Policy

Nurul Lubis

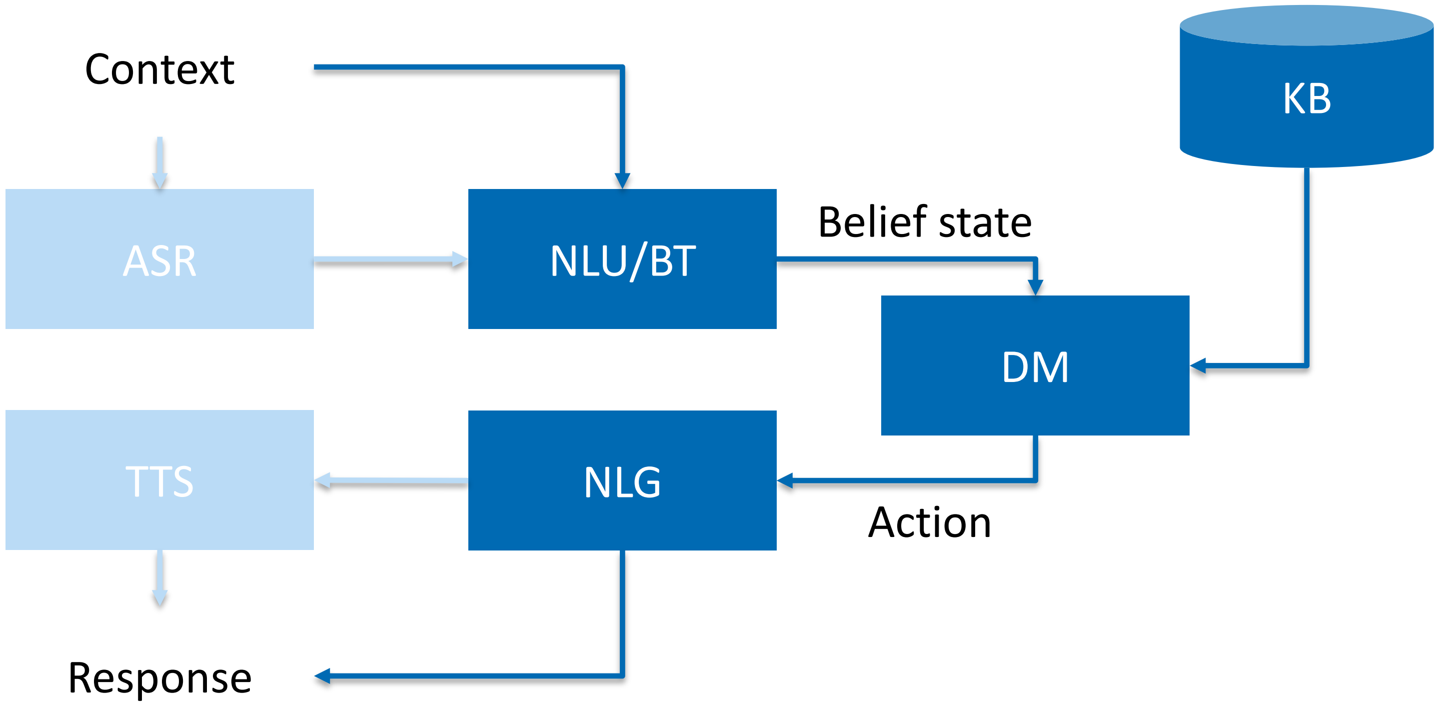
Dialogue Systems and Machine Learning Group

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

1

Statistical Dialogue Systems

Modular view of a dialogue system



HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

2

Policy

What is a policy?

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Informally:
  - A way for the machine to decide what to do at each point in time
- More formally:
  - A mapping from state to action

Policy in different tasks

- Games
- Autonomous driving
- Robotics
- Dialogue
- …

How to obtain a good policy?

Three learning paradigms

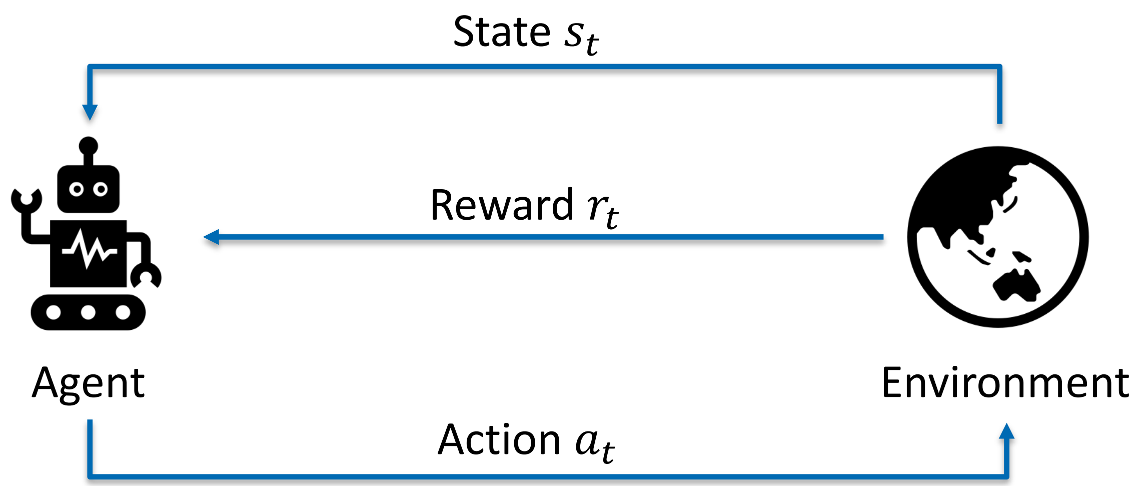
- Supervised learning
  - Provide a correct response to every possible input
- Unsupervised learning
  - Finding hidden structure in data
- Reinforcement learning
  - Learn from interaction, aim to maximize rewards

3

Reinforcement Learning

Reinforcement learning

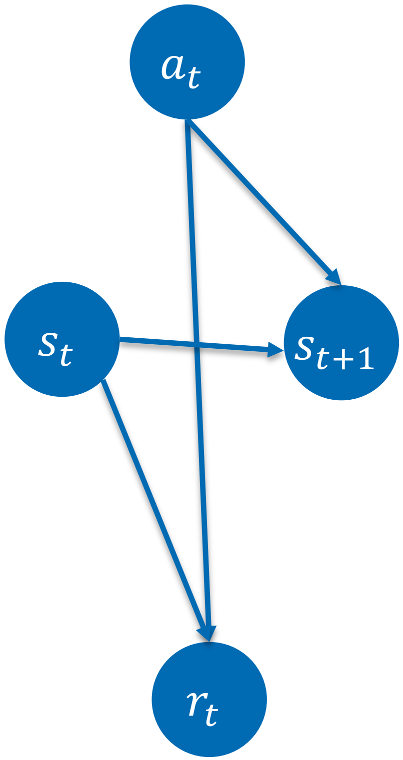
- Through interactions with the environment, the agent try to find the best policy based on some measure of reward.



- Huge number of interactions are typically needed
  - With dialogue systems, often a simulator is used in place of real users

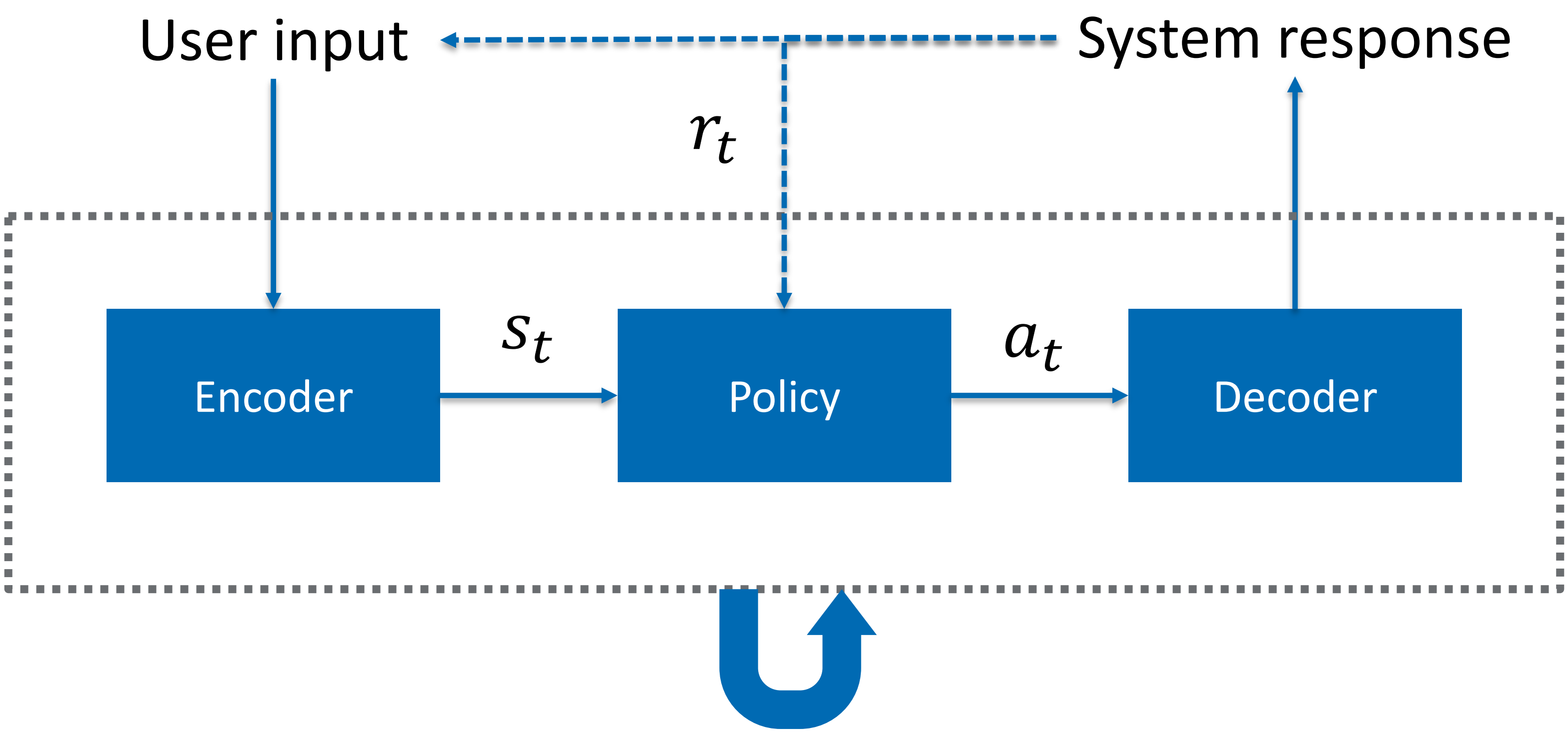
Formulating dialog as an MDP

[Levin and Pieraccini, 1997]



- s_t state
- a_t system actions
- r_t reward

Formulating dialog as an NN



Optimizing a policy

- **Return** R_t : *discounted* cumulative reward from that point onwards until termination

Under policy π:

- **Q-function** $Q_\pi(s_t, a_t)$: how good it is (measured through expected return) to take a a_t in s_t and then following π
- **Value-function** $V_\pi(s_t\ )$: Expected return of following π from s_t

Consider…

- Agent must plan to **maximize cumulative reward**
  - An action that has negative impact now may yield high reward in the future
  - However a sure reward may be more preferred than a potential reward
- Agent must balance between **exploration** and **exploitation**
  - Exploration is risky, but it is a way to gain new experience
  - Exploitation is safe, but agent may miss out on bigger reward in the unexplored space

Challenges in dialogue system optimization

1. Error in the dialogue system pipeline
   - Uncertainty
2. Infinite state and action space
   - Data and computation
3. Domain-dependent training
   - State and action space relies on ontology
   - New domain, new policy
4. Reward is not obvious
   - Human dialogue has multitude of facets, what is most important?

4

Tackling challenges in policy optimization for dialogue systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Handling uncertainty

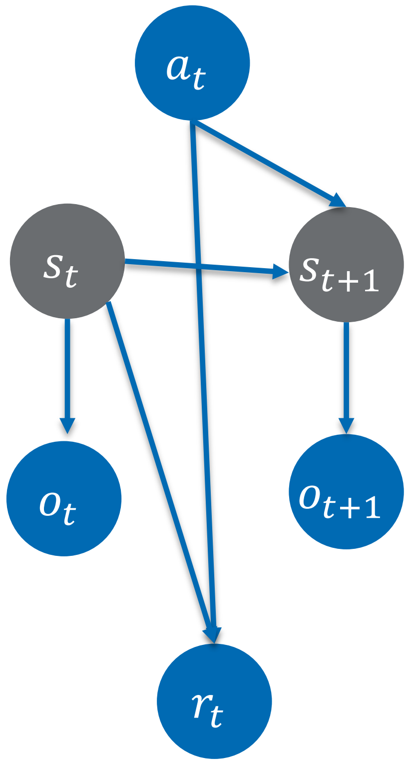
HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Two levels of uncertainty

- Input level: input to a dialogue system might be corrupted or only partially observable
  - E.g. ASR error, sensor imprecision, etc.
  - Need infer user intent from observation
- Output level: Uncertainty in estimating return
  - Return is a collection of random variables. In low data setting, expectation may high variance, i.e. estimation has high uncertainty
  - Need to consider this in learning

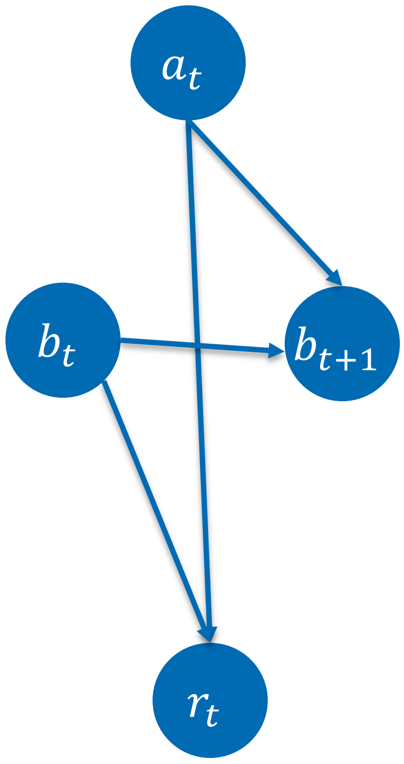
Modeling dialog as POMDP

[Young, 2006], [Williams and Young, 2007]



- s_t dialogue states (unobservable)
  - State generates o_t noisy observations
  - with observation probability $P(o_{t+1}|s_{t+1})$
- a_t system actions
  - Next state depends on s_t and a_t
  - With transition probability $P(s_{t+1}|s_t, a_t)$
- r_t reward
- Uncertainty can be modeled by considering distribution over unobservable states $b_t(s_t)$
  - Inference and optimization are tractable only for very simple cases [Kaelbing et al., 1998]

Modeling dialog continuous MDP



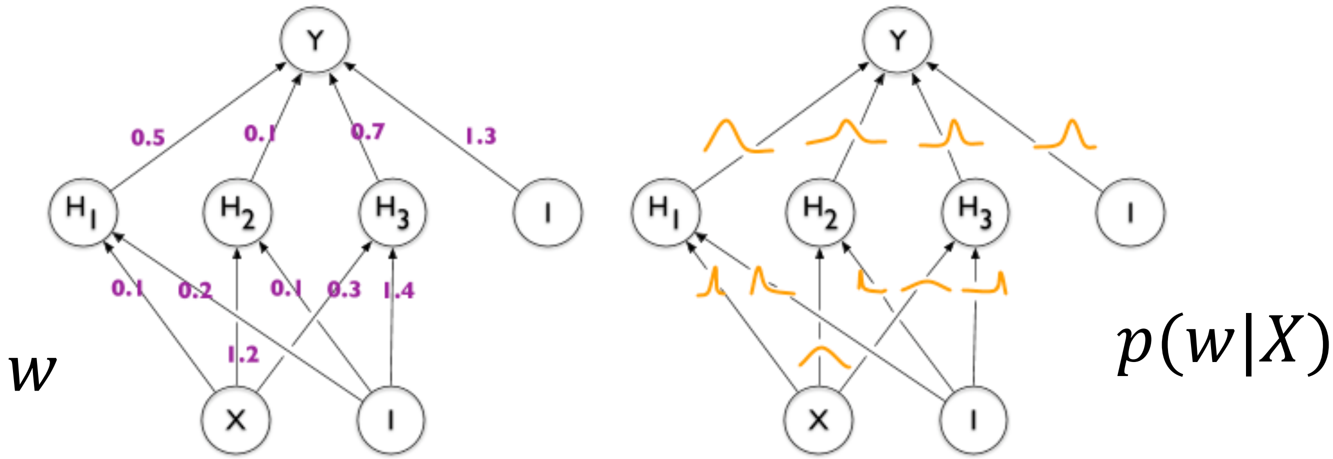
- A POMDP can be modeled as continuous MDP
- b_t belief state
  - Continuous distribution over possible states
  - $b_t = b(s_t)$
  - Belief state is supplied by belief tracker
- a_t system actions
- r_t reward
- This allows us to use standard MDP algorithms

A Gaussian process approach

- Uncertainty at output: can we model how certain we are about estimations?
- Q-function can be modeled as a Gaussian process (GP) [Engel et al., 2005]
  - GP: a non-parametric Bayesian model for function approx.
  - Incorporates prior knowledge through kernel function
  - Provides uncertainty meas. through variance of the posterior
- Optimal Q-function can be approximated with GP-SARSA algorithm [Gašić and Young, 2014]
  - Value estimation using a kernel function in the belief-action space
    - Choose a kernel that takes into account similarities of different parts of the space
    - If we encounter a point that is similar to previous experience, we could be more certain about our estimates
  - Use mean and variance to balance exploration and exploitation

Modeling uncertainty in neural networks

- Bayesian neural networks (BNNs): in place of single parameter w, use distribution conditioned by input X, i.e. $p(w|X)$ [Neal, 2012]
  - Yields infinitely many models
  - Sampling or variational inference methods is used for prediction



Source: https://sanjaykthakur.files.wordpress.com/2018/12/bayes_nn.png

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

[Tegho et al., 2017]

- Bayesian methods to extract uncertainty estimates
  - Variational inference methods: Bayes-by-backprop (BBQN), α-divergence, Bayesian inference with (concrete) dropout
- With DQN as the model [Mnih et al., 2015]

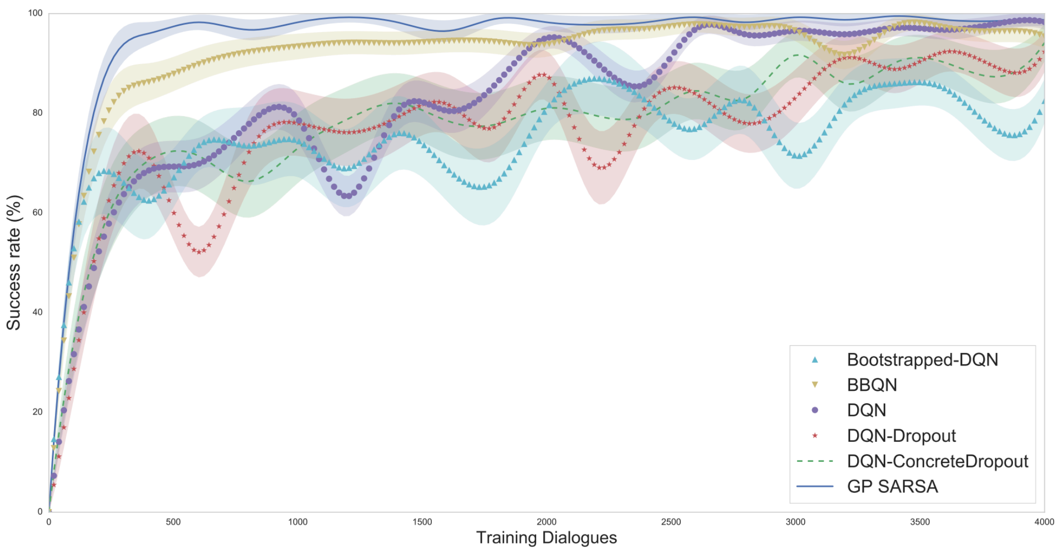


Figure 1: The success rate learning curves for all analyzed models under noise-free conditions.

- Only BBQN achieves comparable result
  - Complexity for NN $O(N)$ depends on #parameter
  - Complexity for GP-SARSA $O(nk^2)$ depends on #data points and #rep. data points

Summary

- Uncertainty is present in
  - Input level: noise, partial obervation
    - POMDP, continuous MDP
  - Output level: uncertainty in estimation
    - GP, BNN
- Remaining limitations
  - High computational cost, difficulty to train
  - Under-explored
    - Unique problem in dialogue, not present in game envs

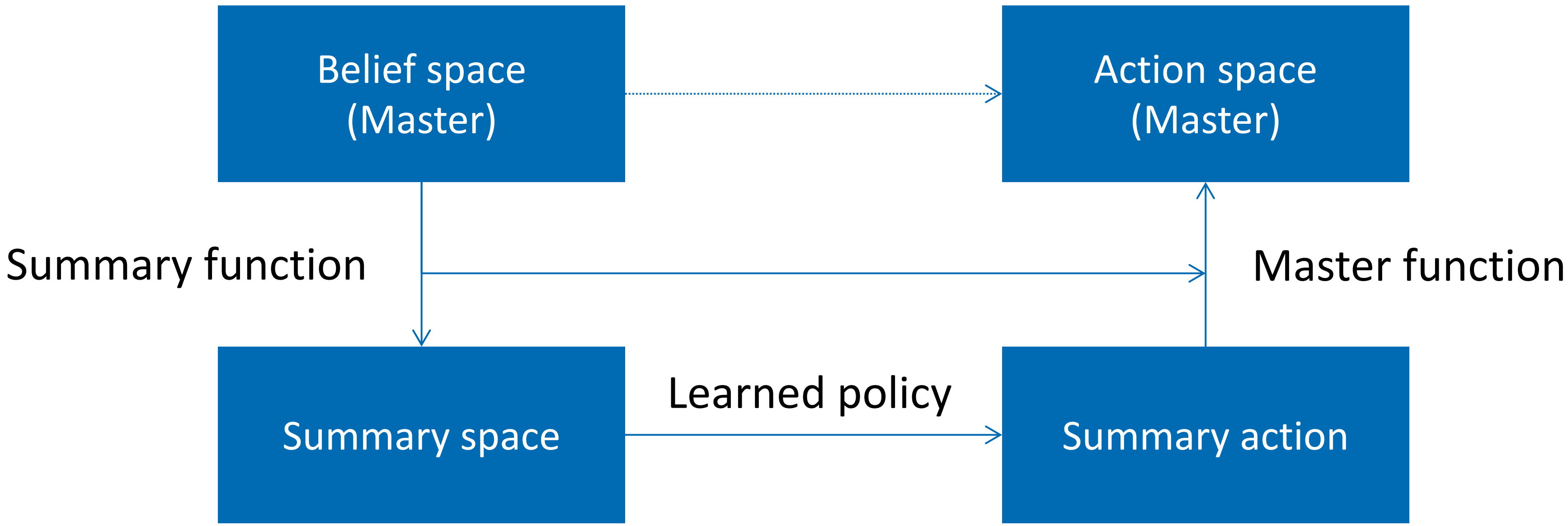
Handling infinite (or very large) spaces

Human dialogue is infinite

- In its purest human form, dialogue has infinite state, action, and trajectories
- To optimize a policy, need to formulate dialog as a problem that is tractable and solvable
  - Summarizing belief-action space
  - Decomposing decision making
  - Abstraction of action to shorten the trajectory
- Employ sample-efficient learning

Working on summary space

[Young et al., 2010]



Actor-Critic Experience Replay (ACER) for dialog

[Weisz et al., 2018]

- Employs two policies
  - Behavior policy μ for exploration
  - Main policy π optimized based on experience from μ

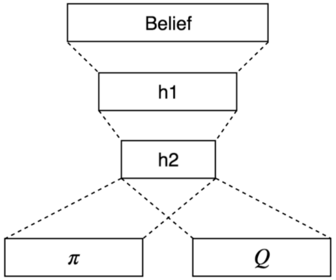


Fig. 1. ACER neural network architecture for dialogue management.

- Applies various methods to reduce bias and variance
  - Lambda-returns: balancing bias-variance
  - Retrace: estimate Q in a safe, efficient way with small variance
  - Recursive formulation of Q to reduce computational cost
  - and more

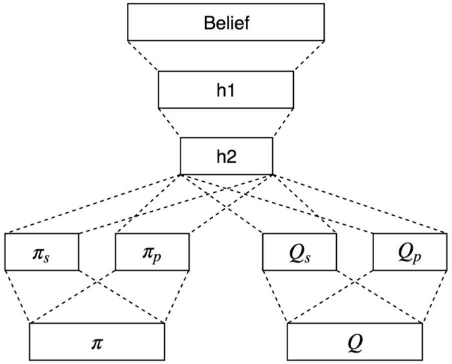


Fig. 2. Architecture of the actor-critic neural network for the master action space.

[Weisz et al., 2018]

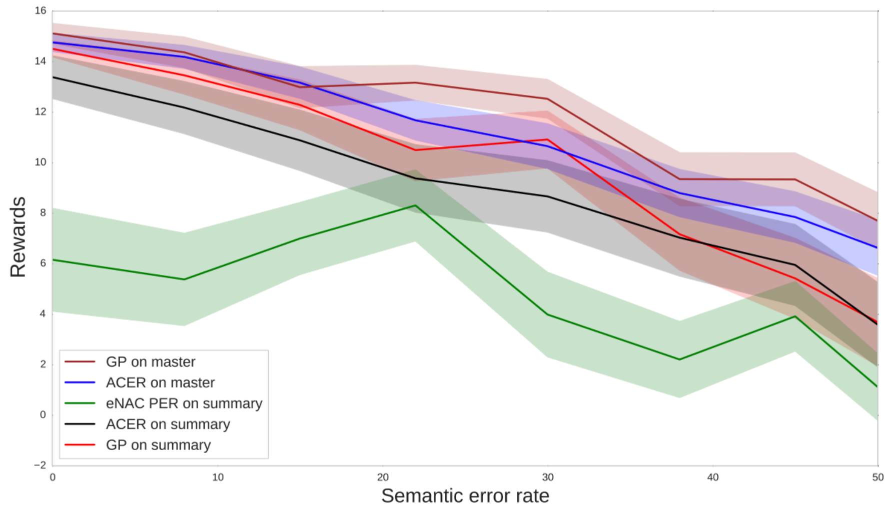


Fig. 14. Rewards of key algorithms when training them on 15% and testing them on varying error rates. Shaded areas represent a 95% confidence interval.

- Master action: 1035
- Summary action: 15

- Especially for high noise level, model trained in master space is more robust
  - Model learns mapping from summary to master action space
  - Learns decision making under uncertainty
- Handles large action spaces better

Feudal RL [Casanueva et al., 2018]

- Policy is modeled with DQN
- Decision making can be decomposed into two steps
    - Master policy π_m selects a sub-policy based with highest Q-value
        - Provide information actions under slot independent policy π_i
        - Gather information actions under slot dependent policy π_d
            - Comprises slot specific policies π_s
    - An action is chosen out of the selected subset to max. Q-value
- Each sub-decision deals with parts of the belief state, encoded heuristically

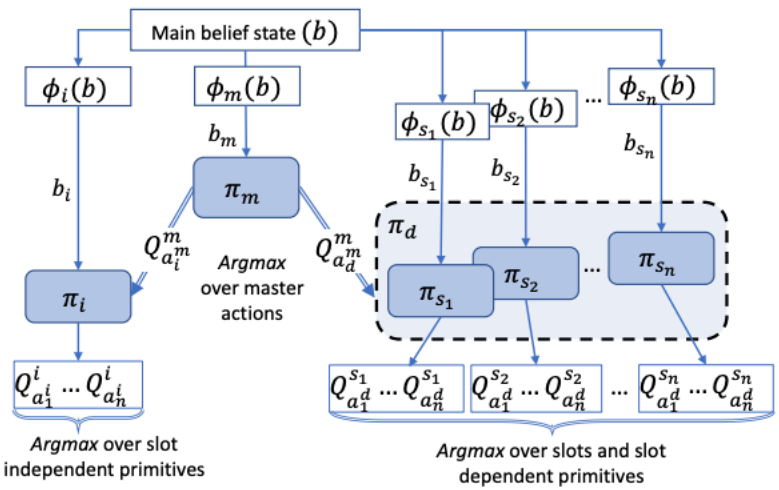


Figure 1: Feudal dialogue architecture used in this work. The sub-policies surrounded by the dashed line have shared parameters. The simple lines show the data flow and the double lines the sub-policy decisions.

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

LIDM [Wen et al., 2017], LaRL [Zhao et al., 2019]

- LaRL: Unsupervisedly induce action space $\mathbf{z}$ from data then perform RL on top
- Factorizing response generation $p(\mathbf{x}|\mathbf{c}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|\mathbf{c})$
  - Apply REINFORCE in the latent action space
  - Latent action shortens the RL horizon, decrease the action space dimensionality, and decouple decision making from language generation

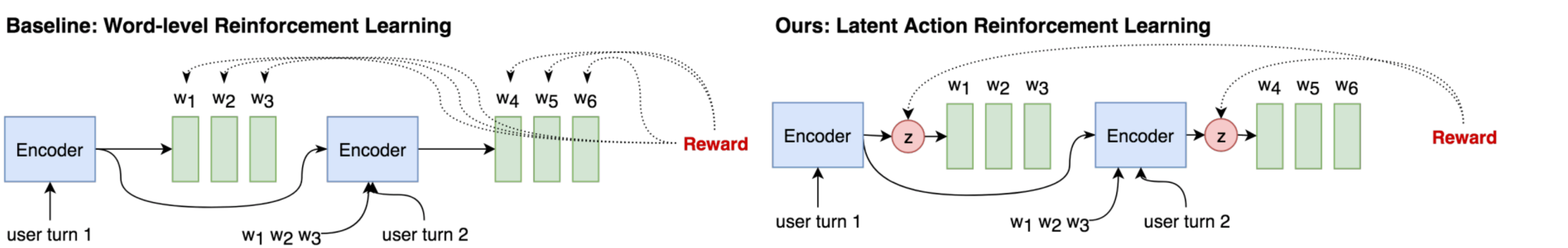


Figure 1: High-level comparison between word-level and latent-action reinforcement learning in a sample multi-turn dialog. The green boxes denote the decoder network used to generate the response given the latent code $\mathbf{z}$. Dashed line denotes places where policy gradients from task rewards are applied to the model.

Discrete or continuous latent action space?

- Two types of latent action $\mathbf{z}$
  - Continuous: M dimensional Gaussian multivariate
  - Categorical: M independent K-way random variables
- Models with categorical action consistently outperforms models with continuous one
  - Applying REINFORCE on cont. latent action is unstable
    - Latent space is unbounded
    - Exploration in cont. space in areas not covered in supervised re-training
  - Is assumption of a Gaussian distribution accurate?

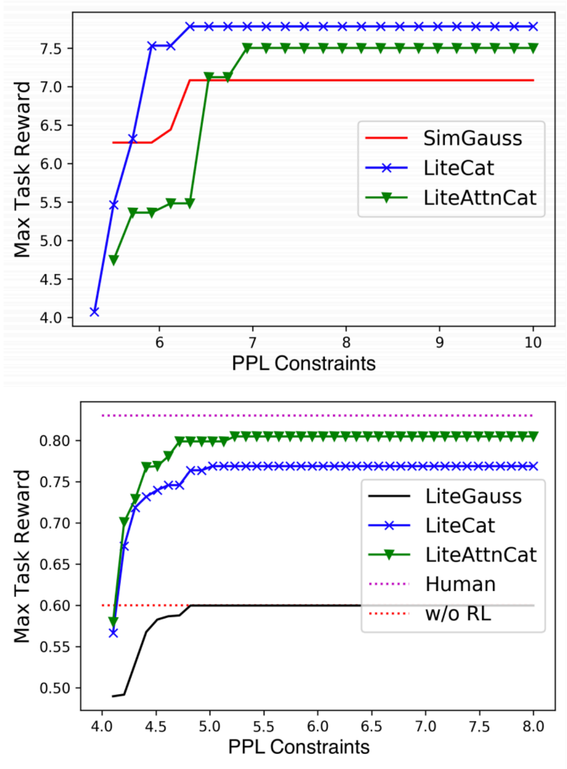


Figure 5: LCR curves on DealOrNoDeal and MultiWoz. Models with $\mathcal{L}_{full}$ are not included because their PPLs are too poor to compare to the Lite models.

Summary

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Very large spaces can be handled by
  - Factorization or partitioning of belief-action space
  - Employing sample-efficient methods
  - Decomposing decision hierarchically
  - Decoupling high level action (e.g. language generation) from decision making
- Can we perform RL for dialog in continuous action space?

Will that allow a more dynamic inference given an unseen state?

Formulating reward

Reward in dialogue systems

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- What can system use as reward?
  - In task-oriented dialogues, learning is typically aimed towards (domain-depenent) task success (TS)
  - Is that the best measure of a „good“ dialogue?
- Where do reward signal come from?
  - In case of TS: From user at the end of dialog
  - Can be intrusive, and need user to cooperate.
  - Sparse reward
    - One reward for the entire dialogue
    - Which actions are actually beneficial?

User satisfaction or interaction quality

[Ultes, 2019]

- User satisfaction is more domain independent
  - Reflects other aspects of the dialogue that underlies task success
  - Task success can only be obtained for pre-defined task
- More user-centered
  - Better represent the view of user‘s intent
  - Evaluates over all user experience
- Utilize domain-independent features to predict interaction quality, and use this as RL reward
  - Needs training data

On-line active learning

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

[Su et al., 2016]

- Jointly train dialogue policy alongside the reward model via active learning
  - Train Bi-LSTM unsupervised recurrent auto-encoder
  - Reward from GP in form of binary prediction of dialogue success

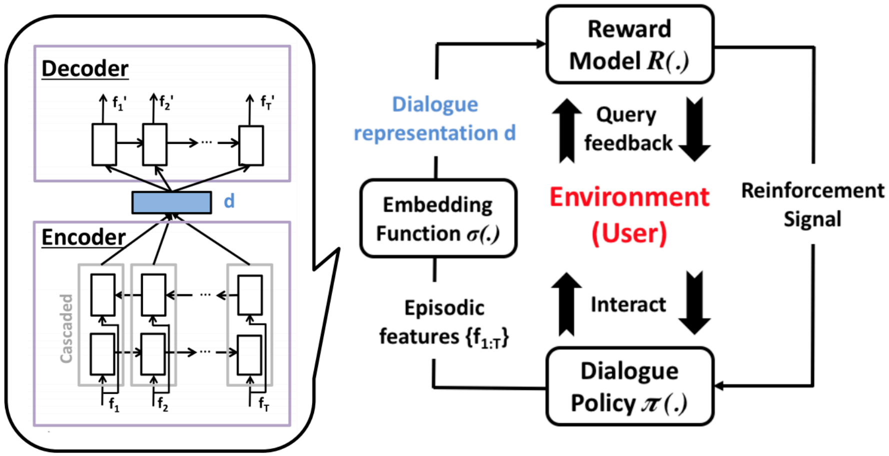


Figure 2: Schematic of the system framework. The three main system components dialogue policy, dialogue embedding creation, and reward modelling based on user feedback, are described in §3.

GP reward model

[Su et al., 2016]

- Takes continuous dialog representation $\mathbf{d}$ and a collection of previously classified dialogues $\mathcal{D}$
- Determines predictive mean and variance
- Decides whether it should seek user feedback based on a threshold of uncertainty
  - Reduce the need of user feedback
- Actually performs better than model trained with only human feedback

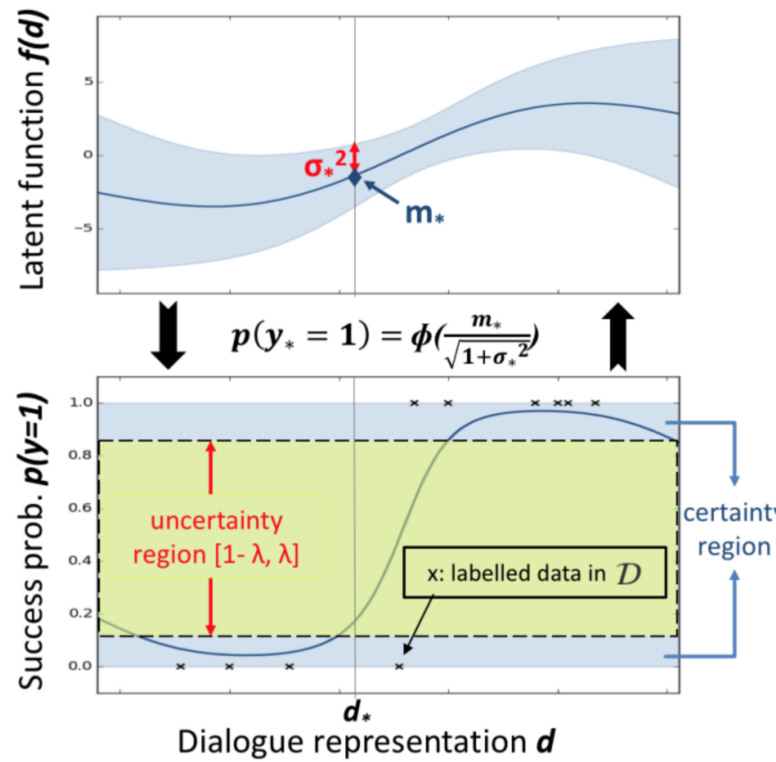


Figure 3: 1-dimensional example of the proposed GP active reward learning model.

Adversarial learning for reward estimation

[Liu and Lane, 2018]

- Relying on human feedback for reward
  - Inconsistencies
  - Non-cooperative user
- Learn rewards directly from dialogue samples and use in RL
  - Use adversarial learning framework
  - Generator: given current utterance, previous action, and dialog history, predict next action
  - Discriminator: predict the probability that current dialog will end successfully (based on similarity with human dialog)
    - Used as reward to optimize the generator

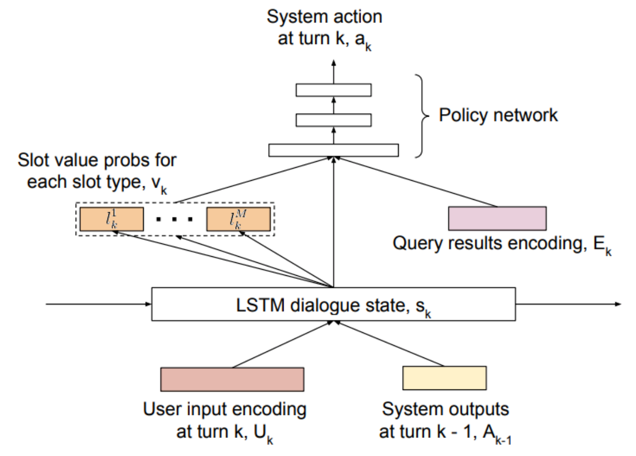


Figure 1: Design of the task-oriented neural dialog agent.

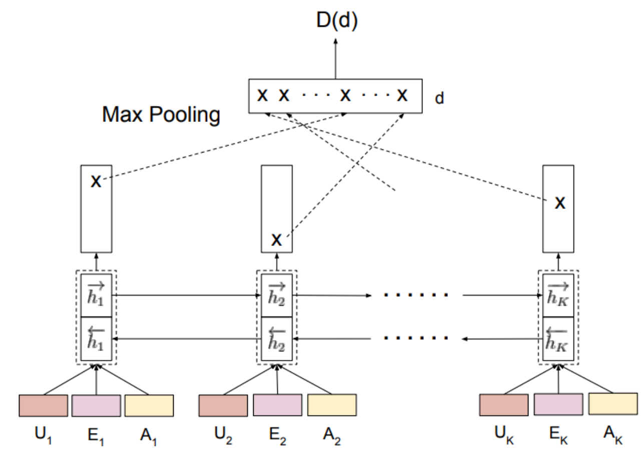


Figure 2: Design of the dialog reward estimator: Bidirectional LSTM with max pooling.

Adversarial learning for reward estimation

Algorithm 1 Adversarial Learning for Task-Oriented Dialog

1: **Required:** dialog corpus S_{demo}, user simulator U, generator G, discriminator D
2: Pretrain a dialog agent (i.e. the generator) G on dialog corpora S_{demo} with MLE
3: Simulate dialogs S_{simu} between U and G
4: Sample successful dialogs $S_{(+)}$ and random dialogs $S_{(-)}$ from $\{S_{demo}, S_{simu}\}$
5: Pretrain a reward function (i.e. the discriminator) D with $S_{(+)}$ and $S_{(-)}$ $\triangleright$ eq 8
6: **for** number of training iterations **do**
7:     **for** G-steps **do**
8:         Simulate dialogs S_b between U and G
9:         Compute reward r for each dialog in S_b with D $\triangleright$ eq 6
10:         Update G with reward r $\triangleright$ eq 7
11:     **end for**
12:     **for** D-steps **do**
13:         Sample dialogs $S_{(b+)}$ from $S_{(+)}$
14:         Update D with $S_{(b+)}$ and S_b (with S_b as negative examples) $\triangleright$ eq 8
15:     **end for**
16: **end for**

- Generator: Supervised pre-training on DSTC 2 data before interactive adversarial training
- Using model-based simulator as user
- Discriminator: pre-trained from dialog sample from generator and simulator
- Optimize generator and discriminator in turn

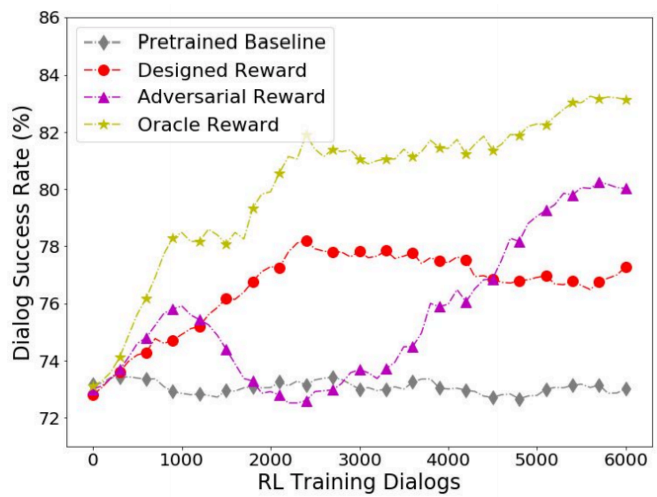


Figure 3: RL policy optimization performance comparing with adversarial reward, designed reward, and oracle reward.

Curiosity driven learning

- Curiosity as an intrinsic reward that drives agent‘s learning
  - Human learning are often not task-oriented, but simply driven by desire to explore the unknown
- Helps overcome reward sparsity
  - the reward comes from the agent
- More efficient state-action space exploration
  - Informed exploration, as opposed to random

Self-supervised prediction as curiosity for dialog

[Wesselmann et al., 2019]

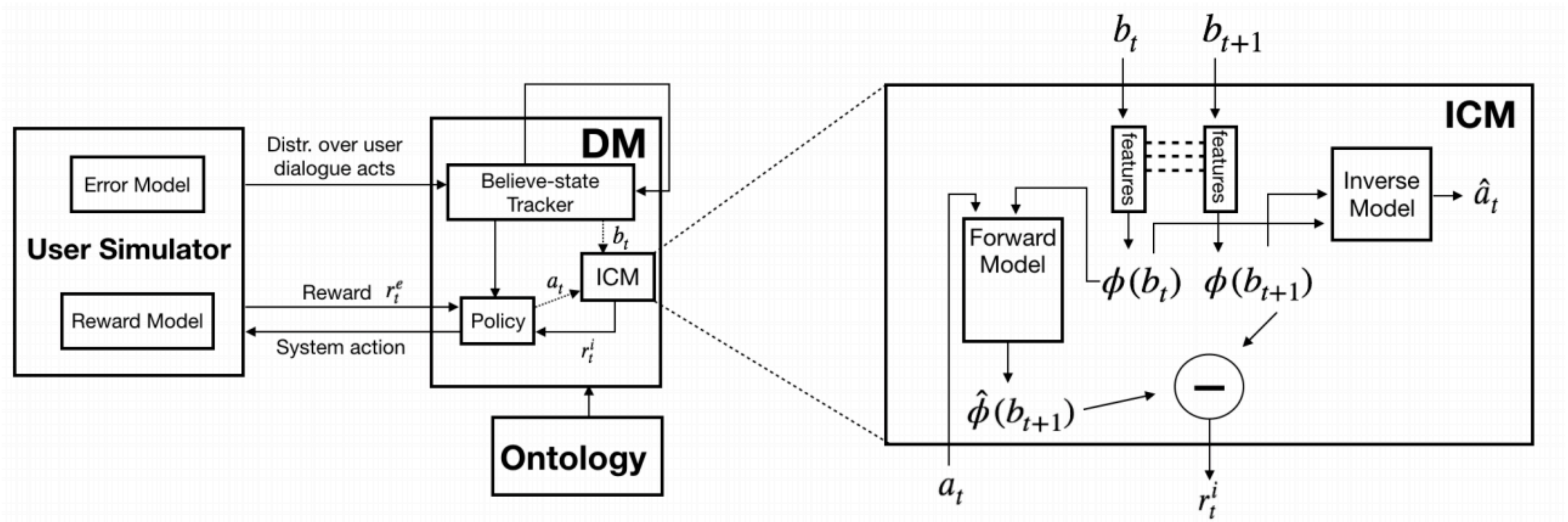


Fig. 1: Illustrated formulation for self-supervised prediction as curiosity in context with the DM. In belief-state b_t the agent interacts with the user by executing an action a_t sampled from policy π to get to state b_{t+1}. The ICM encodes belief-states b_t and b_{t+1} into features $\phi(b_t)$ and $\phi(b_{t+1})$, that are trained to predict a_t (inverse model). a_t and $\phi(b_t)$ are inputs for the forward model predicting the feature representation $\hat{\phi}(b_{t+1})$ of b_{t+1}. The prediction error is used as intrinsic reward signal r_t^i which can be used in addition to external rewards r_t^e. (this model is adapted from [5])

Summary

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- Sparse reward can be avoided by
  - Relying on intrinsic reward or reward prediction
- Creative thinking of what constitutes a „reward“
  - Curiosity, interaction quality has shown to be useful for learning
  - Train a model to abstract these signals from dialog sample
- Can we expand the definition of reward to other human qualities, e.g. emotion?

Domain adaptation

Domain adaptation

- State-action space definition relies on domain-specific ontology
  - Policy is domain-specific. Meaning, new domain, new policy
- Training a DS is expensive
  - Data, computation, human feedback
- Can combine policies or adapt a policy from one domain to another?
  - Exploit similarities between domain
  - Train a domain-generalizable model

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Combining GP policies [Gašić et al., 2016]

- Decompose dialogue policy into a set of topics
- First learn a generic policy from small data, i.e. a general policy accross domains
  - Prediction of Q is learned using kernel that spans accross the combined belief-action space
- A specific policy can be derived for each topic given the generic policy and more data
  - E.g. after deployment

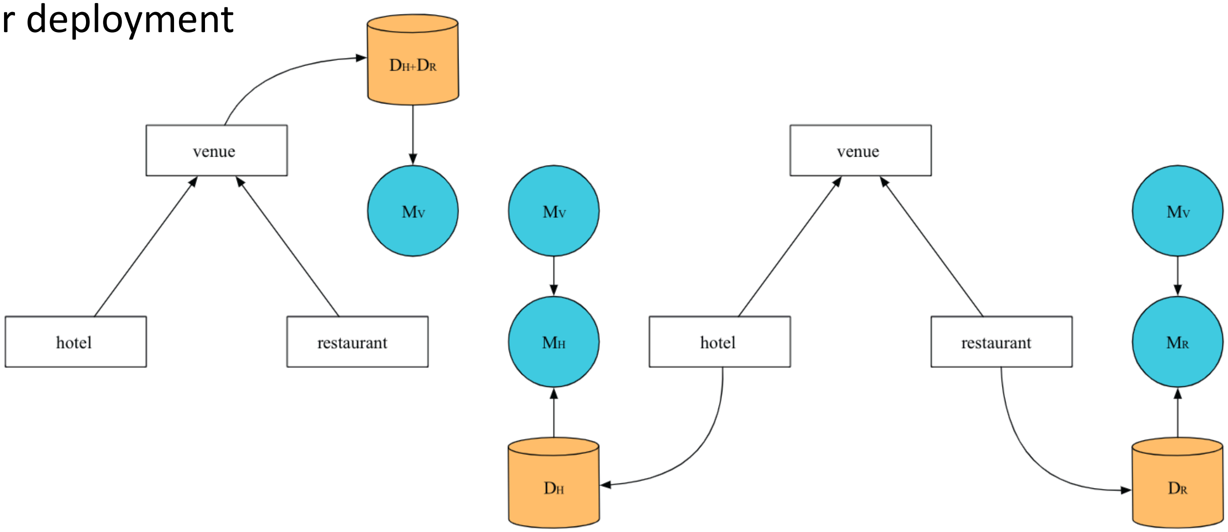


Fig. 1. Training a generic venue policy model M_V on data pooled from two subdomains $D_R + D_H$ (left); and training specific policy models M_R and M_H using the generic policy M_V as a prior and additional in-domain training data (right).

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Combining GP policies [Gašić et al., 2016]

- A way to combine estimators that have been trained on different datasets
  - Each member estimates their Q-function, and a gating mechanism is used to combine these outputs
- Multi-domain manager
  - Unlike distributed policy, possible to combine domain with no shared slots
  - Calculate kernel function between belief state and action from the domains
- Multi-agent learning
  - Reward is distributed to agent to optimize each of their policy
    - Different distribution schemes

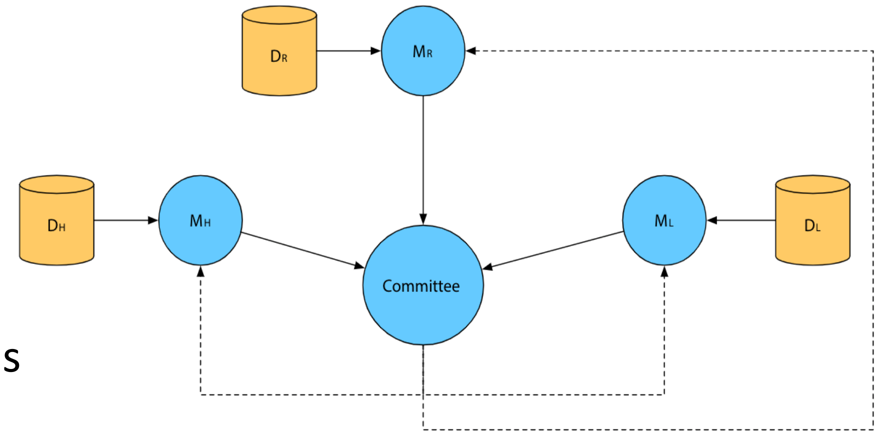


Fig. 4. Multi-agent policy committee model.

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Zero-shot NLG in dialogue [Zhao et al., 2018]

- Project response wrt to context and dialog label (separately) into a shared space
  - Training in turn to minimize distance between resp-context and resp-dialogue label
- Produces an action space that is shared between domains

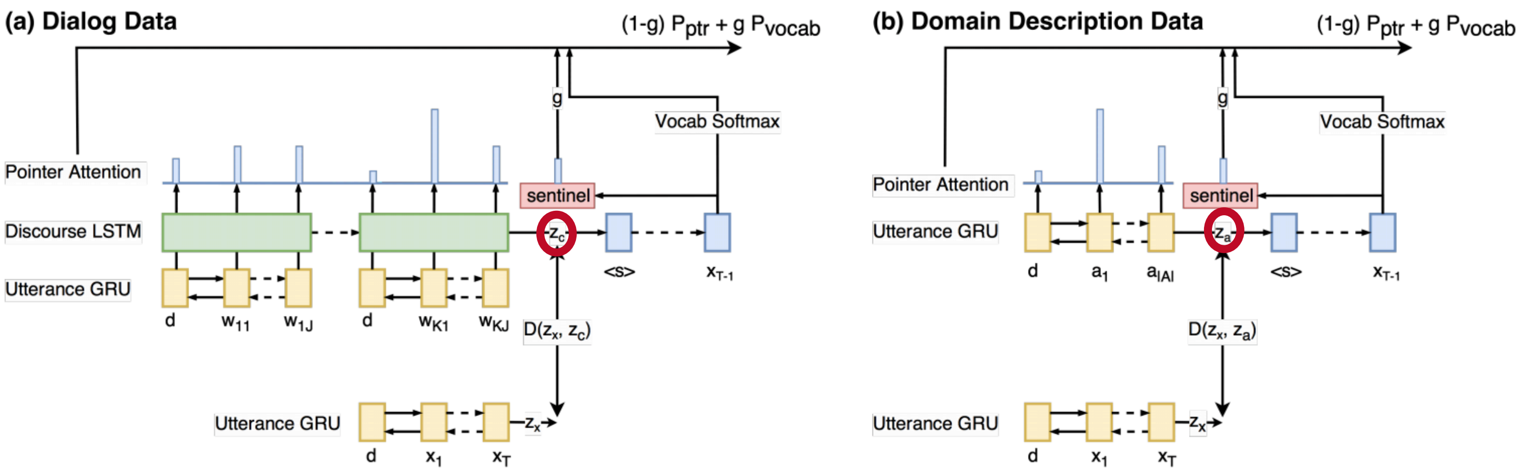


Figure 2: Visual illustration of our AM encoder decoder with copy mechanism (Merity et al., 2016). Note that AM can also be used with RNN decoders without the copy functionality.

HEINRICH HEINE UNIVERSITÄT DÜSSELDORF

Action-matching algorithm [Zhao et al., 2019]

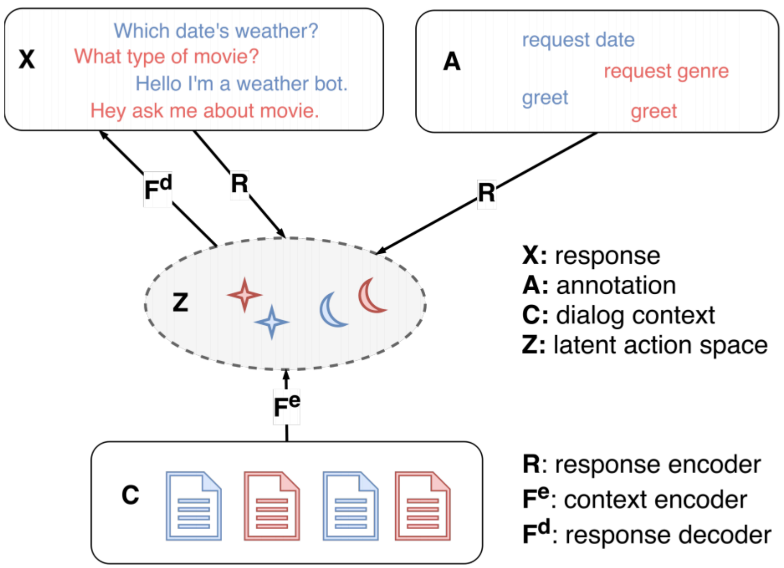


Figure 1: An overview of our Action Matching framework that looks for a latent action space Z shared by the response, annotation and predicted latent action from $\mathcal{F}^e$.

- Model performance significantly improves on
  - Unseen slot, unseen NLG, new domain
  - As well as in-domain test
- Ability to generalize to different levels of unseen data

Summary

- Domain transfer can be done by
  - Learn specialized policy on top of generic one
  - Employing a committee over multiple policies
  - Defining a shared state-action space between domains
- Domain adaptation relies of dialog data. Can we utilize unstructured world knowledge for domain transfer?

Closing

Open questions

Human‘s dialogue model is quite sophisticated!

- Modeling and utilizing uncertainty estimates
  - Is there a more computationally efficient model?
  - Can we pass uncertainty to NLG? Can we incorporate uncertainty from NLG in decision making? Can we express uncertainty through in NLG to aid learning?
- RL in continuous action space
  - Why has RL in continuous space not been succesful?
  - Can we induce an action space that is continuous and fluid? Contains knowledge? Allows inference in unfamiliar state? Reduce performance dependence on NLG?

Open questions

Human‘s dialogue model is quite sophisticated!

- Robust, human inspired reward
  - What makes a quality dialogue? Should we pay attention to different aspect at different times?
  - How can we handle noise that comes from human feedback? Or avoid having it in the first place?
- Domain adaptation
  - Can we adapt to new domain using unstructured data? i.e., can we disentangle learning about a domain and learning to talk about it?

Thank you!

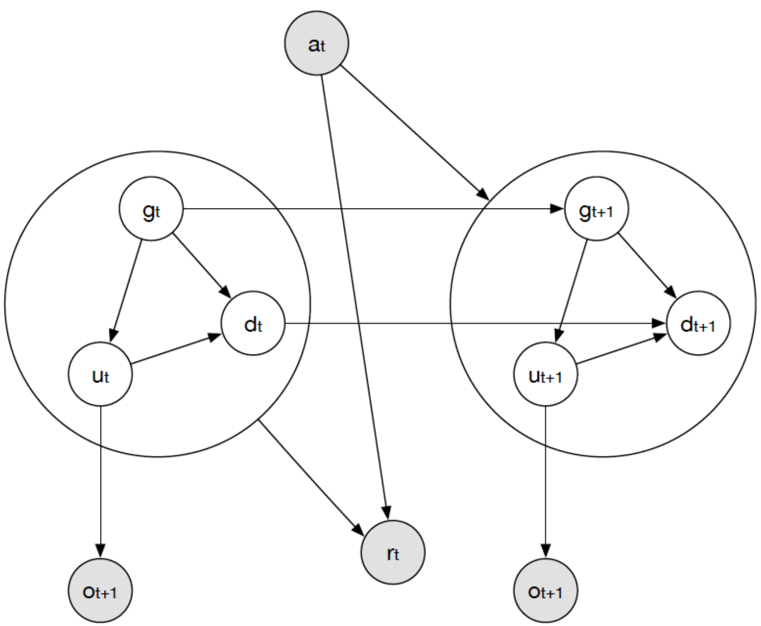
Defining the environment

- State space
  - Collection of information which describes the environment at a certain point in time
  - All possible states in the environment makes up the state space
- Action space
  - Possible actions that the system can take in the environment
  - Agent‘s actions will affect the state of the environment
- Reward
  - Some goal that drives the agent‘s actions

Hidden Information State Model

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

- HIS decomposes dialogue state into conditionally independent elements
  - User goal, user action, and dialog history
- Over the course of the dialog user goal is partitioned into mutually exclusive sets

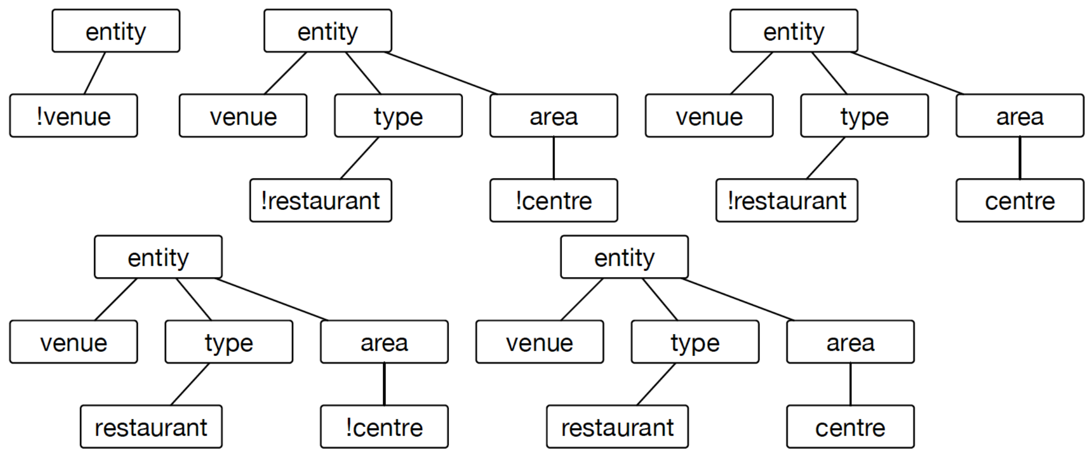


System: How may I help you?
request(task)
User: I'd like a restaurant in the centre.
inform(entity=venue,type=restaurant, area centre)

entity=venue

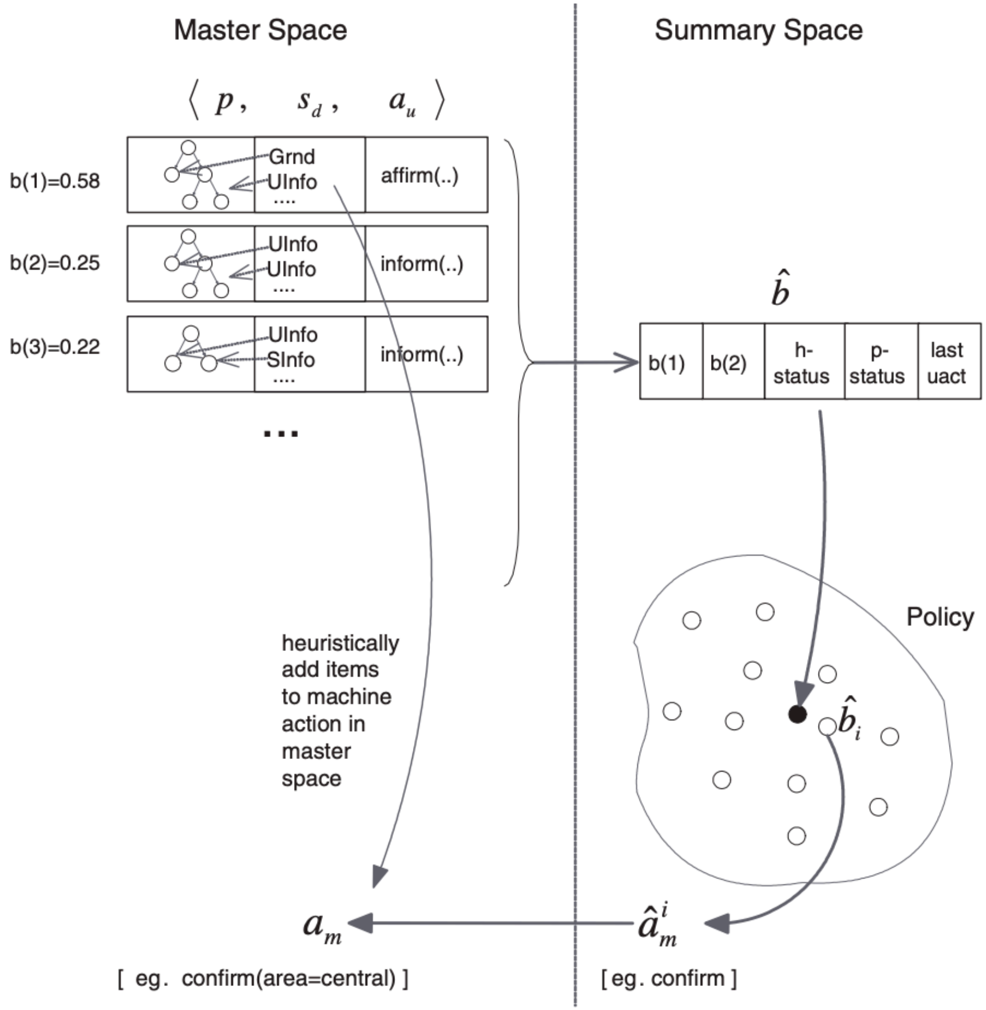
area=centre

type=restaurant



HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Master-summary mapping [Young et al., 2010]



- Belief in the master space is the distribution over hypothesis
  - combination of user act, partition and history
- Belief state in the master space is summarized with some heuristics
- The summary belief is used by the policy to decide on the action in the summary space
- The summary action is mapped back into master space by inferring the slot-value from the master belief

Using predicted IQ as reward in RL

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Table 1: The parameters used for IQ estimation extracted on the exchange level from each user input plus counts, sums and rates for the whole dialogue (#,%,Mean) and for a window of the last 3 turns ({·}).

	Parameter	Description
Exchange level	ASRRecognitionStatus	ASR status: *success*, *no match*, *no input*
	ASRConfidence	confidence of top ASR results
	RePrompt?	is the system question the same as in the previous turn?
	ActivityType	general type of system action: *statement*, *question*
	Confirmation?	is system action confirm?
Dialogue level	MeanASRConfidence	mean ASR confidence if ASR is success
	#Exchanges	number of exchanges (turns)
	#ASRSuccess	count of ASR status is success
	%ASRSuccess	rate of ASR status is success
	#ASRRejections	count of ASR status is reject
	%ASRRejections	rate of ASR status is reject
Window level	{Mean}ASRConfidence	mean ASR confidence if ASR is success
	{#}ASRSuccess	count of ASR is success
	{#}ASRRejections	count of ASR status is reject
	{#}RePrompts	count of times RePromt? is true
	{#}SystemQuestions	count of ActivityType is question

- Model: Bi-directional LSTM with attention
- Data: LEGO corpus
  - Real users
  - 200 dialogues, 4,8k turns
  - Each turn is labeled by 3 experts
- Performance: 0.54 UAR, eA 0.94
- The predicted IQ is then used for RL.

Compared to that trained with task success, it yields:

  - higher average user satisfaction
  - comparable task success rate