Learning to Generalize for Sequential Decision Making

Xusen Yin, Ralph Weischedel, and Jonathan May
Information Sciences Institute
University of Southern California
{xusenyin,weisched,jonmay}@isi.edu

Abstract

We consider problems of making sequences of decisions to accomplish tasks, interacting via the medium of language. These problems are often tackled with reinforcement learning approaches. We find that these models do not generalize well when applied to novel task domains. However, the large amount of computation necessary to adequately train and explore the search space of sequential decision making, under a reinforcement learning paradigm, precludes the inclusion of large contextualized language models, which might otherwise enable the desired generalization ability. We introduce a teacher-student imitation learning methodology and a means of converting a reinforcement learning model into a natural language understanding model. Together, these methodologies enable the introduction of contextualized language models into the sequential decision making problem space. We show that models can learn faster and generalize more, leveraging both the imitation learning and the reformulation. Our models exceed teacher performance on various held-out decision problems, by up to 7% on in-domain problems and 24% on out-of-domain problems.

1 Introduction

We make many decisions as we interact with the world. When we are rewarded (respectively, punished), we learn to modify not only the proximal cause of the stimulus but the chain of decisions leading up to it, to encourage (respectively, discourage) future similar results. This process naturally is the paradigm of *Reinforcement Learning* (RL). Policy-based learning seeks to find good estimates for $Q(s, a)$, a function that returns the expected cumulative reward (known as a *Q-value*) if action a is chosen at state s. A desirable property of methodologies to learn Q is their ability to *generalize* such

that an appropriate action can be taken when encountering a previously unseen state.

Recent advances have shown strong evidence of generalization in spatiotemporal modalities such as robotic manipulation (Xu et al., 2018), video games (Tessler et al., 2017), and autonomous navigation (Zhu et al., 2017). However, in the modality of language, there is less work applying generalization approaches to decision making. Useful applications of sequential decision making language models are personal assistants that proactively anticipate client needs; anti-phishing mediation agents that waste a would-be thief's time with relevant but non-helpful responses; and investigative journalist assistants that determine what to read, whom to contact, and what questions to ask to create a revelatory news report.

Neural reinforcement learning (RL) training approaches, such as those used to play action video games (Mnih et al., 2013), have potential applicability in language-based decision making due to their ability to learn to navigate adversarial or exploratory scenarios. Naturally, the generalization and background knowledge capability afforded by large contextualized language models such as *BERT* (Devlin et al., 2019) may be applicable as well. A useful virtual world proxy in which to explore these approaches' applicability is that of *text adventure game playing*. In a text adventure game, a player is immersed in an environment by reading textual descriptions of a scene and issuing natural language commands to navigate inside the scene. The player discovers and interacts with entities and accomplishes goals, while receiving explicit rewards for doing so.

Learning to play text games is a useful pursuit because it is a *convenient* proxy for the real world cases cited above. Unlike these, plentiful data for numerous games exist, an endless supply of games can be constructed, and text games have built-in re-

ward functions, making them suitable for RL. This class of problems is also useful because it is *challenging*: after exposure to a *family* of games that explore the same topic and have similar gameplay (e.g., games involving cooking a specified recipe), human players perform nearly perfectly on additional games, but computer models struggle.

Why is this? Humans quickly understand the situation they are placed in and can make rational decisions based on trial-and-error and life experience, which we can call *commonsense knowledge*. Knowing *a priori* that, e.g., a `closed` door should be `open` or that it is helpful to `light a lamp` in a `dark dungeon` allows (human) players to learn faster. Even though these games have the complexity of finite-state machines, computer models cannot learn to play them well. The problem appears to be due to a lack of generalization caused by a lack of commonsense. To a computer model, considering whether to `fry` using a `fridge` is no more ludicrous than considering whether to `fry` using a `plate` (which, to an untrained human cook, may be plausible, though is certainly not a good idea). Both actions can be discouraged by negative reinforcement, but a human only needs to learn not to do the latter. Furthermore, a computer player learning that one can `chop carrots` with a `knife` may not generalize that one can `chop celery` the same way, but a human surely will.

There is existing work in learning to play text games with RL (Narasimhan et al., 2015; Yuan et al., 2018; Kim, 2014; Zahavy et al., 2018; Yin and May, 2019a; Tessler et al., 2019) but the standard pattern of incorporating large language models such as *BERT* (Devlin et al., 2019) has not yet been seen in current literature. It turns out that this integration is not trivial. Most models that use *BERT* and its ilk predominantly apply their results to *supervised learning* tasks that have training data with ground truth (Zellers et al., 2018; Wang et al., 2018) or at least, in the case of generation-based tasks like dialogue and translation, a corpus of desirable output to mimic (Wolf et al., 2019; Imamura and Sumita, 2019). For tasks suited to RL such as the exploration of and interaction with a world, there is no true target or even, initially, a corpus, and thus learning can only proceed iteratively via, e.g., exploration-exploitation (Mnih et al., 2013), which requires millions of training iterations to converge (Yin and May, 2019a; Narasimhan et al., 2017; Mnih et al., 2013). Integrating this process
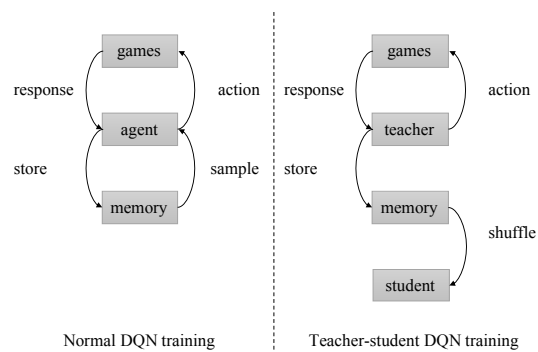

Figure 1: Comparison of the training process between DQN (left) and teacher-student DQN (right). In regular DQN training, the agent plays games to collect partial game-playing into memory, for later sampling and training. In the teacher-student training method, the teacher—a well-trained agent—generates a partial play curriculum. Student agents use that curriculum only to train their models.

with the additional overhead of fine-tuning a large model like *BERT* leads to an impractical slowdown: for the experiments considered in this work, the baseline models that use *CNN* require a little more than three weeks to train on an Nvidia P100 GPU-equipped machine. Using the same models on the same tasks run for the same number of iterations on the same hardware while fine-tuning a 12-layer *BERT* model would take more than two *years*.

In this work, we compare different previously used representation models for deep RL through an imitation learning method that first trains a light-weight teacher using exploration-exploitation, and then uses that trained model to train a more heavy-weight student model. This dramatically decreases the amount of training time needed to learn. Moreover, we devise a means of casting an RL problem into a supervised learning paradigm, allowing better exploitation of large contextualized language models. In so doing, we show that agents can benefit from both the imitation learning and the reformulation, converging faster than other models, and exceeding teacher performance by 7% and 24% on both in- and out-of-domain problems, despite the limited search space.

The novel contributions of this work are:

- We develop a teacher-student model training method for sequential text-based decision making problems, enabling the efficient incorporation of heavy-weight external information models.
- We develop a method for casting student RL

model training in the same form as a supervised Natural Language Understanding task, enabling solutions to those tasks to be applied to sequential decision making.

- We evaluate our methods on in-domain and out-of-domain text game data sets, extrinsically and intrinsically demonstrating the effect of external commonsense knowledge and generalization at improving model abilities.
- We release our data, models, and code for documentation, replicability, and to enable subsequent improvements.[1]

2 Background: Reinforcement Learning for Text Games

The core approach of Deep Q-Networks (DQN) as described by Mnih et al. (2015) is to build a *replay memory* of partial games with associated scores, and use this to learn a function $f_{DQN} : (S, A) \rightarrow \mathcal{R}$, where $f_{DQN}(s, a)$ is the Q-value obtained by choosing action $a \in A$ when in state $s \in S$; from s, choosing $\arg\max_{\tilde{a} \in A} f_{DQN}(s, \tilde{a})$ affords the optimal action policy and this is generally used at inference time.[2]

The DQN, which predicts Q given a (state, action) pair, is trained in an exploration-exploitation method known as ϵ-*greedy* (Mnih et al., 2015): first, the agent plays the game stochastically, generally guided by f_{DQN}, but with probability ϵ choosing a random action instead. The hyperparameter ϵ usually decays from 1 to 0 during the training process. As the agent plays, it collects partial play samples (s, a, r, s'), denoting taking action a at the game state s, and the immediate reward r plus the next state s' reached for doing so, into a replay memory. The DQN is then improved by sampling from the replay memory, and reducing loss between f_{DQN} and the true Q, which is estimated as

$$Q(s, a) = r + \lambda \max_{a'} f_{DQN}(s', a'). \quad (1)$$

Square error loss is minimized to improve f_{DQN} along the gradient:

$$l_{dqn} = \|f_{DQN}(s, a) - Q(s, a)\|^2. \quad (2)$$

The improved DQN is used to collect more replay data as the process iterates, as depicted in Figure 1 (left).

Equation 1 shows that at every step, for every sampled state, we can only estimate loss for a single state-action pair; we do not have the r or s' for actions other than a. The models eventually converge, but only after millions of training steps (Mnih et al., 2013; Yin and May, 2019a).

3 Teacher-Student DQN Training

After running DQN training as described in Section 2 for some time, our well-trained agent, which we call the *teacher*, can provide Q-values for a set of actions A at every step. We can then collect lots of (state s, action set A, Q-table Q) game-play tuple data into a *curriculum pool* by repeatedly playing the game and obtaining $f_{DQN-\text{teacher}}(s, a)$ for all $a \in A$. We now use that data to train a new agent $f_{DQN-\text{student}}$ (the *student*), using the same DQN approach described in Section 2. However, unlike in the previous DQN scenario, the curriculum pool now contains Q-values for all of A at each state.[3] We can train *all actions at one step* for each trajectory since we have Q-values for all actions. Thus the loss is

$$l_{se} = \sum_{a \in A} \|f_{DQN}(s, a) - Q(s, a)\|^2 / \|A\|, \quad (3)$$

and the learning procedure is as depicted on the right side of Figure 1.

The main disadvantage of teacher-student learning is that in the student phase, the search space is bounded by that of the curriculum pool generated by the teacher agent. While a student model can generalize based on the curriculum pool's data, it cannot explore any more of the search space. On the other hand, student learning is much faster than teacher learning. The experience replay pool does not need to be repeatedly generated, and many more loss gradients can be calculated all at once. We will explore several architectures and configurations that take advantage of this speed.

3.1 State Representations

A fundamental parameter that must be specified is the input signal used to form the game state s and how it is encoded. For action video games,

---

[1] https://github.com/yinxusen/learning_to_generalize

[2] There are various methods to choose actions according to policies, but for exposition, the greedy method is the most representative and straightforward one. See Section 4.2 for details.

[3] It would not be viable or helpful to collect Q-values exhaustively during the ϵ-greedy phase because of the poor initial estimates of f_{DQN}.

this generally consists of a sequence of images from the game display. We use a history of system description-player action sequences for text games, which we call a *trajectory*. We consider the following representation architectures for the trajectory, some of which are only possible to use in the significantly faster student learning scenario:

CNN. While much work applied to text games uses LSTMs (Hochreiter and Schmidhuber, 1997) to represent the trajectory (Narasimhan et al., 2015; Ammanabrolu and Riedl, 2019; Yuan et al., 2018; Kostka et al., 2017; Ansari et al., 2018), we favor *CNN* encoders with position embeddings, which are faster to train than LSTMs (Zahavy et al., 2018; Yin and May, 2019a; Kim, 2014). This encoder is the only representation that is fast enough for training the teacher model, given the fact that the trajectory length is usually much longer than a single sentence or paragraph. We also experiment with it as a student model trajectory representation. This baseline *CNN* encoder uses randomly initialized word embeddings that are fine-tuned during training. This encoder has one layer, with 32 convolutional filters for each of size 3–5 (Kim, 2014).

CNN-GloVe. The *CNN-GloVe* encoder is identical to the *CNN* encoder except for the use of GloVe (Pennington et al., 2014) for word embeddings; these are not fine-tuned.

Transformer. We use the *Transformer* (Vaswani et al., 2017) architecture configured in the same way as the *BERT*-base uncased model with 12 layers (Devlin et al., 2019), but with all weights randomly initialized. This model will serve as a comparison with the following model.

BERT. We use the *BERT*-base uncased model with 12 layers. This model has the same architecture as *Transformer* but is initialized with *BERT* weights (Devlin et al., 2019).

We use a max-pooling layer over the output of *CNN* as the encoded state in the same way that we do with *CNN-GloVe*, while for *Transformer* and *BERT*, we use the pooling output from the CLS token as the encoded state. All encoded states from different encoders are passed through a dense linear layer of 32 dimensions to ensure the encoded state is of equal size across models.

We use *BERT*'s provided Byte-Pair Encoding (Sennrich et al., 2016) sub-word tokenizer and vocabulary with 30,522 tokens for *CNN*, *Transformer*, and *BERT*. For *CNN-GloVe*, we use the GloVe 6B model with 400,000 tokens and the TreeBank

	S1: You find yourself in a kitchen. You see a counter. on the counter you can make out a knife. You have a potato and a bottle of water.

P1: take the knife from counter
S2: You take the knife, which is suitable for slicing, peeling, and chopping
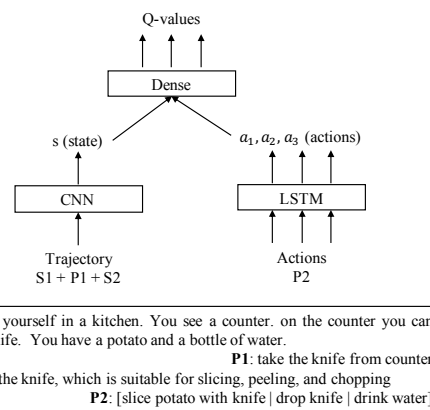**P2**: [slice potato with knife | drop knife | drink water]

Figure 2: The architecture of the DRRN model. Trajectories and actions are encoded by a *CNN* (in this case) and an LSTM into state and action representations, respectively, followed by a dense layer to compute the Q-values. On the bottom, we show a truncated example of dialogue from a text game in the cooking genre, with S1 and S2 representing the system's descriptions, and P1 showing the player's first actions in response to S1. S1 + P1 + S2 is an example of a trajectory. P2 shows a set of admissible actions.

word tokenizer from NLTK (Loper and Bird, 2002) since GloVe embeddings are pre-determined and not compatible with BPE. We use a zero vector as the padding token and average of all word embeddings as the unknown token for *CNN-GloVe*. *CNN* uses a word embedding size of 64, while for *CNN-GloVe* and *BERT*, we use the pre-trained word embedding size, i.e., 50 dimensions for *CNN-GloVe* (we choose this dimension because it is close to our *CNN*) and 768 for *BERT* (so does *Transformer*).

3.2 Action Representations

A consequence of learning to play different games is that actions differ from one game to another. Vanilla DQNs, introduced by (Mnih et al., 2015), are incompatible with this modification since they presume a predefined finite and consistent action space, such as the directions and push buttons of a joystick. Additionally, vanilla DQNs presume no semantic relatedness among action spaces. In text games, by contrast, it would make sense for, e.g., `open the door` to be semantically closer to `shut the door` than to `dice the carrot`. In our experiments, we assume the action set for a test game may be unknown at training time, and that actions may have some interrelatedness.[4] We

---

[4]This is itself still a simplification, as many text games allow *unbounded* action generation. We leave this problem for future work.

Q-value

↑

| Dense |

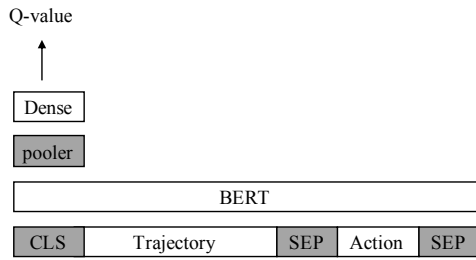| pooler |

| BERT |

| CLS | Trajectory | SEP | Action | SEP |

Figure 3: The architecture of the *BERT*-NLU models (*BERT*-NLU-SE shown). Trajectories and actions are concatenated together as the input to *BERT*, followed by a dense layer over the pooling output of the CLS token to compute the Q-value. Unlike the DRRN model, we need to concatenate a trajectory with an action to compute a Q-value, which increases computation time.

thus represent actions using Deep Reinforcement Relevance Networks (DRRN) (He et al., 2016), a modification of the standard DQN, as shown in Figure 2. Actions are encoded via an LSTM (Hochreiter and Schmidhuber, 1997) and scored against state representations by inner products and an extra dense layer. In preliminary experiments, we found that LSTMs worked better than *CNN* on the small and similar actions in our space, such as *take yellow potato from fridge* and *dice purple potato*. We limit these actions to a maximum length of 10 tokens. We use DRRNs in both teacher and student scenarios.

3.3 Game-Playing as Sequential Natural Language Understanding Tasks

Large contextualized language models built on the *Transformer* architecture such as *BERT* and GPT (Radford et al., 2018) have been used in supervised multiple-choice Natural Language Understanding (NLU) tasks. While we have so far encoded trajectories and actions separately in the DRRN formulation of DQN (Section 3.2), NLU task architectures commonly encode context and hypothesis together, using a dense final layer to obtain scalar confidence in the hypothesis being the correct result of the context. This is then trained (with a cross-entropy loss) across all hypotheses for that context. By considering trajectories as context, actions as hypotheses, and $\arg\max_{\tilde{a} \in A} f_{DQN-\text{teacher}}(t, \tilde{a})$[5] as a label for trajectory t from the curriculum pool, we may easily switch to this framework, now minimizing a standard cross-entropy loss in place of DQN student learning. We call this model *BERT*-NLU-CE.

---

[5]Henceforth we abuse notation and replace s, the state, with t, the trajectory used to represent it

At evaluation time, the model chooses an action to take given a trajectory, but we are no longer explicitly learning a new Q-function other than simply learning a preferred choice.

Of course, having an existing Q-table from the teacher model, we may instead replace the cross-entropy loss with the familiar mean squared error loss (Section 3). This model, which we call *BERT*-NLU-SE, operates the same way as *BERT*-NLU-CE, but the values associated with each (trajectory, action) pair are once again regarded as Q-values. Figure 3 depicts the architecture of *BERT*-NLU-SE; *BERT*-NLU-CE is identical except the output is not explicitly intended to be a Q-value.

While most NLU tasks like SWAG (Zellers et al., 2018) or ARC (Clark et al., 2018) have no more than five hypotheses to choose from, even artificially constrained text-based games may have hundreds of potential choices. To make training feasible for text games, given each trajectory, we randomly sample three possible actions, along with the teacher model's most favored one. At evaluation time, the model can choose from all admissible actions.

4 Games and Evaluation Methodology

Unlike most video- or text-game-playing work (Mnih et al., 2013; Zahavy et al., 2018; Yin and May, 2019a; Narasimhan et al., 2015) which incrementally learns to play games through RL approaches and reports results on those same games, we evaluate on games that are not seen during learning. Our games are generated from the TextWorld platform (Côté et al., 2019), which procedurally generates a wide variety of game variants with different maps, objectives, actions, threats, and background text, given user-supplied inputs. The platform provides the set of *admissible* actions, i.e., legal actions available at each state of each game. There are between 10–100 of these actions depending on the context.

4.1 Training and Evaluation Data

We use the games released by Microsoft for the *First TextWorld Problems*[6] competition for our training set and an evaluation set of unseen but in-domain games. The competition provides 4,440 cooking games generated by the TextWorld framework. The goal of each game is to prepare a

---

[6]https://www.microsoft.com/en-us/research/project/textworld

recipe. The action space is simple, yet expressive, and has a moderately large, though domain-limited, vocabulary. One example presents a recipe with directions such as `fry the pork chop`, `chop the yellow bell pepper`, `fry the yellow bell pepper`, `prepare meal`. To succeed, the player must find and take the items in various rooms and containers, use the correct cooking implements, and not spoil the ingredients, e.g., by `frying` an `apple` that has already been fried. We hold out 444 of these games as an in-domain test set.

To evaluate our models' ability to generalize beyond their training domain, we also evaluate on an out-of-dojain set, comprising 208 newly generated games[7] in a treasure-hunting genre. These have quite different actions, objects, and goals from cooking games. They generally require the player to navigate around rooms, find a specific object, and take a specified action with the entity, e.g., picking up a `key` and `insert`ing into a `gate`'s lock in a different room to `unlock` it. These games have little vocabulary overlap with any cooking games apart from basic commands like `take` and `drop`.

4.2 Evaluation

We report scores on each test set as a percentage of the possible total score. Each game has 1–6 points available. At evaluation time, we play each game twice, stopping after the sooner of 100 steps, game completion, or game failure, and consider each play independently. Scores can vary because each gameplay uses an initial knowledge graph map construction built via random walks (Ammanabrolu and Riedl, 2019) and because confidence bound is learned per action (Yin and May, 2019b), such that at evaluation time, lower-confidence actions are chosen with more stochasticity. An agent taking purely random walks (a low-bar baseline) scores 14% on the in-domain test and 16% on out-of-domain.

We train the teacher agent for 10 million steps on the 3,960 training games in the cooking domain, using deep Q-learning described in Section 3.2. We use a curriculum training schema (Yin and May, 2019b) to train our teacher model. During training, each 5,000-step checkpoint takes 25 minutes on a single P100 GPU. We decay ϵ from 1 to 0

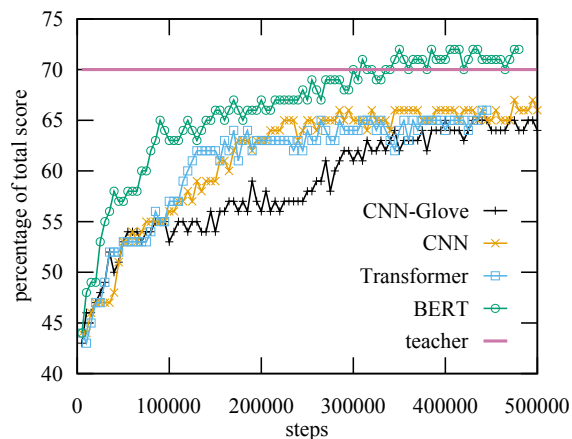[7] https://github.com/microsoft/TextWorld#usage

Figure 4: Comparing DRRN state encoders over in-domain test games. *BERT* converges faster and to a higher point (72%) than other students (*CNN* 67%; *CNN-GloVe* 65%; *Transformer*: 67%) and the teacher (70%).

during training. The teacher agent scores 70% on in-domain test and 33% on out-of-domain test.

5 Student Experiment Results

Having trained a teacher model using DQN and allowing unlimited exploration of the game space, we now experiment with several student learning approaches. Relative to the teacher model, the students are constrained to explore using data generated from the trained teacher model. This restriction limits their ability to search but enables much faster training and, consequently, richer models. All student models are trained for 500,000 steps of 32 (trajectory, action) pairs per batch, saving checkpoints every 5,000 steps and generating results for in- and out-of-domain test sets. Running on a single P100 GPU, all *Transformer*-based models take 75-80 minutes per 5000 steps, while *CNN*-based models take 13 minutes.

5.1 Data Generation from Teacher Models

We generate student curriculum pools from the trained teacher model by playing all Cooking-Train games in random order. Specifically, we play games with the teacher agent using ϵ-greedy search (Section 2). We uniformly sample $\epsilon \in [0, 1]$ among different game playing episodes to increase the variety of trajectories exhibited to student learners. We collect the trajectory, all actions, and Q-values assigned to each action by the teacher model for each game playing step. In total, we collect 10 million instances of such tuples from the 3,960 Cooking-Train games.
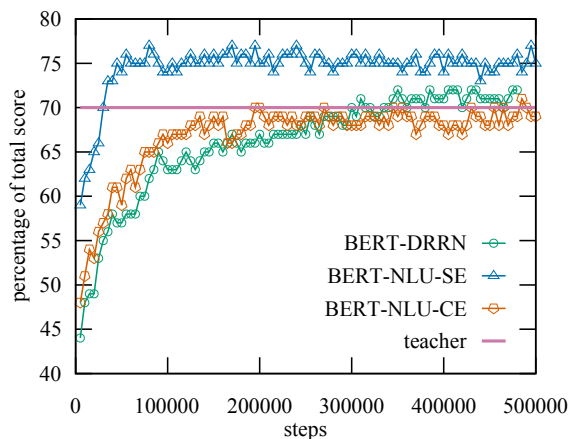
3051

Figure 5: Comparing the use of *BERT* in a DRRN (Section 3.2) or NLU (Section 3.3) student model over in-domain test games. *BERT*-NLU-SE converges 7% higher than the teacher model very rapidly, indicating quick generalization capability.

5.2 In-Domain DRRN Results

In Figure 4, we compare student model learning to the teacher model's final position (horizontal line). We see that for many of the models, the trade-off to a potentially more sophisticated architecture is not worth the damage caused by limited exploration. As expected, our baseline model, *CNN*, which is the same model used for teacher training, converges to 67% of the total possible score at around 300,000 steps of training; the teacher model is at 70%. *CNN-GloVe*, compared to *CNN*, is even worse and converges more slowly. Even though *CNN-GloVe* is equipped with pre-trained word embeddings, the student agent cannot benefit from it.

Transformer performs comparably to *CNN*, but *BERT* learns much more quickly than all other models, reaching 72% of the total score on test games; 5% higher than any other student models and somewhat better than the teacher model, which is an encouraging preliminary result.

5.3 In-Domain NLU Results

In Figure 5 we explore the performance of the NLU-inspired architecture (Section 3.3). The cross-entropy-based approach, *BERT*-NLU-CE, is the most similar to standard supervised NLU tasks and performs comparably to the DRRN teacher-student framework. However, *BERT*-NLU-SE, which directly regresses to the Q-function's value, quickly converges to around 77% of optimal scores, 7 points higher than the teacher model.

Independent of the method for learning Q-values, we can choose between multiple methods to *apply*

	GRD	UCB	Sampling		
			t=0.01	t=0.1	t=1.0
SE	72	77	72	66	27
CE	67	71	65	65	66

Table 1: Comparing different inference methods for *BERT*-NLU agents over cooking-Test (percentage of total possible score). We compare two *BERT*-NLU agents, one trained with square error loss (SE) and one with cross entropy loss (CE). We use three different methods of choosing actions from policies: greedy (GRD), LinUCB (UCB) and sampling at three different temperatures.

	SE	CE
BERT	77	71
no-init	69	60
freeze-all-but-pooler	26	25
freeze-to-penultimate	61	54

Table 2: Ablation study of the NLU method. We compare the best test-set evaluation scores (percentage of total) over the 100 epochs of training for each model trained with either square error loss (SE) or cross-entropy loss (CE).

the policy at inference time. We compare three frequently used methods—ϵ-greedy, sampling, and LinUCB, a bandit feedback method (Auer, 2003; Abe et al., 2003; Abbasi-yadkori et al., 2011)—in Table 1. Following Yin and May (2019b), we use $\epsilon = 0$ for the ϵ-greedy method. For the sampling method, we choose different temperatures over the Q-values. We follow Yin and May (2019b) for the LinUCB method. In Table 1, we ablate *BERT*-NLU-CE and *BERT*-NLU-SE training and five different inference approaches. The same Q-values are used for each setting using *BERT*-NLU-CE and for each setting using *BERT*-NLU-SE. We find that models trained with square error loss and evaluated using sampling are highly sensitive to temperature; cross-entropy-trained models are fairly insensitive. However, both the ϵ-greedy and the sampling methods perform worse than the LinUCB method.

We ablate the impact of fine-tuning *BERT* in Table 2, showing what happens if we do not fine-tune except the pooler (*freeze-all-but-pooler*), only fine-tune the last layer and the pooler (*freeze-to-penultimate*), or fine-tune all layers (*BERT*). We also show the fine-tuned equivalent *Transformer* that is not pre-trained (*no-init*) for comparison. All settings fine-tune the 768-parameter last dense layer on top of *Transformer* to compute Q-values.
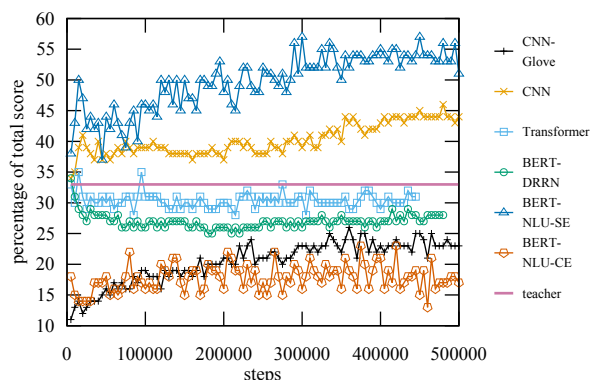
3052

Figure 6: Comparing performance on out-of-domain test games. Only two agents exceed teacher results (by 24% and 13%, respectively). Analysis in Figure 7 shows that the superior performing *BERT*-NLU-SE benefits from external knowledge, while *CNN*-DRRN mostly benefits from stochasticity.

The *freeze-to-penultimate* allows the final Q-value layer, the *pooler*, and the last layer of *BERT* to train. In total, more than seven million parameters are trainable in *freeze-to-penultimate*. However, the performance still has a 16% gap compared to the fully fine-tuned 110-million-parameter *BERT* models. This ablation study shows that the benefits coming from *BERT* can not be reproduced by simply using out-of-the-box *BERT* weights, which would speed up the training process, and underscores the importance of imitation learning.

5.4 Out-of-Domain Results

Figure 6 shows the result of evaluating with out-of-domain games. These games have different goals and action sets from the training games, so it is possible during training to observe performance curve drops, an indication that the model is overfitting on the cooking game genre and not properly generalizing. Most of the DRRN student models exhibit some overfitting; only the *CNN* model can learn somewhat well and exceeds the performance (46%) of the teacher model (33%). *BERT*-NLU-SE, the NLU-style architecture that fine-tunes *BERT* and is trained to directly estimate Q-values,greatly exceeds the teacher model's performance (57%) on these games from an unseen genre.

6 Discussion

In this section we seek to understand the following:

- What extra information *BERT*-NLU-SE leverages compared to other DRRN models (Figures 4 and 5);
- What generalization and extra information

BERT-NLU-SE leverages on out-of-domain games, and why the *CNN* student model performs better than expected on out-of-domain games (Figure 6).

A qualitative investigation of model performance on in-domain test sets shows that game failure arises when a model decides to prepare an ingredient improperly, (e.g., to use the BBQ instead of the stove to fry). Models initialized with *BERT* have fewer such failures, indicating that *BERT* provides background cooking knowledge, beyond what can be learned from the curriculum pool. Example gameplays and complete statistics on test games are provided in the Appendix.

A similar pattern is observed for out-of-domain tests. One test requires the player to use four different kinds of keys with matched locked containers. As the curriculum pool does not have any information relevant for this task, models without general background knowledge suffer. In the key/lock test game (a readout is in the Appendix), the teacher model repeatedly unlocks and locks a single box, and puts and takes the same key without making progress. The *BERT*-NLU-SE model, however, can correctly open the sequence of containers.

Figure 7 provides more insight into model performance, including an explanation for the surprising success of the *CNN* model. That figure shows the KL-divergence (Kullback and Leibler, 1951) between a uniform distribution and the distribution formed from the Q-values (the categorical choice distribution for *BERT*-NLU-CE) at every step during every out-of-domain test, computed from the final point of each model. The *CNN* model's distribution is closer to uniform than the others. As stochastic choices are made at test time when the action distribution is uncertain (see Section 4.2), the *CNN* model performs more exploration during the evaluation of Treasure hunting games. These games do not have failure cases like the in-domain test games, so there can be some benefit to stochasticity. The other models are more confident[8] and, except for *BERT*-NLU-SE, are generally wrong. This result indicates that equipped with the ability to generalize from *BERT* pre-training, *BERT*-NLU-SE has learned the skill of decision making, rather than the ability to memorize patterns.

---

[8]particularly *BERT*-NLU-CE, which is trained to make very peaked decisions
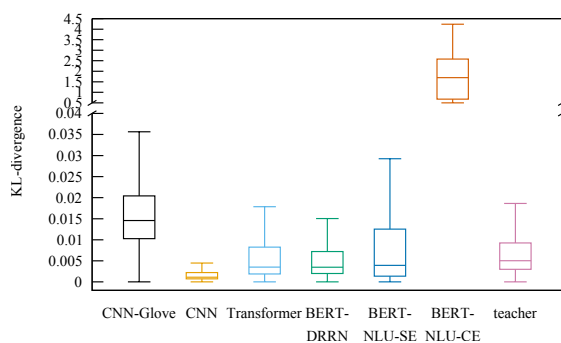
Figure 7: Boxplot of KL-divergence between Q-values and uniform distributions for agents over treasure hunting games, showing agent *confidence*. A larger KL-divergence value indicates more confidence and thus a sharper distribution of Q-values. The *CNN* student agent is the least confident agent, but this allows it to explore more; other models are more confident but are wrong. Only *BERT*-NLU-SE is both confident and correct, able to generalize well.

7 Related Work

Many recent works (Narasimhan et al., 2015; He et al., 2016; Ansari et al., 2018; Fulda et al., 2017; Côté et al., 2019; Kostka et al., 2017) on building agents to play text-based games apply DQNs (Mnih et al., 2015) or their variants. Different aspects of DQN have been presented, such as action reduction with language correlation (Fulda et al., 2017), a bounding method (Zahavy et al., 2018), the introduction of a knowledge graph (Ammanabrolu and Riedl, 2019), text understanding with dependency parsing (Yin and May, 2019a), and the bandit feedback method for agent evaluation (Yin and May, 2019b).

However, previous work uses different games to evaluate, making it difficult to compare results comprehensively. With the TextWorld framework's availability, there is more and more work concentrating on the generalization ability of agents, which seldom appears in the video game playing domain. Yuan et al. (2018) work on generalization of agents on variants of a very simple coin-collecting game. The simplicity of their games enables them to use an LSTM-DQN method with a counting-based reward. Ammanabrolu and Riedl (2019) use a knowledge graph as a persistent memory to encode states, while we use a knowledge graph later on to make actions more informative.

The TextWorld competition has yielded a variety of works that use different approaches and methods: Yuan et al. (2019) co-train a DQN with a question

answering system for building new interactive machine reading comprehension tasks while creating agents to solve games. Madotto et al. (2020) describe a non-RL method to learn agents, by first randomly playing on training games, then collecting all winning trajectories. By using these trajectories as training data, they manage to transform an RL problem into supervised learning. Adolphs and Hofmann (2020) use an actor-critic framework and prune the action-space by using hierarchical RL and a specialized module trained on a recipe database to build better agents. Jain et al. (2020) apply the action elimination method proposed by Zahavy et al. (2018) on Zork to the cooking games.

For teacher-student training, Rusu et al. (2015) design a policy distillation method that trains different agents as teacher agents. Each of these teacher agents learns to play a single and separate game. Then they build one student learner that can be trained with a supervised learning method to distill the policy knowledge for multi-game playing. Ansari et al. (2018) also use teacher-student training for text-based games. However, our teacher-student training method is different: we use one teacher that can play multiple games to guide multiple student agents' learning processes.

8 Conclusion

We provide a recipe for integrating large contextualized language models and deep reinforcement learning, applying to sequential decision making and a demonstration on the proxy task of text games, showing dramatic improvements over the standard practice, particularly in out-of-domain held-out tests. We expect to apply this approach to various challenging real-world sequential decision scenarios, such as goal-directed dialogue and active information-gathering.

Acknowledgments

References

Yasin Abbasi-yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved algorithms for linear stochastic bandits. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2312–2320. Curran Associates, Inc.

Naoki Abe, Alan W. Biermann, and Philip M. Long. 2003. Reinforcement learning with immediate rewards and linear hypotheses. *Algorithmica*, 37:263–293.

Leonard Adolphs and Thomas Hofmann. 2020. Ledeepchef deep reinforcement learning agent for families of text-based games. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7342–7349.

Prithviraj Ammanabrolu and Mark Riedl. 2019. Playing text-adventure games with graph-based deep reinforcement learning. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3557–3565, Minneapolis, Minnesota. Association for Computational Linguistics.

Ghulam Ahmed Ansari, Sagar J. P, Sarath Chandar, and Balaraman Ravindran. 2018. Language expansion in text-based games. *CoRR*, abs/1805.07274.

Peter Auer. 2003. Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.*, 3:397–422.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, and et al. 2019. TextWorld: A learning environment for text-based games. *Computer Games*, page 41–75.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. 2017. What can you do with a rock? affordance extraction via word embeddings. In *IJCAI*, pages 1039–1045. ijcai.org.

Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng, and Mari Ostendorf. 2016. Deep reinforcement learning with a natural language action space. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1621–1630. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Kenji Imamura and Eiichiro Sumita. 2019. Recycling a pre-trained BERT encoder for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 23–31, Hong Kong. Association for Computational Linguistics.

Vishal Jain, William Fedus, Hugo Larochelle, Doina Precup, and Marc G. Bellemare. 2020. Algorithmic improvements for deep reinforcement learning applied to interactive fiction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):4328–4336.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751. Association for Computational Linguistics.

Bartosz Kostka, Jaroslaw Kwiecieli, Jakub Kowalski, and Pawel Rychlikowski. 2017. Text-based adventures of the golovin AI agent. In *CIG*, pages 181–188. IEEE.

S. Kullback and R. A. Leibler. 1951. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86.

Edward Loper and Steven Bird. 2002. NLTK: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, page 63–70, USA. Association for Computational Linguistics.

Andrea Madotto, Mahdi Namazifar, Joost Huizinga, Piero Molino, Adrien Ecoffet, Huaixiu Zheng, Alexandros Papangelis, Dian Yu, Chandra Khatri, and Gokhan Tur. 2020. Exploration based language learning for text-based games. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen

King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –.

Karthik Narasimhan, Regina Barzilay, and Tommi S. Jaakkola. 2017. Deep transfer in reinforcement learning by language grounding. *CoRR*, abs/1708.00133.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Technical Report.

Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *AAAI*.

Chen Tessler, Tom Zahavy, Deborah Cohen, Daniel J. Mankowitz, and Shie Mannor. 2019. Action assembly: Sparse imitation learning for text based games with combinatorial action spaces. *CoRR*, abs/1905.09700.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019. TransferTransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. 2018. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE.

Xusen Yin and Jonathan May. 2019a. Comprehensible context-driven text game playing. *2019 IEEE Conference on Games (CoG)*, pages 1–8.

Xusen Yin and Jonathan May. 2019b. Learn how to cook a new recipe in a new house: Using map familiarization, curriculum learning, and bandit feedback to learn families of text-based adventure games. *CoRR*, abs/1908.04777.

Xingdi Yuan, Marc-Alexandre Côté, Alessandro Sordoni, Romain Laroche, Remi Tachet des Combes, Matthew J. Hausknecht, and Adam Trischler. 2018. Counting to explore and generalize in text-based games. *CoRR*, abs/1806.11525.

Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Chris Pal, Yoshua Bengio, and Adam Trischler. 2019. Interactive language learning by question answering. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J. Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 3566–3577.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 93–104, Brussels, Belgium. Association for Computational Linguistics.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*.

A Appendix

This appendix contains comprehensive results of the models explored in this work on the two test sets. Table 3 shows the number of in-domain games won, lost due to incorrect handling of materials, and lost due to taking too many steps; the latter category is helpful, along with Q-table KL-divergence from a uniform distribution, in explaining the poor performance of overly deterministic *BERT*-NLU-CE, which fails very infrequently, but often gets stuck in a step loop instead. Table 4 shows the number of out-of-domain games won per model, as out-of-domain games cannot be lost other than by taking too many steps. We can subtract the numbers in this table from 416 to find how many games a model lost. The column 'wins over teacher,' for all models other than the teacher, shows the number of games the model won that the teacher model did not. As all failures here were due to taking too many steps, stochasticity helps *CNN* just as much as determinism hurts *BERT*-NLU-CE. Sections B and C contain one complete out-of-domain game, as played by *BERT*-NLU-SE and the teacher agent. Sections D and E show the same for a cooking game.

B *BERT*-NLU-SE on one Treasure Hunting Game

Who's got a virtual machine and is about to play through an life changing round of TextWorld? You do! Here is how to play! Your first objective is to pick up the keycard in the cubicle. Then, insert the keycard into the box within the cubicle's lock to unlock it. After you have unlocked the box, ensure that the box in the cubicle is open. Having pulled open the box, retrieve the type A latchkey from the box. After that, unlock the type A locker inside the cubicle. Then, ensure that the type A locker in the cubicle is open. After that, pick up the key from the type A locker. Then, unlock the chest. Then, make it so that the chest inside the cubicle is open. After that, retrieve the passkey from the chest in the cubicle. Got that? Good!

-= Cubicle =- I never took you for the sort of person who would show up in a cubicle, but I guess I was wrong.

You make out a locked chest. You see a locked type A locker here. You can see a box. You make out an open cabinet. The cabinet is empty! What a waste of a day!

There is a closed door leading south. There is a closed passageway leading west.
There is a keycard and a worm on the floor.

take keycard
You pick up the keycard from the ground.

take worm
You pick up the worm from the ground.

unlock box with keycard
You unlock the box.

examine keycard
The keycard looks useful

open box
You open the box, revealing a type A latchkey and a mouse.

take mouse from box
You take the mouse from the box.

take type A latchkey from box
You take the type A latchkey from the box.

unlock type A locker with type A latchkey
You unlock the type A locker.

open type A locker
You open the type A locker, revealing a key.

take key from type A locker
You take the key from the type A locker.

drop mouse
You drop the mouse on the ground.

take mouse
You pick up the mouse from the ground.

examine cabinet
The cabinet looks strong, and impossible to crack open. You can see inside it.

insert key into cabinet
You put the key into the cabinet.

take key from cabinet
You take the key from the cabinet.

close cabinet
You close the cabinet.

open cabinet
You open the cabinet.

drop key
You drop the key on the ground.

take key
You pick up the key from the ground.

examine chest
The chest looks strong, and impossible to open. You can't see inside it because the lid's in your way.

drop keycard
You drop the keycard on the ground.

take keycard
You pick up the keycard from the ground.

	Win (↑)	Fail (↓)	too many steps (↓)
teacher	578	245	65
CNN	543	306	39
CNN-GloVe	525	317	46
Transformer	551	233	104
BERT-DRRN	596	190	102
BERT-NLU-CE	572	87	229
BERT-NLU-SE	631	150	107

Table 3: Tally of in-domain test game outcomes for the models explored in this work, indicating which games are won, which games lead to a failure (by spoiling ingredients), and which end when the step limit (100) is reached. *BERT*-NLU-CE and *BERT*-NLU-SE fail on in-domain tests the least, indicating a better intuitive understanding of how to prepare recipes. Because *BERT*-NLU-CE is so confident (Figure 7) it can get stuck in a deterministic loop; this is reflected in a larger number of 'too many steps' outcomes relative to *BERT*-NLU-SE.

	Wins	Wins over teacher
teacher	122	-
CNN	173	70
CNN-GloVe	85	24
Transformer	112	25
BERT-DRRN	102	19
BERT-NLU-CE	56	5
BERT-NLU-SE	209	106

Table 4: Tally of out-domain test game outcomes for the models explored in this work, indicating which games are won and which games are won in excess of those won by the teacher.

> **unlock chest with key**

You unlock the chest.

> **open chest**

You open the chest, revealing a passkey.

> **take passkey from chest**

You take the passkey from the chest.
Your score has just gone up by one point.
*** The End ***
You scored 1 out of a possible 1, in 26 turn(s).

C Teacher on the Treasure Hunting Game

Who's got a virtual machine and is about to play through an life changing round of TextWorld? You do! Here is how to play! Your first objective is to pick up the keycard in the cubicle. Then, insert the keycard into the box within the cubicle's lock to unlock it. After you have unlocked the box, ensure that the box in the cubicle is open. Having pulled open the box, retrieve the type A latchkey from the box. After that, unlock the type A locker inside the cubicle. Then, ensure that the type A locker in the cubicle is open. After that, pick up the key from

the type A locker. Then, unlock the chest. Then, make it so that the chest inside the cubicle is open. After that, retrieve the passkey from the chest in the cubicle. Got that? Good!

-= Cubicle =- I never took you for the sort of person who would show up in a cubicle, but I guess I was wrong.

You make out a locked chest. You see a locked type A locker here. You can see a box. You make out an open cabinet. The cabinet is empty! What a waste of a day!

There is a closed door leading south. There is a closed passageway leading west.

There is a keycard and a worm on the floor.

> **examine latchkey**

The latchkey is surprisingly heavy.

> **inventory**

You are carrying: a gummy bear a latchkey

> **drop latchkey**

You drop the latchkey on the ground.

> **examine gummy bear**

that's a normal gummy bear!

> **eat gummy bear**

You eat the gummy bear. Not bad.

> **take latchkey**

You pick up the latchkey from the ground.

> **examine worm**

The worm appears to fit in here

> **examine keycard**

The keycard looks useful

> **examine door**

it is what it is, a door It is closed.

> **examine cabinet**

The cabinet looks strong, and impossible to crack open. You can see inside it.

> **examine chest**

The chest looks strong, and impossible to open. You can't see inside it because the lid's in your way.

examine passageway

it is what it is, a passageway It is closed.

take worm

You pick up the worm from the ground.

drop worm

You drop the worm on the ground.

take keycard

You pick up the keycard from the ground.

drop keycard

You drop the keycard on the ground.

insert latchkey into cabinet

You put the latchkey into the cabinet.

take latchkey from cabinet

You take the latchkey from the cabinet.

examine box

The box looks strong, and impossible to crack open. It is closed.

look

-= Cubicle =- I never took you for the sort of person who would show up in a cubicle, but I guess I was wrong.

You make out a locked chest. You see a locked type A locker here. You can see a box. You make out an open cabinet. The cabinet is empty! What a waste of a day!

There is a closed door leading south. There is a closed passageway leading west.

There is a keycard and a worm on the floor.

close cabinet

You close the cabinet.

open cabinet

You open the cabinet.

examine type A locker

The type A locker looks strong, and impossible to bash open. It is closed.

examine latchkey

The latchkey is surprisingly heavy.

inventory

You are carrying: a latchkey

drop latchkey

You drop the latchkey on the ground.

take latchkey

You pick up the latchkey from the ground.

examine worm

The worm appears to fit in here

examine keycard

The keycard looks useful

examine door

it is what it is, a door It is closed.

examine cabinet

The cabinet looks strong, and impossible to crack open. You can see inside it.

examine chest

The chest looks strong, and impossible to open. You can't see inside it because the lid's in your way.

examine passageway

it is what it is, a passageway It is closed.

take worm

You pick up the worm from the ground.

insert worm into cabinet

You put the worm into the cabinet.

take worm from cabinet

You take the worm from the cabinet.

drop worm

You drop the worm on the ground.

take keycard

You pick up the keycard from the ground.

unlock box with keycard

You unlock the box.

lock box with keycard

You lock the box.

insert keycard into cabinet

You put the keycard into the cabinet.

take keycard from cabinet

You take the keycard from the cabinet.

drop keycard

You drop the keycard on the ground.

insert latchkey into cabinet

You put the latchkey into the cabinet.

take latchkey from cabinet

You take the latchkey from the cabinet.

examine box

The box looks strong, and impossible to crack open. It is closed.

look

-= Cubicle =- I never took you for the sort of person who would show up in a cubicle, but I guess I was wrong.

You make out a locked chest. You see a locked type A locker here. You can see a box. You make out an open cabinet. The cabinet is empty! What a waste of a day!

There is a closed door leading south. There is a closed passageway leading west.

There is a keycard and a worm on the floor.

close cabinet

You close the cabinet.

open cabinet

You open the cabinet.

examine type A locker

The type A locker looks strong, and impossible to bash open. It is closed.

examine latchkey

The latchkey is surprisingly heavy.

inventory

You are carrying: a latchkey

drop latchkey

You drop the latchkey on the ground.

take latchkey

You pick up the latchkey from the ground.

examine worm

The worm appears to fit in here

examine keycard

The keycard looks useful

examine door

it is what it is, a door It is closed.

examine cabinet

The cabinet looks strong, and impossible to crack open. You can see inside it.

examine chest

The chest looks strong, and impossible to open. You can't see inside it because the lid's in your way.

examine passageway

it is what it is, a passageway It is closed.

take worm

You pick up the worm from the ground.

insert worm into cabinet

You put the worm into the cabinet.

take worm from cabinet

You take the worm from the cabinet.

drop worm

You drop the worm on the ground.

take keycard

You pick up the keycard from the ground.

unlock box with keycard

You unlock the box.

open box

You open the box, revealing a type A latchkey and a mouse.

examine type A latchkey

The type A latchkey is light.

take type A latchkey from box

You take the type A latchkey from the box.

drop type A latchkey

You drop the type A latchkey on the ground.

take type A latchkey

You pick up the type A latchkey from the ground.

unlock type A locker with type A latchkey

You unlock the type A locker.

lock type A locker with type A latchkey

You lock the type A locker.

insert type A latchkey into cabinet

You put the type A latchkey into the cabinet.

take type A latchkey from cabinet

You take the type A latchkey from the cabinet.

take mouse from box

You take the mouse from the box.

examine mouse

The mouse would seem to be to fit in here

insert latchkey into box

(the latchkey into the box) You put the latchkey into the box.

take latchkey from box

You take the latchkey from the box.

insert type A latchkey into box

You put the type A latchkey into the box.

drop mouse

You drop the mouse on the ground.

insert keycard into box

You put the keycard into the box.

take keycard from box

You take the keycard from the box.

close box

You close the box.

take mouse

You pick up the mouse from the ground.

insert mouse into cabinet

You put the mouse into the cabinet.

take mouse from cabinet

You take the mouse from the cabinet.

lock box with keycard

You lock the box.

insert keycard into cabinet

You put the keycard into the cabinet.

take keycard from cabinet

You take the keycard from the cabinet.

examine mouse

The mouse would seem to be to fit in here

drop mouse

You drop the mouse on the ground.

take mouse

You pick up the mouse from the ground.

insert mouse into cabinet

You put the mouse into the cabinet.

take mouse from cabinet

You take the mouse from the cabinet.

drop keycard

You drop the keycard on the ground.

insert latchkey into cabinet

You put the latchkey into the cabinet.

take latchkey from cabinet

You take the latchkey from the cabinet.

examine box

The box looks strong, and impossible to crack open. It is closed.

look

-= Cubicle =- I never took you for the sort of person who would show up in a cubicle, but I guess I was wrong.

You make out a locked chest. You see a locked type A locker here. You can see a box. You make out an open cabinet. The cabinet is empty! What a waste of a day!

There is a closed door leading south. There is a closed passageway leading west.

There is a keycard and a worm on the floor.

=====
won: False
lost: False
total steps: 100

D *BERT*-NLU-SE on one Cooking Game

You are hungry! Let's cook a delicious meal. Check the cookbook in the kitchen for the recipe. Once done, enjoy your meal!

-= Pantry =- You are in a pantry. An usual one.

You can see a shelf. But the thing hasn't got anything on it. Oh! Why couldn't there just be stuff on it?

There is an open frosted-glass door leading north.

go north

-= Kitchen =- You're now in the kitchen.

Look out! It's a- oh, never mind, it's just a fridge. Suddenly, you bump your head on the ceiling, but it's not such a bad bump that it's going to prevent you from looking at objects and even things. The fridge contains some water, a diced block of cheese, a diced yellow bell pepper, a raw yellow potato, an orange bell pepper, a raw pork chop and a cilantro. You see a closed oven nearby. You can make out a table. The table is massive. But the thing hasn't got anything on it. You can make out a counter. The counter is vast. On the counter you see a cookbook. Look over there! a stove. Now why would someone leave that there? The stove is conventional. But the thing is empty.

There is an open patio door leading east. There is an open frosted-glass door leading south. You need an exit without a door? You should try going west.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying nothing.

take block of cheese from fridge

You take the block of cheese from the fridge.

Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced block of cheese

take water from fridge

You take the water from the fridge.

Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: some water a diced block of cheese

take yellow bell pepper from fridge

You take the yellow bell pepper from the fridge.

Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced yellow bell pepper some water a diced block of cheese

cook yellow bell pepper with oven

You roasted the yellow bell pepper.

Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced roasted yellow bell pepper some water a diced block of cheese

drop block of cheese

You drop the block of cheese on the ground.

take block of cheese

You pick up the block of cheese from the ground.

inventory

You are carrying: a diced block of cheese a diced roasted yellow bell pepper some water

go east

-= Backyard =- You arrive in a backyard. A typical kind of place. You start to take note of what's in the room.

You can make out a patio chair. But the thing is empty, unfortunately. Hopefully, this discovery doesn't ruin your TextWorld experience! You rest your hand against a wall, but you miss the wall and fall onto a patio table. The patio table is stylish. However, the patio table, like an empty patio table, has nothing on it. Oh, great. Here's a BBQ.

There is an open barn door leading south. There is an open patio door leading west. There is an exit to the east.

cook block of cheese with BBQ

You grilled the block of cheese.

Your score has just gone up by one point.

go west

-= Kitchen =- You're now in the kitchen.

Look out! It's a- oh, never mind, it's just a fridge. Suddenly, you bump your head on the ceiling, but it's not such a bad bump that it's going to prevent you from looking at objects and even things. The fridge contains a raw yellow potato, an orange bell pepper, a raw pork chop and a cilantro. You see a closed oven nearby. You can make out a table. The table is massive. But the thing hasn't got anything on it. You can make out a counter. The counter is vast. On the counter you see a cookbook. Look over there! a stove. Now why would someone leave that there? The stove is conventional. But the thing is empty.

There is an open patio door leading east. There is an open frosted-glass door leading south. You need an exit without a door? You should try going west.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced grilled block of cheese a diced roasted yellow bell pepper some water

prepare meal

Adding the meal to your inventory.

Your score has just gone up by one point.

eat meal

You eat the meal. Not bad.

Your score has just gone up by one point.

*** The End ***

You scored 7 out of a possible 7, in 27 turn(s).

E   Teacher on the Cooking Game

You are hungry! Let's cook a delicious meal. Check the cookbook in the kitchen for the recipe. Once done, enjoy your meal!

-= Pantry =- You are in a pantry. An usual one.

You can see a shelf. But the thing hasn't got anything on it. Oh! Why couldn't there just be stuff on it?

There is an open frosted-glass door leading north.

go north

-= Kitchen =- You're now in the kitchen.

Look out! It's a- oh, never mind, it's just a fridge. Suddenly, you bump your head on the ceiling, but

it's not such a bad bump that it's going to prevent you from looking at objects and even things. The fridge contains some water, a diced block of cheese, a diced yellow bell pepper, a raw yellow potato, an orange bell pepper, a raw pork chop and a cilantro. You see a closed oven nearby. You can make out a table. The table is massive. But the thing hasn't got anything on it. You can make out a counter. The counter is vast. On the counter you see a cookbook. Look over there! a stove. Now why would someone leave that there? The stove is conventional. But the thing is empty.

There is an open patio door leading east. There is an open frosted-glass door leading south. You need an exit without a door? You should try going west.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying nothing.

take block of cheese from fridge

You take the block of cheese from the fridge.
Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced block of cheese

take yellow bell pepper from fridge

You take the yellow bell pepper from the fridge.
Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal.

Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: a diced yellow bell pepper a diced block of cheese

take water from fridge

You take the water from the fridge.
Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: some water a diced yellow bell pepper a diced block of cheese

cook yellow bell pepper with oven

You roasted the yellow bell pepper.
Your score has just gone up by one point.

examine cookbook

You open the copy of "Cooking: A Modern Approach (3rd Ed.)" and start reading:

Recipe #1 ——— Gather all following ingredients and follow the directions to prepare this tasty meal. Ingredients: block of cheese water yellow bell pepper

Directions: dice the block of cheese grill the block of cheese dice the yellow bell pepper roast the yellow bell pepper prepare meal

inventory

You are carrying: some water a diced roasted yellow bell pepper a diced block of cheese

drop block of cheese

You drop the block of cheese on the ground.

take block of cheese

You pick up the block of cheese from the ground.

inventory

You are carrying: a diced block of cheese some water a diced roasted yellow bell pepper

cook block of cheese with stove

You fried the block of cheese.
*** You lost! ***

You scored 4 out of a possible 7, in 20 turn(s).