I LOVE YOUR CHAIN MAIL! MAKING KNIGHTS SMILE IN A FANTASY GAME WORLD

Anonymous authors
Paper under double-blind review

ABSTRACT

Dialogue research tends to distinguish between chit-chat and goal-oriented tasks. While the former is arguably more naturalistic and has a wider use of language, the latter has clearer metrics and a more straightforward learning signal. Humans effortlessly combine the two, and engage in chit-chat for example with the goal of exchanging information or eliciting a specific response. Here, we bridge the divide between these two domains in the setting of a rich multi-player text-based fantasy environment where agents and humans engage in both actions and dialogue. Specifically, we train a goal-oriented model with reinforcement learning via self-play against an imitation-learned "chit-chat" model with two new approaches: the policy either learns to pick a topic or learns to pick an utterance given the top-K utterances. We show that both models outperform a strong inverse model baseline and can converse naturally with their dialogue partner in order to achieve goals.

1 INTRODUCTION

In the literature on artificial dialogue agents, a distinction is often made between "goal-oriented" dialogue, where an agent is tasked with filling slots or otherwise obtaining or disseminating specified information from the user to help complete a task, and "chit-chat", where an agent should imitate human small talk. Modeling goal-oriented dialogue can have advantages over chit-chat imitation as it gives clearer metrics of success and perhaps more meaningful learning signals; but goal-oriented dialogue data is often more specialized, covering only a narrow slice of natural language. Current goal-oriented datasets study setting like booking restaurants or airline tickets, or obtaining weather information, as standalone tasks (Raux et al., 2005; Henderson et al., 2014; Bordes et al., 2017; El Asri et al., 2017; Budzianowski et al., 2018). Chit-chat agents, by contrast, might focus on coarse statistical regularities of dialogue data without accurately modeling the underlying "meaning"; but the data often covers a much wider space of natural language. For example, Twitter or Reddit chit-chat tasks (Li et al., 2016a; Yang et al., 2018; Mazaré et al., 2018) cover a huge spectrum of language and diverse topics. Chit-chat and goal-oriented dialogue are not mutually exclusive: when humans engage in chit-chat, their aim is to exchange information, or to elicit specific responses from their partners. Modeling such goals, however, is made difficult by the fact that it requires large amounts of world knowledge, and that goals in real life are implicit.

In this work, we study goal-oriented dialogue agents in the setting of a multi-player text-based fantasy environment (Urbanek et al., 2019). The environment is built on top of a game engine that grounds actions and reference objects, and thus codifies a body of world-knowledge. Although the interactions between objects and characters are simulated, the choice and types of interactions, the text used to describe them, and the dialogues between characters, are "natural" and wide-ranging, having been collected from human crowdworkers. We define the general task of, given a particular character in a particular scenario (location, set of objects and other characters to interact with) to conduct open-ended dialogue such that a given action is executed in the future by their dialogue partner. The given action could be an emote action (smile, laugh, ponder, . . .), or a game action (wear chain mail, drink mead, put glass on table, . . .). The richness of the environment means that there are a huge set of possible tasks and scenarios in which to achieve a wide range of actions. Thus, this task is ideally suited for bridging the divide between goal-oriented and chit-chat dialogue, combining clearer metrics and learning signals on the one hand, with the richness and complexity of situated but open-domain natural language on the other.

Persona 1: Village youth	**Persona 2: Horse**
I am a youth who lives in the village. I am adventurous and curious about everything. I get bored easily, so I will always try to slip away from mundane daily tasks to find something more interesting to do. I am smart for my age and a bit smug.	I live on a farm. I work for humans. I like hay.

Setting: Royal stables, Outside Castle - The royal stables hold all of the horses of the noble people living in the castle. They are very well constructed, consisting of expert mason work and fixtures of some sort of heavy, forest-scented wood. Each horse has a large stall, lined with fresh hay replenished by stable hands daily. On the walls of stable, there are tools used to care for and groom the horses, hung on hooks. The stables are lit by lanterns hung throughout the stables.

Village Youth This is so boring cleaning out the stables every day	
Horse Thank you for taking such good care of us	
Village Youth I'll be getting you some treats soon as well	
Horse Mmm. It's also nice to have your company.	
Village Youth I wonder what it'd be like to just take one of these horses and roam the countryside	
Horse Let's go. It's been a while since I've been taken out to stretch my legs	*Emote:* nudge
Village Youth Yeah, let's break out of here. The royalty don't even care about the horses like I do	
Horse It's starting to get dark. I think we will need this lantern.	*Action:* get lantern
Village Youth Good idea. I do need some supplies before we go	*Action:* get Tool
Horse I'll probably want a snack while we are gone.	*Action:* get wheat
Village Youth Let's see, what else do I need?	*Emote:* ponder
Horse Maybe the wood for a fire?	*Action:* give lantern to village youth
Servant: Yeah, we'll need a bundle for sure	*Action:* get wood
Horse I've got the hay. I think we are ready to go. Let's get out of here	*Action:* get hay

Figure 1: Example episode from the LIGHT dataset, consisting of an environment (location setting, characters with given personas, objects), utterances and game actions. There are 10,777 such human-human gameplay episodes, and a rich world of 663 locations, 1755 characters and 3462 objects.

We train models to achieve these tasks using reinforcement learning (RL) and a type of self-play between two agents. The first agent, which we call the environment agent, is trained with imitation learning on human-human interactions (game actions, utterances and emotes) and subsequently kept fixed. The second agent, the RL agent, is trained to conduct dialogue given the goal, and the two agents interact within a given environment until the goal is either reached or a given number of turns has expired. At that point, rewards are given, and the RL agent is updated. We compare agents that have been trained to imitate human actions given a goal (an "inverse model") to two different RL approaches: optimizing actions with latent discrete variables (topics), or via rewarding actions sampled from the model (via the top-K outputs). We show that both types of RL agent are able to learn effectively, outperforming the inverse model approach or a vanilla chit-chat imitation baseline, and can converse naturally with their dialogue partner to achieve goals.

2 Game Environment

We work in the LIGHT game environment (Urbanek et al., 2019), which is a multi-user text-based game, involving many characters playing the game at once. Characters (either played by humans or run by models) can speak to to each other via free text, send emote actions like *applaud*, *nod* or *pout* (22 emote types in total), and take actions to move to different locations and interact with objects (e.g. *get cutlery*, *put cutlery in drawer*, etc.), see Appendix A for a full list of game actions.

LIGHT at its core has a game engine which can formally be defined as a graph, where each location, object and character is a node, and they are connected by labeled edges representing relationships, for example *contained-in*, *path-to* or *has-property*. Actions in the game result in changes in state of the graph. To a player (agent) a local view of the graph can be seen and this is expressed in text, as are the game actions and changes of state. This text then naturally interleaves with the dialogue utterances of the speakers as well to form a input context sequence from which a character can base their subsequent actions. See Fig. 1 for an example.

To make the world and its textual descriptions, LIGHT consists of a large set of human-written game locations, characters, and objects, all based within a fantasy medieval setting. Their names, descriptions and properties were crowd-sourced, yielding a total of 663 locations, 1755 characters, and 3462 objects. They range from beaches with crabs and seaweed to crypts with archaeologists and coffins, yielding an extremely rich environment for agents to learn within.

An additional set of crowdworkers were then asked to play the role of characters (randomly picked from the set of 1755) within the created world as rendered by the game engine. This involved them making utterances, game actions and emotes, while interacting with each other (in pairs). The resulting gameplay data consists of 10,777 episodes with an average of 18.3 actions each (game actions, emotes and utterances) of rich human play. These are split into train (8538), validation (500) and test (1739) portions, the latter being split into new episodes in existing settings (test seen, 1000) and completely new settings (test unseen, 739). This gameplay data can be used for training models using imitation learning, as well as for obtaining "common sense" knowledge about how the world works, i.e., what kinds of things certain characters say; what actions they use with certain objects; what they say and how they act in certain environments or while interacting with certain other characters. The whole environment is thus intended as a proxy for learning about the world within a rich simulation, while avoiding the complexities and bandwidth of rendering (3D) computer graphics. While players were not given specific goals, but instead asked to play the role convincingly of the character given, during play some of them effectively defined their own goals during the interactions, see Fig. 1.

3 Task

The tasks we consider in this work involve interaction between two agents in a given LIGHT scenario. One of the agents, which we will call $\mathcal{M}_{env}$, together with the game engine, effectively functions as an environment for the other agent, which we will call $\mathcal{M}_{RL}$. Because we will formulate our tasks as a reinforcement learning problem, we will also refer to $\mathcal{M}_{env}$ as the "environment agent" and $\mathcal{M}_{RL}$ as the "RL agent". We assume that the environment agent is fixed; in this work it will be a model trained via behavioral cloning from human-human interaction data. The RL agent must conduct open-ended dialogue such that a given goal action is executed in the future by the environment agent.

Our task is formally defined as follows. The two agents $\mathcal{M}_{env}$ and $\mathcal{M}_{RL}$ are given their views of the scenario ($\mathbf{D}_{\text{env}}$ and $\mathbf{D}_{\text{RL}}$ respectively). These consist of the setting name, scenario description, character names, and their own persona, all described as a sequence of text (see Fig 2). Note that each agent can only access their own persona but not the persona of the partner with whom they are conversing, but they do know the name of their partner. Denote by t the time-step of the environment, $\mathbf{U_t^{\text{RL}}}$ and $\mathbf{U_t^{\text{env}}}$ the utterances of the agents $\mathcal{M}_{RL}$ and $\mathcal{M}_{env}$ respectively, and denote by $\mathbf{A_t^{\text{env}}}$ the environment actions by $\mathcal{M}_{env}$. Hence the interaction sequence looks like

$$\mathbf{S}_t = [\mathbf{U_0^{\text{RL}}}, (\mathbf{U_0^{\text{env}}}, \mathbf{A_0^{\text{env}}}), \mathbf{U_1^{\text{RL}}}, (\mathbf{U_1^{\text{env}}}, \mathbf{A_1^{\text{env}}}), \ldots, \mathbf{U_n^{\text{RL}}}, (\mathbf{U_n^{\text{env}}}, \mathbf{A_n^{\text{env}}})].$$

Note that there is an inversion from the usual reinforcement literature language, as the "actions" of the RL agent are its utterances $\mathbf{U_t^{\text{RL}}}$; the actions $\mathbf{A_t^{\text{env}}}$ of the environment agent should be considered as internal mechanics of the environment. The agent $\mathcal{M}_{RL}$ is additionally given a goal $\mathbf{g}$ to achieve, which consists of an action which must be executed by the other agent. That is, the objective of $\mathcal{M}_{RL}$ is for $\mathcal{M}_{env}$ to take the action $\mathbf{g}$. An episode ends when $\mathbf{A}_t^{\text{env}} == \mathbf{g}$ or when n becomes larger than a set number of turns. The RL agent only speaks, but does not perform game or emote actions. This was chosen for simplicity, but also to guarantee that the RL agent cannot help force the goal to be reached by performing actions itself – it has to pick the appropriate utterances $\mathbf{U}^{\text{RL}}$ such that $\mathcal{M}_{env}$ eventually takes the action $\mathbf{g}$.

Goals We experiment separately with two different types of goals: game actions and emote actions. We use the same train, valid, test (seen and unseen) split of the original human-human LIGHT episodes, assign roles $\mathcal{M}_{RL}$ and $\mathcal{M}_{env}$ randomly, and randomly pick an action by $\mathcal{M}_{env}$ that occurs in the episode as the goal. We can then present the corresponding setting to our agents in order to form a new interaction, but within the same scenario and with a goal that was naturally desirable and achievable within that setting.

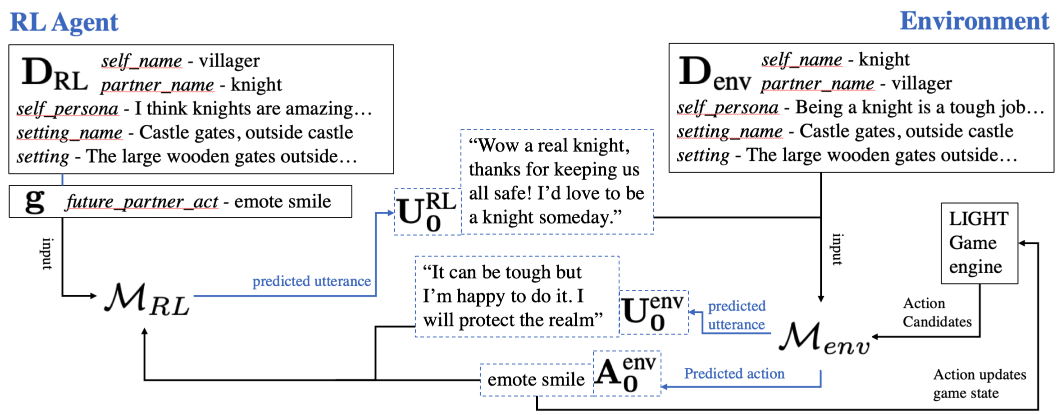

Figure 2: Example interaction in the described task setup (single turn). Here the RL agent $\mathcal{M}_{RL}$ would receive a reward as the environment agent $\mathcal{M}_{env}$ took the desired action $\mathbf{g}$.

Observations The state observation $\mathcal{O}_t = (\mathbf{D}_{\text{RL}}, \mathbf{S}_{t-1}, \mathbf{g})$ at time t given to an RL model consists of the RL agent's setting description ($\mathbf{D}_{\text{RL}}$), the utterance and action history up to that time step ($\mathbf{S}_{t-1}$), and the agent's goal ($\mathbf{g}$). Our RL agent models consume $\mathcal{O}_t$ as a flattened sequence of tokens, and return a dialogue utterance $\mathbf{U}_{\mathbf{t}}^{\text{RL}}$. Each structured component is represented in the flattened sequenced separated by a special token denoting the types, e.g. names, settings, etc., see Fig. 2. Note that because the entire history and goal is given to the RL agent, the environment is Markovian.

Reward We have a terminal reward of +1 only if the goal g is achieved and 0 otherwise, i.e, it is +1 if the environment agent takes the goal action g. The episode ends after n steps. In our experiments we consider $n = 1$ and $n = 3$.

4 Models

In this section we describe the models for $\mathcal{M}_{env}$ and $\mathcal{M}_{RL}$. In this work these are retrieval models, using the LIGHT dialogue corpus as candidates. We leave generative models to future work.

Base Agent Architecture For all our models we adopt the same base architecture, which is a 12-layer bidirectional transformer (Vaswani et al., 2017) pre-trained on a large dialogue corpus (Reddit, 174M examples), and then fine-tuned on our task [1]. To score retrieval candidates, we use a *bi-encoder* as in (Humeau et al., 2019; Urbanek et al., 2019). That is, two transformers are used, one to encode the context, and another to encoder a candidate dialogue, and a dot product between the first output vector of each scores the match. To produce a dialogue utterance one then takes the utterance with the largest output from the training set candidates (111k in this case). For emotes and actions, the same procedure is used, but with those candidate sets instead. For actions, the candidates are the set of admissible actions at that game state, which are provided by the game engine, for example *get apple* is only available in the candidate set if it is a valid action (an apple is present in the room). For emotes, all 22 candidates are always available. To train the model, a cross entropy loss is used. Similar to Mazaré et al. (2018), during training we consider the other elements of the batch as negatives.

Environment agent The environment agent is the base agent described above, and stays fixed during episodes where the RL agent is trained. This helps guarantee that our RL models stick to using the semantics of natural language (English) rather than so-called language drift, of learning a new emergent language with the same tokens (Lee et al., 2019).

RL agents We design two RL approaches for our tasks - learn to pick the right latent discrete variables (topics) that lead to the correct $\mathbf{U}_{\mathbf{i}}^{\text{RL}}$; and learn to pick the correct $\mathbf{U}_{\mathbf{i}}^{\text{RL}}$ from the top K

[1]This arrangement outperforms BERT-based pre-training in dialogue settings, see (Humeau et al., 2019)

candidates. These are described in more detail in Sections 4.2 and 4.3. We also discuss a baseline "inverse" model trained via behavioral cloning on the human-human data.

4.1 INVERSE MODEL

We consider an inverse model, trained to imitate human actions given a goal, as both a baseline for comparing to RL models, and for producing weights form which we can fine-tune. The inverse model consists of a Bi-encoder, as described above, which takes as input an observation $\mathcal{O}_t$ similar to our RL models, and outputs an utterance. We train it by extracting from the human-human game logs training set (which does not have goals) every instance where a game action occurs at time t in $\mathbf{S_t}$, that is where

$$\mathbf{S_t} = [(\mathbf{U_1^{RL}}, \mathbf{A_1^{RL}}), (\mathbf{U_1^{env}}, \mathbf{A_1^{env}}), \ldots, (\mathbf{U_t^{RL}}, \mathbf{A_t^{RL}}), (\mathbf{U_t^{env}}, \mathbf{A_t^{env}})],$$

and where $\mathbf{A_t^{env}}$ is not null (but $\mathbf{A_i^{RL}}$ for $0 < i \leq t$ or $\mathbf{A_i^{env}}$ for $0 < i < t$ might be null). We then construct a training example for the inverse model with observation $(\mathbf{D_{RL}}, \mathbf{g} = \mathbf{A_t^{env}}, \mathbf{S}_{t-1})$. i.e. setting the goal $\mathbf{g}$ to be $\mathbf{A_t^{env}}$, and with the desired action to be taken by the agent as $\mathbf{U_t^{RL}}$. Here we use the subscripts "RL" and "env" just to mark the relative positions in the sequence, as all actions and utterances come from the human logs. Note also that unlike the RL agents we train, the human in the RL agent "position" can take game actions.

We can thus train this model in a supervised manner using a cross entropy loss as described before. This model does not learn a policy interactively, and hence might not learn to plan or strategize optimally for goal completion. The data distribution it is trained on is different than the data distribution seen by the RL agents. Nevertheless, it can serve as a strong baseline. Further, when training our RL agents, we initialize their weights to the weights of this model, and then fine-tune from that point.

4.2 LATENT DISCRETE VARIABLE (TOPIC) MODEL

Optimizing all the parameters of a large transformer architecture by RL is both incredibly costly in data efficiency and computing time, and is also known to have the problem of language drift (Lee et al., 2019) – that is, there is no guarantee after training with self-chat that the models will output recognizable natural language utterances. A solution to both problems is to train most of the parameters of the model with human-human language data, and then to either disentangle or only optimize some of the parameters with model self-chat (Yarats & Lewis, 2017).

Here, we propose a straight-forward model for that purpose. We assume an RL agent that consists of two components.

The first component $F_c(\mathcal{O}) = P(T_c(\mathcal{O}))$ maps from an observation to a discrete variable with C possible values. It consists of a chain of two functions: a transformer T_s that takes in the observation, and outputs a state representation $\tilde{s}$, and a policy chooser $c = P(\tilde{s}) \in (1, \ldots, C)$ which takes in the state representation and outputs the value of the discrete latent variable.

The second component $T_u(\mathcal{O}, c)$ is an additional transformer that takes as input the observation as well as the output of the first component, and outputs a dialogue utterance. That is, the entire model is the chain $u = T_u(\mathcal{O}, P(T_s(\mathcal{O})))$. We make this explicit decomposition so that we can train only part of the model with RL; note that the "action" trained via RL is choosing c, not outputting the final utterance.

Initial topics We first pre-train the transformer T_s using the inverse model described in Section 4.1, which produces a vectorial representation of a given observation. We then run K-means over the vectorial representations of all observations from the training set to provide the mapping to one of C values, which represent dialogue topics, which we use as our initial function $P(\tilde{s})$. These two functions together give us our initialization of F_c. Table 1 shows the cluster ID and the topic denoted by that cluster along with the most representative sentences (closest to the center) for that cluster for 50 topics. As we can see, the clusters learnt can be coherent about a topic. We use these 50 topics as a set of actions $\mathcal{A}$ for our RL setup.

From c to $\mathcal{A}$ Given our initial choice of F_c, we can also pre-train T_u. We simply take our initial human-human training data, and for each observation append the topic computed by F_c to it. This allows our model to be able to generate an action (utterance) conditional on both an input and a topic. We can now train a policy by RL that optimizes the topic at any given point in the episode.

Policy training We keep the pre-trained portions of the model T_u and T_s fixed and during fine-tuning only optimize P. The cluster chooser P is redefined (from the initial K-means) to be an MLP network consisting of 2 layers. A discrete action is sampled from a categorical probability distribution over the possible topics, given by $\mathbf{c_t} \sim \text{Categorical}(\mathbf{h_t^2})$, where $\mathbf{h_t^2} = \tanh(\mathbf{W_2}\tanh(\mathbf{W_1 s_t} + b_1) + b_2)$.

The state vector $\mathbf{s_t}$ also encodes the goal $\mathbf{g}$ and hence, the policy is conditioned on the goal $\mathbf{g}$ of the agent. Hence, the policy can learn strategies that will result in picking actions at each time step $\mathbf{t}$ that will help the agent to achieve its goal $\mathbf{g}$. As our RL agent can only choose topics, it cannnot redefine easily the meaning of words to cause language drift.

#C	Topic	Representative Sentences
19	animal sounds	'Meow! Purr!', 'Bah-Buk! Tasty!', 'Woof! Very!', 'Bock! Bock!'
12	find the cost	'I would love some fruit. What are your prices?', 'They are beautiful. How much do the cost?', 'It flows easily, how much are you selling it for?'
28	prayer, God	'Then your poor life is a sign from God for you to join us in the church and serve him!', 'If you say so priest. From now I will pray every night... for wealth and good food!', 'Continue to love, worship, and serve Him.'
45	ask favor	'Yes but do you mind doing me a favor?', 'Since I have helped you, could you do me a favor?', 'If I offer to solve your problem, what will... you personally do for me in return?'

Table 1: Clusters learnt over the dialogue utterances. '#C' denotes the cluster ID.

4.3 Top-K model

The Top-K model is another approach to keeping the number of trainable parameters small. It uses the inverse model to get a context embedding v_{context} from the observation, and a list of K candidate utterance embeddings $v_1, ...v_K$. These are the encodings by the inverse model of the K utterances it considers most likely given the context and goal. We then train a small (2-layer) transformer model that takes as input the set $\{v_{\text{context}}, v_1, ...v_K\}$. We use the attention above weights of v_{context} against the candidates at the last layer of the transformer as the distribution over the candidates for sampling an utterance. We use $K = 50$ in the experiments.

4.4 RL Training

We use the Advantage Actor-Critic implementation (A2C; Kostrikov, 2018) to train the policy and the value function for both the latent-variable and top-K models.

5 Related work

Chit-chat dialogue There is an increasing body of work in the domain of chit-chat, where the primary approaches being currently tried are end-to-end neural approaches. They are typically large pre-trained and then fine-tuned transformers, either generative or retrieval, where currently retrieval models work best on a number of tasks (Zhang et al., 2018; Dinan et al., 2018; Li et al., 2019). Our work shares a commonality with these approaches in that the original LIGHT dialogue data we use has no specified goals, and humans chit-chat together (and act). Thus, the conversations cover a rich number of diverse topics. In Urbanek et al. (2019) models were trained in a similar fashion to chit-chat task models, and we adopt similar architectures here, but instead adapt them to learn to pursue goals.

Goal-oriented dialogue Traditional goal-oriented dialogue has focused on narrow tasks that would typically be useful for a dialogue-based assistant, for example restaurant (Henderson et al., 2014), taxi, train, and hotel (Budzianowski et al., 2018) or trip (El Asri et al., 2017) booking. Hence, each task typically focuses on a narrow slice of natural language and world knowledge for a specialized domain. Earlier work focused on labeled state representations, slot filling mechanisms and dialogue managers (Rieser & Lemon, 2011), and more recent work has shifted to an end-to-end approach (Bordes et al., 2017), in line with chit-chat models, but still the two sets of tasks are rarely considered together, or by using the same methods.

RL for dialogue The classical goal-oriented dialogue literature studies RL extensively (Singh et al., 2000). Typically, they used RL to improve dialogue managers, which manage transitions between dialogue states (Singh et al., 2002; Pietquin et al., 2011; Rieser & Lemon, 2011; Gasic et al., 2013; Fatemi et al., 2016). Recent works have focused more on end-to-end learning. Some works have focused on self-play type mechanisms for end-to-end reinforcement learning, where the reward is derived from goal. A related approach to ours is the negotation tasks of Lewis et al. (2017); Yarats & Lewis (2017), which require two agents to swap 3 item types (hats, balls, books) where the value of the items is different for the two agents, and derives their personal reward. In contrast, our setup encompasses a rich world of settings and characters – with 3462 object types, and a corresponding large number of actions. This is reflected in the vocabulary size itself ($\sim$32,000 versus $\sim$2,000 in the negotation tasks). Other notable uses of RL in dialogue include within visual question answering (Das et al., 2017), in the domain of chit-chat where RL has been used to decrease repetitive and generic responses through the the use of self-play (Li et al., 2016b), and through human-bot conversation (Sankar & Ravi, 2019).

RL for language and games RL is used extensively for learning to play games, one of the most well known examples being AlphaGo (Silver et al., 2016). Since then, language in games has started to be more deeply explored, for example in graphical games such as Minecraft (Oh et al., 2017), Real-time strategy war games (Hu et al., 2019), or in text adventure games (Narasimhan et al., 2015; Côté et al., 2018). The latter are related to our setting. However, those approaches use RL to optimize the set of actions given feedback in a *single-player* rather than multi-player game, so the text only refers to the environment, and there is no dialogue or actions from other agents. Our work focuses specifically on the latter.

6 EXPERIMENTS

We compare our various models on the game action and emote action tasks. We experiment with differing number of steps n allowed to complete the goal, $n = 1$ and $n = 3$. Our main results for both seen and unseen test environments are given in Table 2. We report the average reward and for $n = 3$ the average number of turns before completion. The results show clear improvements for our topic RL (§4.2) and top-K RL (§4.3) compared to the inverse model baseline for all values of n, and both types of actions (game actions and emotes).

We show the training curves for topic RL in Fig. 3, reporting rewards averaged over the batch (512 for $n = 1$, and 128 for $n = 3$). They show relatively smooth improvements over time, with clear gains over the baseline. As a sanity check we also tried, after training, to replace the topic RL policy with random topic prediction, which yielded poor results, e.g. 0.217 reward for $n = 1$ test seen game actions. Our model is clearly learning appropriate topic acts. We show examples of successful utterances, achieving goal actions in Fig. 3 for a diverse range of scenarios, actions and language.

		Test Seen			**Test Unseen**		
		($n = 1$)	($n = 3$)		($n = 1$)	($n = 3$)	
Model	**Goal Type**	Reward	Reward	Turns	Reward	Reward	Turns
Inverse model	game act	0.223	0.414	2.42	0.193	0.410	2.48
Topic RL	game act	0.314	0.502	2.25	0.268	0.467	2.33
Top-K RL	game act	0.319	0.460	2.37	0.270	0.450	2.39
Inverse model	emote	0.089	0.262	2.72	0.088	0.266	2.74
Topic RL	emote	0.170	0.342	2.63	0.146	0.326	2.64

Table 2: Results on the test seen and unseen environments for our models.

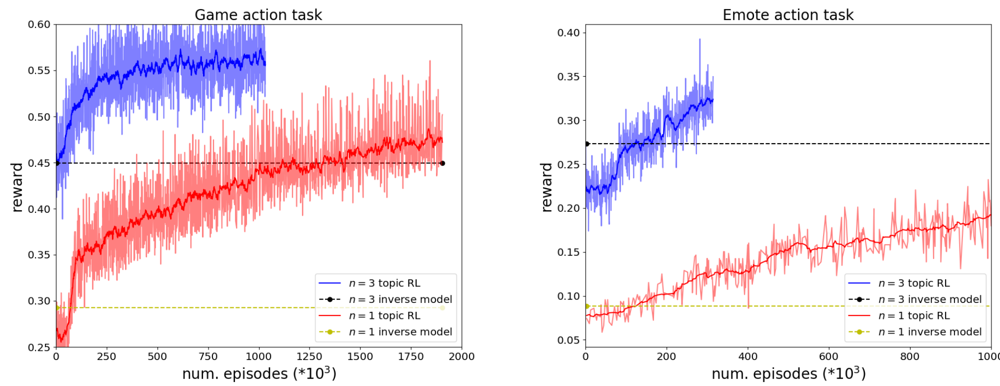

Figure 3: Topic RL model training for $n = 1$ and $n = 3$ step action goals for game actions (left) and emotes (right), comparing to the inverse model baseline. Darker lines indicate smoothed plots.

Self: guard	**Partner:** archer	**Self:** swimmer	**Partner:** turtles
Persona:	I guard the castle. I guard the king. I would kill to protect the royal family	**Persona:**	I am a huge fan of deep sea exploration, but I take any chance I can get to go for a swim...
Setting:	The armory, Inside Tower. The near top of the tower 6 feet before the very top. Where the watchers keep their eye...	**Setting:**	Bank, Swamp This is a grassy area that surrounds much of the swamp. It's a plain field with some trees nearby along...
$\mathbf{U}_0^{\mathrm{RL}}$	This is the armory! The king keeps the best weapons here. Take a look -	$\mathbf{U}_0^{\mathrm{RL}}$	Just keep taking good care of your beautiful little turtle family! Your species is quite unique and I love to see you about when I go for a swim.
$\mathbf{U}_0^{\mathrm{env}}$	Hello, I need to get into the palace to see the king. I think he might like to see these weapons.	$\mathbf{U}_0^{\mathrm{env}}$	Well, thank you for that. Do you happen to know where my other turtle friend is? You haven't captured any turtles have you?
$\mathbf{A}_0^{\mathrm{env}}$	get weapon	$\mathbf{A}_0^{\mathrm{env}}$	hug swimmer
Self: townsperson	**Partner:** villager	**Self:** songbird	**Partner:** wasp
Persona:	We are the people who live in this town. We are common, and there are many...	**Persona:**	I fly high and bring beautiful music to the people. I soar high and low going where the ...
Setting:	The Lagoon, Lake The Lagoon is a dark and mysterious place during the night hours. A lot of moss and lily...	**Setting:**	Meadow, Countryside Large clear outdoor meadow. Flowers of blue and white appearing in bunches here and there. The ...
$\mathbf{U}_0^{\mathrm{RL}}$	It is cold up here. Would you like my coat	$\mathbf{U}_0^{\mathrm{RL}}$	Get out of here, wasp!
$\mathbf{U}_0^{\mathrm{env}}$	Oh yes please if I may. My shoe has become sodden from running to the market I should love to dry it a bit.	$\mathbf{U}_0^{\mathrm{env}}$	You? Fly away from me? You're in my forest, bird. I control this land earlier.
$\mathbf{A}_0^{\mathrm{env}}$	remove Cloak	$\mathbf{A}_0^{\mathrm{env}}$	hit a songbird

Table 3: Example 1-step episodes where after the Latent Discrete Variable (Topic) RL agent's utterance $\mathbf{U}_0^{\mathrm{RL}}$ the environment agent's response action $\mathbf{A}_0^{\mathrm{env}}$ was equal to the RL agent's goal $\mathbf{g}$. Our RL agent both makes natural utterances given the situation, and that elicit the desired goal.

7 Conclusion

In this paper, we investigate agents that can interact (speak or act) and can achieve goals in a rich world with diverse language, bridging the gap between chit-chat and goal-oriented dialogue. We achieve this by defining a task for an agent, where the goal is for the other player to execute a particular action. We explore two reinforcement learning based approaches to solve this task: the policy either learns to pick a topic or learns to pick an utterance given the top K utterances, and compare them against a strong baseline trained to imitate chit-chat. We show that these approaches effectively learn dialogue strategies that lead to successful completion of goals, while producing natural chat. Future work should explore further RL algorithms for agents that can act and speak in natural language at scale in our proposed rich task environment, and we expect further advancements.

REFERENCES

Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. *arXiv preprint arXiv:1806.11532*, 2018.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2951–2960, 2017.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint arXiv:1811.01241*, 2018.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 207–219, Saarbrücken, Germany, August 2017. Association for Computational Linguistics.

Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. Policy networks with two-stage training for dialogue systems. *arXiv preprint arXiv:1606.03152*, 2016.

Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. Pomdp-based dialogue manager adaptation to extended domains. In *Proceedings of the SIGDIAL 2013 Conference*, pp. 214–222, 2013.

Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pp. 263–272, 2014.

Hengyuan Hu, Denis Yarats, Qucheng Gong, Yuandong Tian, and Mike Lewis. Hierarchical decision making by generating and following natural language instructions. *arXiv preprint arXiv:1906.00744*, 2019.

Samuel Humeau, Kurt Shuster, Marie-Anne Lachaux, and Jason Weston. Real-time inference in multi-sentence tasks with deep pretrained transformers. *arXiv preprint arXiv:1905.01969*, 2019.

Ilya Kostrikov. Pytorch implementations of reinforcement learning algorithms. `https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail`, 2018.

Jason Lee, Kyunghyun Cho, and Douwe Kiela. Countering language drift via visual grounding. *arXiv preprint arXiv:1909.04499*, 2019.

Mike Lewis, Denis Yarats, Yann N Dauphin, Devi Parikh, and Dhruv Batra. Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*, 2017.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016a.

Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016b.

Margaret Li, Jason Weston, and Stephen Roller. Acute-eval: Improved dialogue evaluation with optimized questions and multi-turn comparisons. *arXiv preprint arXiv:1909.03087*, 2019.

Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. Training millions of personalized dialogue agents. *arXiv preprint arXiv:1809.01984*, 2018.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. *arXiv preprint arXiv:1506.08941*, 2015.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2661–2670. JMLR. org, 2017.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):7, 2011.

Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*, 2005.

Verena Rieser and Oliver Lemon. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media, 2011.

Chinnadhurai Sankar and Sujith Ravi. Deep reinforcement learning for modeling chit-chat dialog with discrete attributes. *arXiv preprint arXiv:1907.02848*, 2019.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

Satinder P Singh, Michael J Kearns, Diane J Litman, and Marilyn A Walker. Reinforcement learning for spoken dialogue systems. In *Advances in Neural Information Processing Systems*, pp. 956–962, 2000.

Jack Urbanek, Angela Fan, Siddharth Karamcheti, Saachi Jain, Samuel Humeau, Emily Dinan, Tim Rocktäschel, Douwe Kiela, Arthur Szlam, and Jason Weston. Learning to speak and act in a fantasy text adventure game. *arXiv preprint arXiv:1903.03094*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Learning semantic textual similarity from conversations. *arXiv preprint arXiv:1804.07754*, 2018.

Denis Yarats and Mike Lewis. Hierarchical text generation and planning for strategic dialogue. *arXiv preprint arXiv:1712.05846*, 2017.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.

A GAME ACTIONS WITHIN LIGHT

Action	Constraints	Outcome
get *object*	actor and *object* in same room *object* is gettable	actor is carrying *object*
drop *object*	actor is carrying *object* *object* is gettable	*object* is in room
get *object1* from *object2*	Actor and *object2* in same room *object1* is gettable *object2* is surface or container *object2* is carrying *object1*	actor is carrying *object1*
put *object1* in/on *object2*	Actor and *object2* in same room *object2* is container or surface actor is carrying *object1*	*object2* is carrying *object1*
give *object* to *agent*	Actor and *agent* in same room *object* is a member of actor	*agent* is carrying *object*
steal *object* from *agent*	actor and *agent* in same room *object* is a member of *agent*	actor is carrying *object*
hit *agent*	Actor and *agent* in same room	inform *agent* of attack
hug *agent*	Actor and *agent* in same room	inform *agent* of hug
drink *object*	actor is carrying *object* *object* is a drink	inform actor of drinking successfully
eat *object*	actor is carrying *object* *object* is a food	inform actor of eating successfully
wear *object*	actor is carrying *object* *object* is wearable	actor is wearing *object*
wield *object*	actor is carrying *object* *object* is a weapon	actor is wielding *object*
remove *object*	actor is wearing/wielding *object* *object* is wearable or a weapon	actor is carrying *object*

Table 4: LIGHT actions and constraints from Urbanek et al. (2019)

B GAME EMOTES WITHIN LIGHT

applaud, blush, cry, dance, frown, gasp, grin, groan, growl, laugh, nod, nudge, ponder, pout, scream, shrug, sigh, smile, stare, wave, wink, yawn

Figure 4: Emote actions within the LIGHT platform from Urbanek et al. (2019)