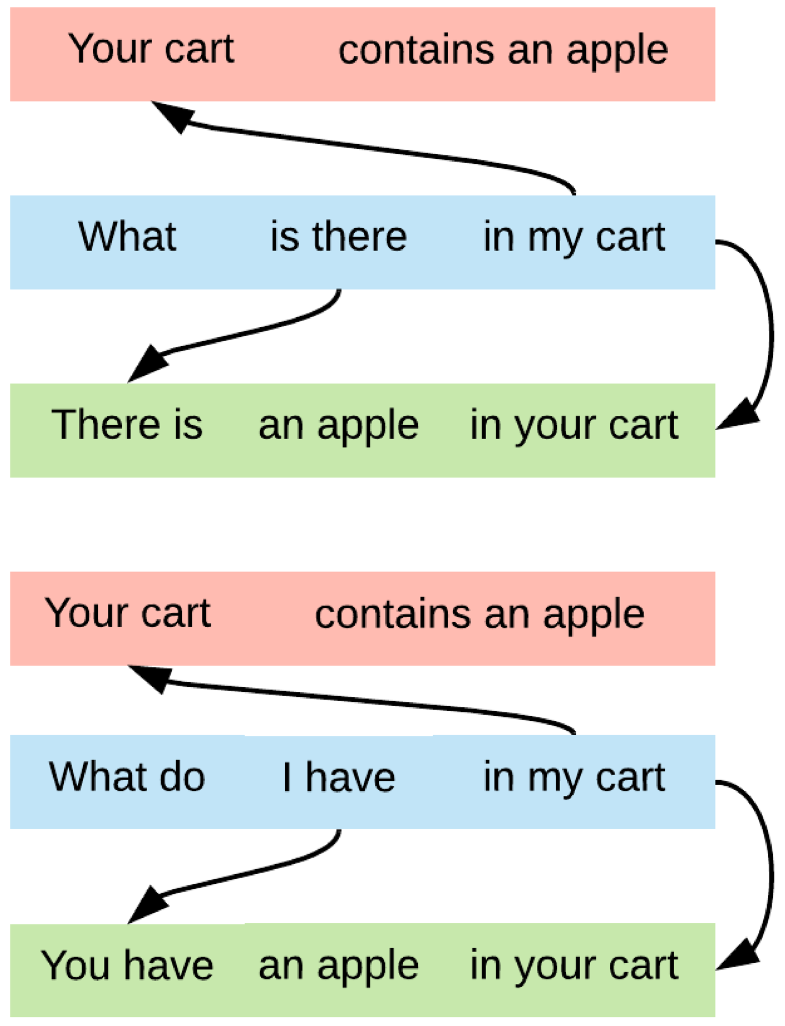

Figure 1: The blue question can be replied with the green entertaining answer but instead we have the red

We solve this problem by automatically generating responses and ensuring that they pay attention to the users way of speaking.

Dataset

Private dataset from WalmartLabs consisting of 20000 synthetic and live examples. The i^{th} data point d_i is a 4-tuple given by

$$d_i = (u_i, I_i, a_i, e_i)$$

where u_i is the user question string, I_i is the intent, a_i is the answer slot and e_i is any named entity if present. An example data row is {*"do you have milk"*, *"search"*, *"low fat milk priced at 3 dollars"*, *"milk"*}

- Test data set: 40 (4 paraphrases from 10 intents)
- Dataset is pre-normalized - all lower case and non-alphanum characters that are not part of a product name are removed.

RL Approach

We model the problem as a game play or a control problem where the state at any given point is defined by

$$S = w_i, w_2, ...w_n \quad (1)$$

where w_i s are the words of the response at that point in time. We define the following 2 types of operators on the state: $P(S)$ - generates a new permutation of S and $T(S_i, a)$ - transforms w_i, including deleting it, depending on the second parameter, a. We use Monte Carlo Tree Search to stochastically explore the search space. At any given time, the search state moves to a new state for which the following quantity is highest.

$$\text{UCB1} = v_i + C\sqrt{\frac{\log N}{n_i}} \quad (2)$$

For evaluating the final generated sentence we use a multi-objective function that rewards similarity to the users original utterance while penalizing parts of speech that are indicative of a question being asked. The reward R for a given state s is given as follows

$$R(s) = t(s) * (c * k(s) + (1 - c) * l(s) + b(s, \text{ref})) \quad (3)$$

where $t(s)$ is the output from a Naive Baye's classifier that determines the probability of s being a valid sentence, $k(s)$ and $l(s)$ are smoothed probabilities from language models and $b(s, \text{ref})$ is the BLEU score.

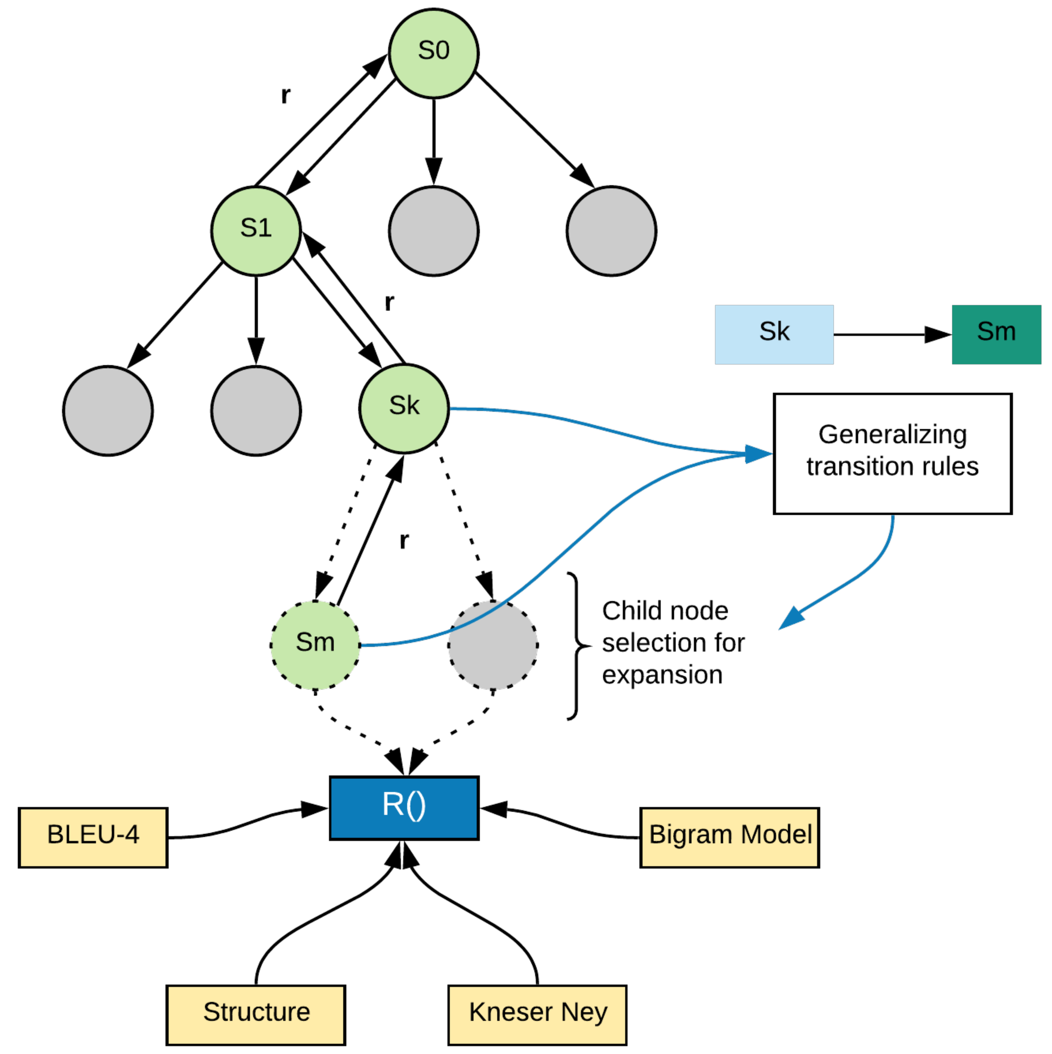

Figure 2: Response generation in MCTS guided by a multi-objective the Reward function R

Results

Length	Bandit	Score	Human
<= 5	UCT	0.7129	80.0%
	PUCT	0.6990	70.0%
	Random	0.4127	50.0%
$5-10$	UCT	0.6274	55.0%
	PUCT	0.6035	60.0%
	Random	0.3255	10.0%
>= 11	UCT	0.5535	40.0%
	PUCT	0.5932	40.0%
	Random	0.3881	0.0%

Table 1: Performance for different Bandit strategies

Generated Responses

Here are some of our best results

- **User:** do you have any apples that are sold individually
- **Agent:** i do have gala apples that are sold individually **[0.648]**

- **User:** which apples are sold individually
- **Agent:** gala apples are sold individually **[0.719]**

- **User:** can you get me some apples that are sold individually
- **Agent:** i can get you gala apples that are sold individually **[0.67]**

Discussions

- Though some results were encouraging, the reward function needs more tuning to correlate more strongly to human evaluation.
- There is no standard implementation of Kneser-Ney smoothing for trigrams. Current NLTK implementation is buggy.
- Though it is normally ill-advised to use BLEU score at sentence level, we managed to use it to our advantage with appropriate smoothing and defensive coding.
- Naive Baye's worked surprisingly well for valid vs invalid sentence classification, when the invalid sentences were generated by sampling randomly from the corpus.

Future Work

This is an area of active research at WalmartLabs and efforts to improve and develop this approach will continue beyond the end of this course. Following will be the main areas of focus

- In addition to deterministic set of transition rules and unsupervised NLG, we will also explore if we can use a stochastic action space and learn the best moves in a supervised setting.
- The reward function can be parameterized, with different weights for each of the component objectives, which can then be learned from a held out data set.
- We will explore Expert Iteration System that generalizes the learning from the state transitions, which can then be used as a feedback to MCTS at the simulation step.

References

[1] Kaori Kumagai, Ichiro Kobayashi, Daichi Mochihashi, Hideki Asoh, Tomoaki Nakamura, and Takayuki Nagai. Human-like natural language generation using monte carlo tree search.

[2] Ondrej Dusek and Filip Jurcícek. A context-aware natural language generator for dialogue systems.