A Sequence Modelling Approach to Question Answering in Text-Based Games

Greg Furman [1] **Edan Toledo** [1,3] **Jonathan Shock** [2,4,5] **Jan Buys** [1]

[1] Department of Computer Science, University of Cape Town
[2] Department of Mathematics and Applied Mathematics, University of Cape Town
[3] InstaDeep [4] INRS, Montreal, Canada [5] NiTHeCS, South Africa
frmgre001@myuct.ac.za, e.toledo@instadeep.com,
jonathan.shock@uct.ac.za, jbuys@cs.uct.ac.za

Abstract

Interactive Question Answering (IQA) requires an intelligent agent to interact with a dynamic environment in order to gather information necessary to answer a question. IQA tasks have been proposed as means of training systems to develop language or visual comprehension abilities. To this end, the Question Answering with Interactive Text (QAit) task was created to produce and benchmark interactive agents capable of seeking information and answering questions in unseen environments. While prior work has exclusively focused on IQA as a reinforcement learning problem, such methods suffer from low sample efficiency and poor accuracy in zero-shot evaluation. In this paper, we propose the use of the recently proposed *Decision Transformer* architecture to provide improvements upon prior baselines. By utilising a causally masked GPT-2 Transformer for command generation and a BERT model for question answer prediction, we show that the Decision Transformer achieves performance greater than or equal to current state-of-the-art RL baselines on the QAit task in a sample efficient manner. In addition, these results are achievable by training on sub-optimal random trajectories, therefore not requiring the use of online agents to gather data.

1 Introduction

Traditional methods for question answering (QA) and machine reading comprehension (MRC) are primarily concerned with the retrieval of *declarative knowledge*, that is, explicitly stated or static descriptions of entities in text documents or within a knowledge base (KB) (Trischler et al., 2017). These models tend to answer questions primarily through basic pattern matching skills, further differentiating their abilities from those of humans. Conversely, *procedural knowledge* is the sequence of actions required to perform a task (Georgeff and Lansky, 1986). To this end, interactive question answering (IQA) has been proposed as a framework for teaching MRC systems to gather the information necessary for question answering (Yuan et al., 2019).

IQA requires an agent to interact with some dynamic environment in order to gather the required knowledge to answer a question (Gordon et al., 2018). As such, the task is well-suited to be approached as a reinforcement learning (RL) problem. Yuan et al. (2019) proposed Question Answering using interactive text (QAit) as a means of testing the knowledge gathering capabilities of an agent required to answer a question about its environment. Here an agent interacts with a partially observable text-based environment, created using Microsoft TextWorld (Côté et al., 2018), in order to gather information and answer questions about the attributes, location, and existence of objects. The QAit task thus aims to benchmark generalisation and provides an environment to train agents capable of gathering information and answering questions.

Yuan et al.'s proposed baselines (using DQN (Mnih et al., 2015), DDQN (Van Hasselt et al., 2016), and Rainbow (Hessel et al., 2018)) all suffered from low sample efficiency and relatively poor performance on all three question types (location, attribute, and existence). These shortcomings suggest that alternative architectures and methodologies are required to improve performance within the QAit setting.

Transformers (Vaswani et al., 2017) have shown success in modelling a diverse range of high-dimensional problems (Brown et al., 2020; Ramesh et al., 2021; Devlin et al., 2019). Additionally, existing language models such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-Trained) (Radford et al., 2019) have been utilised to reduce the size of datasets required for training downstream language tasks (Lee and Hsiang, 2019; Mager et al., 2020). These benefits coupled with the demon-

strated ability of Transformers to model long sequences by utilising the self-attention mechanism makes this architecture ideal for IQA. Recent work (Chen et al., 2021; Janner et al., 2021) have shown the applicability of Transformers to sequential decision making problems as an alternative solution to RL problems. These approaches frame RL trajectories as sequences of states, actions, and rewards modelled autoregressively by a Transformer. This sequence modelling approach is referred to as the *Decision Transformer* (DT) (Chen et al., 2021).

In this paper we apply the Decision Transformer to QAit, replacing the online interaction and training methodology of RL approaches with a Decision Transformer that utilises the GPT-2 (Radford et al., 2019) architecture, closely following the methodology outlined by (Chen et al., 2021). We propose an additional QA module that is a fine-tuned BERT model, with the aim of leveraging pre-trained language models to provide more accurate answers to questions. We show that by framing the QAit task as a sequence modelling problem, a Decision Transformer matches or exceeds the performance of previous RL-based benchmarks when trained on random episodic rollouts, while using significantly less data. Our main contributions are as follows:

1. We show that an offline reinforcement learning method is able to match the performance of online value-based reinforcement learning baselines in the QAit environment.

2. We show that by framing IQA as a sequence modelling problem, the performance of the QAit baselines can be matched using significantly less training data.

3. We show that the Decision Transformer architecture is able to learn policies comparable to those of online reinforcement learning methods from purely random data, illustrating the architecture's ability to find structure in inherently noisy data.

2 Background

2.1 QAit

QAit is implemented in TextWorld[1] (Côté et al., 2018), an open-source simulator for training reinforcement learning (RL) agents for decision making and language comprehension. QAit text-based environments are generated procedurally via sampling from a distribution of world settings. There are two environment map types: A fixed map contains six rooms, whereas random maps sample their number of rooms from a uniform distribution $U(2, 12)$. QAit requires an agent to answer questions about the location, existence and attributes of objects in an environment. An agent interacts with a QAit environment using text commands that consist of an action, modifier, and object triplet, e.g., "open black oven". A generated environment consists of rooms each containing randomly assigned objects and location names. The agent moves around in the environment for a pre-defined number of time steps or until the predicted command action is "wait". TextWorld responds to agent commands with a state string containing information about the room the agent is in and the objects present.

2.1.1 Question types

An agent is required to answer one of three question types:

- **Location** questions assess an agent's ability to navigate the environment to find the location of an object. For example, "Where is copper key?" could be answered with "garden" or "toolbox".

- **Existence** questions requires the agent to navigate and interact with the environment to gather knowledge and determine whether an object exists. Questions are phrased as "is there any X in the world?", where X is an entity in the vocabulary, and answers are either yes ("1") or no ("0").

- **Attribute** questions require that the agent interacts with an object to determine whether it has a particular characteristic or quality. For such question types, the level of interaction and movement required in observing a sufficient amount of information to answer a question greatly exceeds location and existence questions. Answers are also either yes or no. For example, "Is stove hot?" requires an agent to find and interact with "stove" to answer the question correctly. Comprehension of both the question and the environment are required. Entities often have arbitrary names and attributes, making memorisation impossible.

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

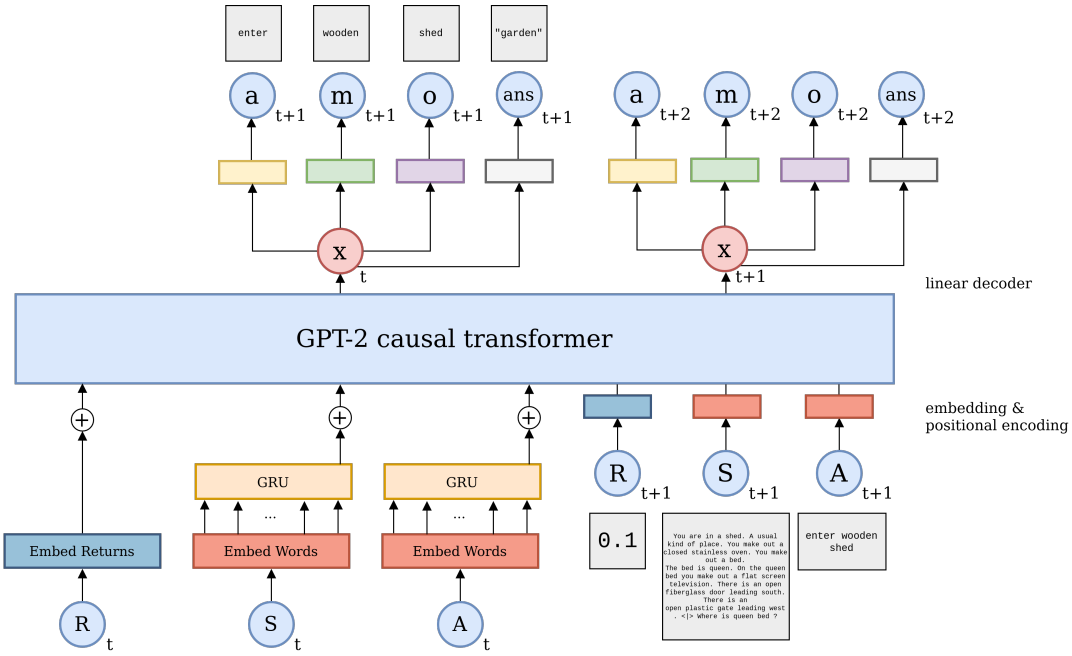

Figure 1: The Decision Transformer (Chen et al., 2021) architecture as adapted to QAit. States S_t and commands A_t have their token sequences encoded with GRUs. An embedding is also learnt for the returns-to-go R_t. Each of these three embeddings (S_t, A_t, R_t) are concatenated with a positional embedding for time step t, and fed into a GPT-2 causal Transformer. Commands are predicted autoregressively through linear decoders for the next command's action a_{t+1}, modifier m_{t+1}, and object o_{t+1} components. A fourth decoder predicts the answer to the question at each time step.

2.1.2 Rewards

Yuan et al. (2019) proposed two reward types:

Sufficient Information: Sufficient information is a metric used to evaluate the amount of information gathered by the agent and whether or not the information was sufficient to answer the question (Yuan et al., 2019). It is also used as part of the reward function. The sufficient information score is calculated when the agent decides to stop the interaction and answer the question. For each question type, the sufficient information score is calculated as follows:

- Location: A score of 1 is given if, when the agent decides to stop the interaction, the entity mentioned in the question is present in the final observation. This indicates the agent has witnessed the information it needs to answer the question successfully. If the mentioned entity is not present in the final observation then a score of 0 is given.
- Existence: If the true answer to the question is *yes* then a score of 1 is given if the entity mentioned in the question is present in the final observation. If the true answer to the question is *no*, then a score between 0 and 1 is given proportional to the amount of exploration coverage of the environment the agent has performed. Intuitively this can be seen as a confidence score - if the agent witnesses the entity, it is 100% confident of its existence; otherwise, until it explores the entire environment, it cannot be completely confident.
- Attribute: Attribute questions have a set of heuristics defined to verify each attribute and assign a score of sufficient information. Each attribute has specific commands that need to be executed for sufficient information to be gathered. This also depends on the agent being in certain states for these outcomes to be observed correctly, e.g. an agent needs to be holding an object to try to eat the object.

Exploration Reward: The agent is also given an exploration reward (Yuan et al., 2018) whenever entering a previously unseen state in order to promote exploration of the environment.

2.1.3 Evaluation

(Yuan et al., 2019) trained agents on multiple *Number of Games* settings, i.e., number of unique environments that an agent interacts with during train-

ing. In this paper, we restrict our experiments to the 500 games setting when generating offline training data for the Decision Transformer.

We measure an agent's performance through both sufficient information score and question answering accuracy. Models are evaluated in a zero-shot evaluation on the QAit test set in order to assess agents' generalisation abilities. Each question type and map type have their own unique set of 500 never-before-seen games, each containing a single question.

2.2 Decision Transformer

The Decision Transformer (Chen et al., 2021) architecture approaches reinforcement learning problems by autoregressively modelling a trajectory of actions/commands, states, and rewards. Command triples (action, modifier, noun) are conditioned upon the total reward that can still be gathered from interacting with the environment. This is referred to as the returns-to-go (RTG) $R_t = \sum_{t'=t}^{T} r_{t'}$ where T is the trajectory length and r_t is the reward at time step t. Thus the initial return-to-go R_1 represents the total reward to be gained from a given episode. After every episodic play-through, the trajectory is represented as $(R_1, s_1, a_1, R_2, s_2, a_2 ... R_T, s_T, a_T)$, where R_t is the RTG, s is a state, and a an action/command.

An example QAit trajectory is shown in Figure 2. The trajectory representation enables training a sequence model such as GPT-2 (Radford et al., 2019), as command prediction is based on gaining some future reward, rather than on how much reward has already been obtained. During testing the model is conditioned on total desired reward by setting R_1 and the starting state to generate command sequences autoregressively. If an agent obtains some reward while interacting with the world, this is deducted from its RTG in subsequent time steps.

3 Approach

Training a Decision Transformer requires offline training data for supervised learning. Online reinforcement learning, in contrast, sees an agent continually interacting with the environment to gather experience and update its policy based on observed rewards.

3.1 Training data generation with random rollouts

We generate offline training data using *random rollouts* for each map type and question type in the 500 games setting. The rollouts are generated using a random agent which uniformly samples commands from all *admissible* commands for a particular time step. This restriction stems from the sparsity of the action space (approximately 1654^3 possible commands compared to approximately 8 admissible commands): sampling commands from the complete vocabulary results in mostly invalid commands. Thus, by only using admissible commands in data generation, we intend for the Decision Transformer to learn which command triplets are admissible (as this is unknown during testing). The sequence of commands and observed states are recorded along with the reward for each command. Training dataset statistics are given in Table 1.

Map Type	Question	Mean Reward	Maximum Reward	Training Set Size
Fixed	Location	0.526	4.10	44k
	Existence	0.554	3.80	39k
	Attribute	0.498	3.73	36k
Random	Location	0.565	4.10	41k
	Existence	0.606	3.94	42k
	Attribute	0.542	4.03	82k

Table 1: Size of each of the training datasets, i.e. number of trajectories generated with random rollouts. The average and maximum total rewards gained per trajectory are also given.

3.2 Decision Transformer

The maximum trajectory input length of the Decision Transformer is set to $K = 50$ time steps, which is the maximum length of a QAit episode. This allows the DT to access the entire trajectory for command generation and question answering. Token embeddings for states and command sequences are obtained using a single embedding layer. At each time step the sequence of tokens representing the current state of the environment is encoded with a GRU (Cho et al., 2014) with the final hidden state h_n representing the entire encoded environment state. We concatenate the question to the end of each state sequence, separated by a "<|>" delimiter token. Commands, which can consist of up to 3 tokens, are similarly encoded with another GRU. Embeddings for returns-to-go R_t are also learnt and projected to the embedding dimension. Finally, a positional embedding representing the environment time step t is concatenated to each input (returns, states & commands) after the embedding and GRU layers. The embedded and positionally

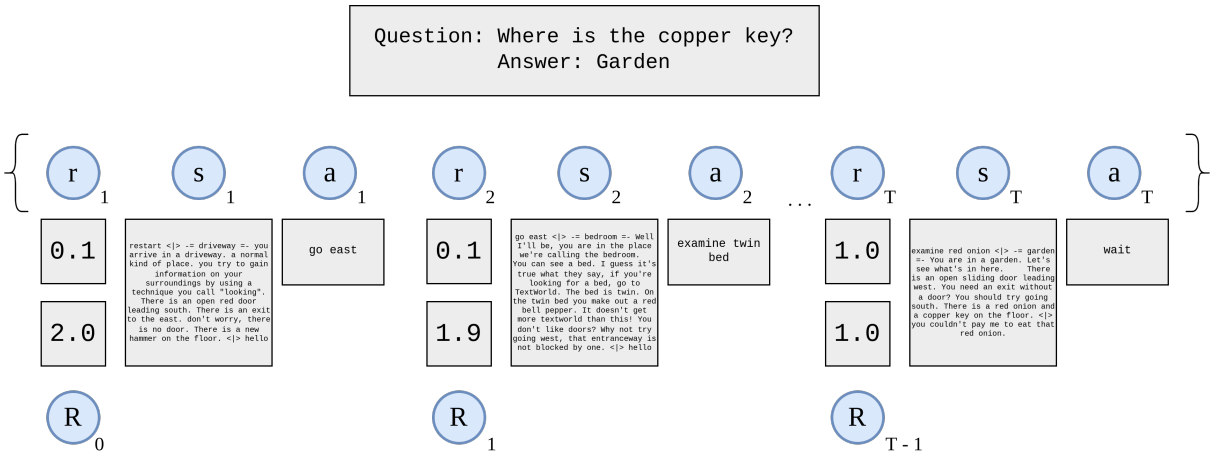

Figure 2: An example QAit trajectory in the form $(r_1, s_1, a_1, r_2, s_2, a_2, \ldots, s_T, a_T)$. Each time step consists of the reward r_t, state s_t and command a_t. The question and correct answer are also given, along with the returns-to-go R_t.

encoded command, state, and return-to-go inputs are fed into the GPT model. Figure 1 shows the Decision Transformer architecture with example input.

At each time step t, the Decision Transformer encoding x_t is fed into four linear decoders. Three decoders predict the next command's action, modifier, and object components, while the fourth decoder predicts the answer to the question (this is the same at each time step). Although we also use a separate QA module to predict the final answer, the answer decoder allows the Decision Transformer to learn some primitive level of question answering, thereby allowing the QA loss to help guide command generation. Chen et al. (2021) found that predicting states and returns-to-go at each time step did not improve performance. This motivates exclusively predicting command triples, along with answers to questions.

The model is trained to optimize the sum of the cross-entropy losses of action, modifier, object, and answer prediction. For each question and map type configuration, a set of unique validation games were generated wherein an agent must interact with an environment to answer a question. During training, the Decision Transformer is evaluated on the set of hold-out games every 250 iterations to monitor sufficient information scores and to avoid overfitting.

3.3 QA module

The QA module consists of a pretrained BERT encoder with a linear classification layer. The per-time step state sequences are joined into a single long sequence of tokens with the question appended at the end. A `[CLS]` token is added to the beginning of the sequence, and a `[SEP]` token before and after the question. This concatenated sequence is tokenised by a *Bert-Base-Uncased* tokeniser (Devlin et al., 2019) and padded or front-truncated (keeping the most recent part of the state sequence) to return a 512 token sequence which is then fed to the BERT encoder. We subsequently pass BERT's output vector corresponding to the CLS token to a linear layer. For attribute and existence questions this model performs binary classification (to predict "yes" or "no"), while for location questions it produces a softmax over the vocabulary. We use cross-entropy to calculate the loss between predicted and correct answers.

Training and validation sets that consist entirely of valid trajectories are created for the QA module. We use the validation set (20% of the generated trajectories) to save the model with the highest validation QA accuracy (after 30 training epochs). In order to simulate more realistic QA training data, we feed the QA training examples back into the trained DT and use it to predict where to cut off the generated sequence, as during testing there is no guarantee that the DT would have explored up until the correct answer has been found. The sequence is cut off when the DT predicts the stop action (*wait*) or the time step limit is exceeded.

3.4 Decoding

To generate the command sequence from the DT during testing, instead of greedy decoding we sample each next command from the probability distri-

Model	Location		Existence		Attribute	
	Fixed	Random	Fixed	Random	Fixed	Random
DQN	0.224(0.244)	0.204(0.216)	0.674(0.279)	0.678(0.214)	0.534(0.014)	0.530(0.017)
DDQN	0.218(0.228)	0.222(0.246)	0.626(0.213)	0.656(0.188)	0.508(0.026)	0.486(0.023)
Rainbow	0.190(0.196)	0.172(0.178)	0.656(0.207)	0.678(0.191)	0.496(0.029)	0.494(0.017)
DT	0.168(0.232)	0.104(0.264)	0.668(0.254)	**0.722(0.277)**	0.504(0.057)	0.526(0.058)
DT-BERT	**0.232(0.232)**	**0.270(0.264)**	0.626(0.258)	0.654(0.277)	0.524(0.058)	**0.538(0.060)**
DT-10K	0.146(0.302)	0.102(0.220)	**0.688(0.240)**	0.618(0.255)	0.488(0.058)	0.490(0.048)
DT-BERT-10K	0.124(0.302)	0.076(0.204)	0.612(0.241)	0.676(0.223)	**0.552(0.060)**	0.518(0.049)

Table 2: QA accuracy and sufficient information score (in brackets) of each model following zero-shot evaluation on the test set in the 500 games setting. A bold value indicates a score to be the highest of that question and map type configuration.

butions over the action, modifier, and object. This motivation is similar to that of stochastic decoding algorithms in natural language generation: The stochasticity minimises the risk of the Decision Transformer entering a loop in which the same command is generated repeatedly, and more closely mirrors natural language which avoids utterances with too high probability (Zarrieß et al., 2021). In the case of answer prediction, we deterministically take the argmax of the output.

3.5 Returns Tuning

A shortcoming in the DT's methodology is the expectation for the environment's maximum achievable return to be known *a priori*. Using an inappropriate value can greatly hamper performance, resulting in premature halting or needlessly excessive exploration. Thus, we tune the value of the initial returns-to-go R_1 as a hyperparameter. We create a validation set of 50 games to evaluate the question answering performance of both the DT's answer prediction head and BERT model. For each question and map type we consider R_1 either set as a fixed value or sampled from an exponential distribution. Following the methodology used by Yuan et al. (2019) for selecting the best model during training, we tune R_1 to maximize the sum of the sufficient information score and question answering accuracy. See Appendix A for details.

4 Results

We evaluate the Decision Transformer both where its own answer prediction head is used for questions answering and where this is done with the BERT QA model. Test set results on the 500 games setting are given in Table 2, together with with RL model results as reported by Yuan et al. (2019). We also report results of training the DT on reduced datasets with only 10,000 episodes, in order to further evaluate the sample efficiency of our approach (see section 4.5). Training results are available in the Appendix in Tables 7 and 8. Table 3 gives the BERT QA model's validation accuracy. We discuss the results for each question type.

Overall, DT-BERT outperforms the Decision Transformer's answer prediction head in location and attribute type questions, while the DT gives a higher accuracy on existence questions. At a high level, these performance differences depend on the state and action space that the model was required to learn and navigate. Existence type and attribute type questions may depend on long-range dependencies. For example, existence type questions require the ability to know whether an object has been witnessed or not. When the answer is that an object doesn't exist, the problem is more than just word matching within the last few states. We believe that this is why the Decision Transformer QA head outperforms the BERT model on existence and attribute questions since it has access to the entire state trajectory. On questions whose answers were more likely to be found within the last 512 tokens, DT-BERT achieves higher question answering accuracy than the Decision Transformer.

4.1 Location Questions

The DT's answer prediction head has a lower QA accuracy than previous RL approaches on both fixed and random maps. However its sufficient information scores, reflecting the DT's information gathering capacities, are higher. For location type questions, QA accuracy normally matches suffi-

Question	Map Type	BERT	BERT-10K
Attribute	Fixed	0.780	0.730
	Random	0.616	0.703
Existence	Fixed	0.778	0.762
	Random	0.779	0.778
Location	Fixed	0.987	0.831
	Random	0.988	0.835

Table 3: QA accuracy of the BERT model and the BERT-10K model on the QA validation data.

cient information results due to QA modules effectively performing word matching once an agent arrives in the correct state. We see this in the RL methods' results as their sufficient information scores are very close to their QA accuracies. This indicates that the DT's question answering prediction head is underfitting the training data.

DT-BERT outperforms the QA accuracy of the RL models on both fixed and random maps. On random maps, QA accuracy is slightly *higher* than sufficient information, suggesting that in a small number of cases the BERT model may be able to deduce the answer from the context even when it does not explicitly appears in the trajectory. The performance gap between the BERT QA model and the DT means that a question can still be correctly answered even if the DT stops in an incorrect state. The high BERT QA accuracy for location questions can also be seen in Table 3.

We suggest two reasons for the BERT QA model answering location type questions more accurately than the Decision Transformer's prediction head. First, it is easier for BERT model to learns skills basic pattern matching skills such as identifying entity and location names from state strings. Second, exploration is not as highly encouraged with location questions as with existence and attribute types. Less exploration means fewer states visited, allowing the state context window to contain less noisy state strings than other question types.

4.2 Existence Questions

The DT outperforms RL baselines on sufficient information and QA accuracy in the random maps setting for existence questions. However on fixed maps it performs worse than the DQN. The BERT QA model underperforms the DT answer prediction head here in both map types, suggesting that jointly optimising answer and command prediction leads to improved performance on existence type questions.

Reasoning about the existence of an object within a TextWorld environment requires knowledge about the entirety of the world. Therefore, existential questions require an agent to fully explore an environment to answer whether or not an entity exists within it. The Decision Transformer's self-attention mechanism makes performing long-term credit assignments possible. The answer prediction head of the DT can thus draw upon information gathered in all previous states to inform question answering. As a result, the ability to model dependencies that stretch throughout all states encountered allows the DT to outperform the BERT model, whose context window is constrained to 512 tokens.

4.3 Attribute Questions

None of the models achieve results that are substantially above 50% on attribute questions, confirming the challenge of this question type. The Decision Transformer did obtain higher sufficient information than all RL baselines. DT-BERT obtains higher QA accuracies than the DT answer prediction head; it obtains the highest QA accuracy among all the models on random maps, and performs slightly worse than the DQN on fixed maps.

Despite the Decision Transformer's ability to learn long-term dependencies via its attention mechanism, we posit that the contextualised embeddings of BERT are able to model a richer semantic representation of TextWorld's state-strings than the embeddings learnt by the DT. This better capturing of the semantic space enables BERT to more fully utilise the context with which it was provided by using pre-existing understanding to help answer questions posed in natural language.

4.4 Rewards and Performance

Based on validation set performance, the optimal initial return-to-go for location type questions was determined to be 2.0 for both fixed and random map settings. This is lower than for existence and attribute types, indicating that exploration is not as highly encouraged. In location type questions, the entity definitively exists somewhere within the environment. This means that the action space required to answer questions of locality is reduced to traversals and basic interactions with containers. Therefore, less exploration is needed as the information to answer a question is more easily ac-

quired. Too high an initial reward would promote unnecessary actions with a high likelihood of leading the agent astray from stopping in the correct state.

Existence questions require far more exploration of an environment than location type questions. Higher starting rewards reflect this need for greater exploration and are associated with better QA and sufficient information scores, as seen in Table 5 in the Appendix. These higher values promote a more complete traversal of the world, allowing for gathering information required to answer the question. However, too high an initial reward means that entering a correct state and receiving a reward of 1.0 may not affect the model's decision making. If the DT has a current RTG of 5.0 and enters the correct state that rewards 1.0, the RTG from then onwards is 4.0. The return-to-go of 4.0 does not suggest to the model that it has entered the correct state, meaning it carries on exploring and gathering information. Likewise, too small a reward could prematurely cause an agent to stop exploring due to gaining rewards for entering new states via the exploration bonus. Therefore, we observe that the best RTG values err on the larger side, which encourages greater world exploration.

Attribute type questions are considered the most sparsely rewarded of all three types (Yuan et al., 2019). We therefore expected higher rewards to be associated with better accuracies. The results, however, paint a different picture. In a fixed map, where the state space is, on average, smaller than that of random maps, we see that a smaller reward yields the best score. This reduction is likely a result of the reduced state and action space making too much exploration and interaction with the environment degrade performance. On the other hand, in a random map setting higher rewards yields better QA and sufficient information scores, allowing us to conclude that higher rewards promote more exploration and thus allows the model to better answer the question.

4.5 Sample Efficiency

The RL agents in QAit were trained for more than 200K episodes. In comparison, most of our Decision Transformers were trained on around 40K episodes (Table 1). The test set results therefore show that DT is able to match or outperform the previous RL methods when trained on approximately 25% of the number of episodes. Moreover, all training data used for the DT was generated via random rollouts - indicating that the Decision Transformer has the ability to learn optimal policies from suboptimal data. We also found that fine-tuning a BERT model for QA on the random rollout data works well, as long as the DT is used to determine where to cut off the trajectory.

In order to further elucidate the DT's sample efficient learning capabilities, we generated new datasets for all question and map types that only contained 10 thousand episodes. The validation results can be seen Table 6 in the Appendix. These experiments indicate that the DT trained on even fewer offline trajectories can achieve results on par with or better than both previous baselines as well as identical models trained on more data. Here we see fixed map sufficient information scores being improved for all question types and QA accuracy increasing for attribute and existence questions. However, QA accuracy for location type questions is worse than previous baselines in both random and fixed maps (see Table 2). While the results are not consistently better, they do further indicate the sample efficiency of the Decision Transformer.

5 Conclusion

We showed that interactive question answering can be framed as a sequence modelling problem by training Transformers for action generation and answer prediction using random roll-outs. Results show that the Decision Transformer approach matches or outperforms current reinforcement learning approaches for QAit on most question types and maps type configurations in the 500 game setting. Additionally, the approach is more sample efficient than reinforcement learning approaches, reducing the amount of training data required even though the data generated via random rollouts is suboptimal. Fine-tuning a BERT model for question answering on the same generated dataset improves performance over using the Decision Transformer directly for question answering in two of the three question types.

6 Acknowledgements

This work is based on research supported in part by the National Research Foundation of South Africa (Grant Number: 129850). Some computations were performed using facilities provided by the University of Cape Town's ICTS High Performance Computing.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. 2018. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer.

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.

Michael P Georgeff and Amy L Lansky. 1986. Procedural knowledge. *Proceedings of the IEEE*, 74(10):1383–1398.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2018. Iqa: Visual question answering in interactive environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4089–4098.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*.

Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Jieh-Sheng Lee and Jieh Hsiang. 2019. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124*.

Manuel Mager, Ramón Fernandez Astudillo, Tahira Naseem, Md Arafat Sultan, Young-Suk Lee, Radu Florian, and Salim Roukos. 2020. GPT-too: A language-model-first approach for AMR-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1846–1852, Online. Association for Computational Linguistics.

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(7540):529–533.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-shot text-to-image generation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Xingdi Yuan, Marc-Alexandre Côté, Jie Fu, Zhouhan Lin, Chris Pal, Yoshua Bengio, and Adam Trischler.

2019. Interactive language learning by question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2796–2813, Hong Kong, China. Association for Computational Linguistics.

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.

Sina Zarrieß, Henrik Voigt, and Simeon Schüz. 2021. Decoding methods in neural language generation: A survey. *Information*, 12(9).

A Hyperparameter Tuning

Hyperparameters of the DT and BERT are given in Table 4.

	DT	BERT
Hyperparameters	**Value**	
Number of layers	2	12
Number of attention heads	8	12
Embedding dimension	256	768
Batch size	128	12
State context window tokens	180	512
Context length (K)	50	-
Max Epochs	2000	30
Dropout	0.5	0.1
Learning rate	1×10^{-4}	1×10^{-5}
Adam betas	(0.9, 0.95)	
Grad norm clip	0.25	
Weight decay	0.1	

Table 4: Decision Transformer and the BERT QA hyperparameters. For the DT, Context length *K* refers to the amount of previous time steps with which the Transformer can conditiona on. Context State context window refers to the number of tokens from the state to be used for prediction. Adam (Kingma and Ba, 2015) is used as optimiser in conjunction with the specified learning rate, linear warmup and cosine weight decay.

A.1 Decision Transformer

A.1.1 Location

As can be seen in Table 5, the sufficient information score peaking at $R_1 = 2$ indicates optimal state-space exploration for location questions when the potential for future reward is moderate for random and fixed map types. While the QA accuracy was highest for both settings when the initial reward was the maximum of the training set, we opted to test the DT's question answering and information gathering capabilities at $R_1 = 2$ as this yielded the highest combined sufficient information and QA accuracy score.

A.1.2 Existence

Using both sufficient information and QA accuracy, the optimal initial reward for fixed map existence questions was determined to be 4.0, with the DT achieving a QA accuracy of 0.660 and a sufficient information score of 0.263 on the validation set. In random map settings, the DT scored a validation accuracy of 0.720 with a corresponding sufficient information score of 0.298, where the initial reward was determined to be the maximum of the training set 3.94.

A.1.3 Attribute

The best sufficient information and QA accuracy combinations for the Decision Transformer were achieved at an initial reward of 2.0 for fixed and 5.0 for random map types. On the validation set, the fixed map DT achieved a QA accuracy of 0.533 and a SI score of 0.056. Random map saw a similar SI of 0.057 but worse QA accuracy of 0.460.

A.2 BERT Model

A.2.1 Location

Based on data gathered using the online-evaluation dataset, the optimal initial return-to-go for location type questions was 2.0. Using the BERT model for QA yielded an accuracy of 0.227 for fixed maps and 0.393 for random maps. The BERT model achieved a higher QA accuracy than sufficient information score during evaluation, indicating that the context window spanning multiple states was a boon to QA accuracy. During training, the BERT model achieved almost perfect scores for question-answering on the held-out set of offline trajectories, seen in Table 3.

A.2.2 Existence

Using the online-validation set, we determined optimal starting reward values of 3.0 for fixed map and 3.94 for random. These values were associated with a QA accuracy of 0.64 for fixed and 0.647 for random map types. However, scores were significantly lower than the offline validation set used during training, where QA accuracy of 0.778 and 0.779 was achieved for fixed and random maps, respectively.

A.2.3 Attribute

In the offline validation set, the BERT model scored a QA accuracy of 0.616 for random and 0.780 for fixed map settings. On the online validation set, we observed the maximum combination of QA and sufficient information for the BERT model at an R_1 of 3.0 for fixed and 2.0 for random where the BERT QA model had an accuracy of 0.507 and 0.660 for random and fixed map types, respectively. However, we opted to use the maximum of the train set 4.03 when evaluating on the test set for random map types. This is due to the BERT QA model having a high standard deviation of 0.156 and an average QA accuracy of 0.640, indicating greater potential for high QA accuracy. Moreover, the sufficient information score associated with this accuracy is 0.056 - higher than the random map with an initial reward of 2.0.

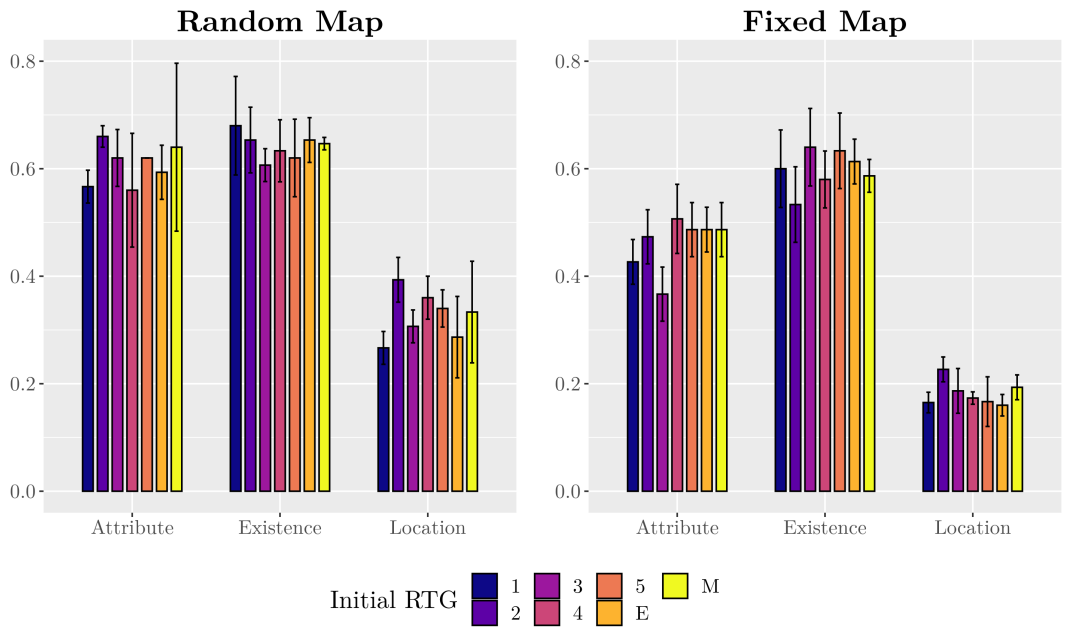

Figure 3: Barplot showing QA accuracy of the BERT QA model on the validation set when trained in the 500 games setting with different initial returns-to-go (RTG). See results in Table 5.

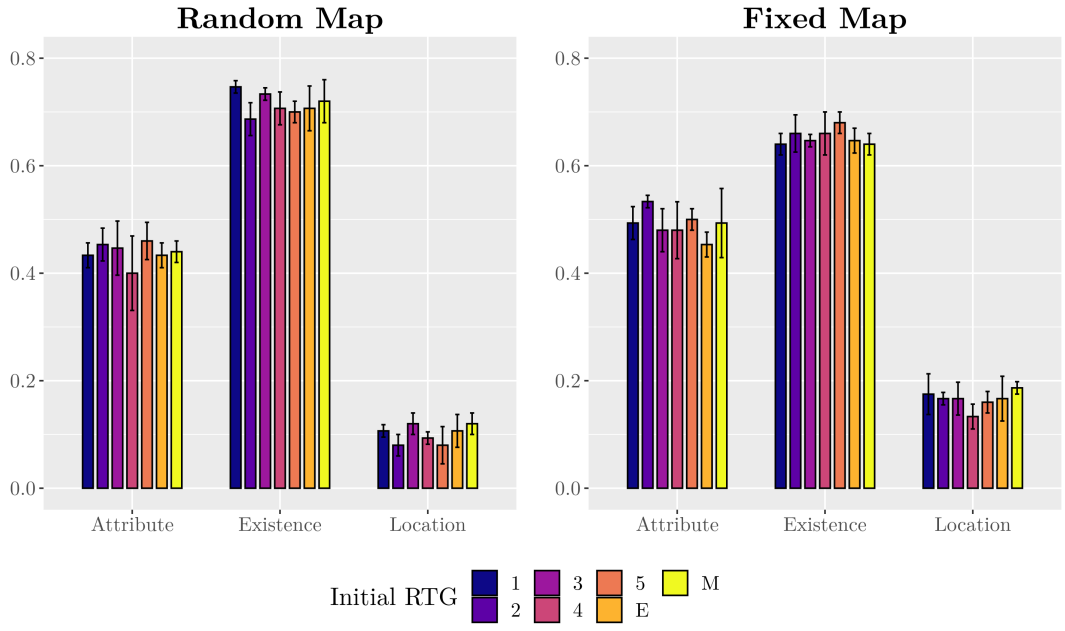

Figure 4: Barplot showing QA accuracy of the Decision Transformer's answer-prediction head on the validation set when trained in the 500 games setting with different initial returns-to-go (RTG). See results in Table 5.

Question Type						
Attribute						
	Fixed			**Random**		
Initial RTG	**BERT**	**DT**	**SI**	**BERT**	**DT**	**SI**
1	0.427 ± 0.0416	0.493 ± 0.0306	0.052 ± 0.0135	0.567 ± 0.0306	0.433 ± 0.0231	0.050 ± 0.0083
2	0.473 ± 0.0503	**0.533 ± 0.0115**	**0.056 ± 0.0094**	**0.660 ± 0.0200**	0.453 ± 0.0306	**0.048 ± 0.0039**
3	0.367 ± 0.0503	0.480 ± 0.0400	0.051 ± 0.0043	0.620 ± 0.0529	0.447 ± 0.0503	0.054 ± 0.0034
4	**0.507 ± 0.0643**	0.480 ± 0.0529	**0.056 ± 0.0058**	0.560 ± 0.1058	0.400 ± 0.0693	0.054 ± 0.0014
5	0.487 ± 0.0503	0.500 ± 0.0200	0.056 ± 0.0007	0.620 ± 0.0000	**0.460 ± 0.0346**	**0.057 ± 0.0033**
Sampling	0.487 ± 0.0416	0.453 ± 0.0231	0.051 ± 0.0050	0.593 ± 0.0503	0.433 ± 0.0231	0.052 ± 0.0097
Max	0.487 ± 0.0503	0.493 ± 0.0643	0.055 ± 0.0021	**0.640 ± 0.1562**	0.440 ± 0.0200	**0.056 ± 0.0020**
Existence						
	Fixed			**Random**		
Initial RTG	**BERT**	**DT**	**SI**	**BERT**	**DT**	**SI**
1	0.600 ± 0.0721	0.640 ± 0.0200	0.216 ± 0.0314	0.680 ± 0.0917	0.747 ± 0.0115	0.200 ± 0.0416
2	0.533 ± 0.0702	0.660 ± 0.0346	0.259 ± 0.0051	0.653 ± 0.0611	0.687 ± 0.0306	0.277 ± 0.0441
3	**0.640 ± 0.0721**	0.647 ± 0.0115	**0.265 ± 0.0161**	0.607 ± 0.0306	0.733 ± 0.0115	0.269 ± 0.0075
4	0.580 ± 0.0529	**0.660 ± 0.0400**	**0.263 ± 0.0180**	0.633 ± 0.0577	0.707 ± 0.0306	0.291 ± 0.0476
5	0.633 ± 0.0702	0.680 ± 0.0200	0.222 ± 0.0166	0.620 ± 0.0721	0.700 ± 0.0200	0.310 ± 0.0205
Sampling	0.613 ± 0.0416	0.647 ± 0.0231	0.180 ± 0.0467	0.653 ± 0.0416	0.707 ± 0.0416	0.250 ± 0.0070
Max	0.587 ± 0.0306	0.640 ± 0.0200	0.270 ± 0.0409	**0.647 ± 0.0115**	**0.720 ± 0.0400**	**0.298 ± 0.0302**
Location						
	Fixed			**Random**		
Initial RTG	**BERT**	**DT**	**SI**	**BERT**	**DT**	**SI**
1	0.165 ± 0.0191	0.175 ± 0.0379	0.165 ± 0.0191	0.267 ± 0.0306	0.107 ± 0.0115	0.267 ± 0.0306
2	**0.227 ± 0.0231**	**0.167 ± 0.0115**	**0.233 ± 0.0115**	**0.393 ± 0.0416**	**0.080 ± 0.0200**	**0.387 ± 0.0503**
3	0.187 ± 0.0416	0.167 ± 0.0306	0.187 ± 0.0416	0.307 ± 0.0306	0.120 ± 0.0200	0.307 ± 0.0306
4	0.173 ± 0.0115	0.133 ± 0.0231	0.173 ± 0.0115	0.360 ± 0.0400	0.093 ± 0.0115	0.347 ± 0.0306
5	0.167 ± 0.0462	0.160 ± 0.0200	0.167 ± 0.0462	0.340 ± 0.0346	0.080 ± 0.0346	0.333 ± 0.0306
Sampling	0.160 ± 0.0200	0.167 ± 0.0416	0.167 ± 0.0115	0.287 ± 0.0757	0.107 ± 0.0306	0.287 ± 0.0757
Max	0.193 ± 0.0231	0.187 ± 0.0115	0.193 ± 0.0231	0.333 ± 0.0945	0.120 ± 0.0200	0.327 ± 0.0702

Table 5: Question-answering accuracy of the BERT model's and the Decision Transformer's answer prediction head as well as the Decision Transformer's average sufficient information (SI) score on validation set at different initial return-to-go (RTG) values. Bold values indicate the combined highest QA and sufficient information score with the associated initial RTG value also bolded. *Sampling* indicates R_1 was randomly sampled from an exponential distribution. *Max* represents the maximum of the training set for that experiment configuration (see Table 1). Summary statistics were calculated over 4 seeds - see code implementation for details.

Question Type						
Attribute						
	Fixed			**Random**		
Initial RTG	**BERT-10K**	**DT-10K**	**SI**	**BERT-10K**	**DT-10K**	**SI**
1	0.515 ± 0.0719	**0.590 ± 0.0258**	0.054 ± 0.0074	0.490 ± 0.0529	0.410 ± 0.0258	0.053 ± 0.0123
2	0.485 ± 0.0342	0.580 ± 0.0542	0.051 ± 0.0094	**0.510 ± 0.1013**	0.410 ± 0.0115	0.044 ± 0.0061
3	0.450 ± 0.0600	0.550 ± 0.0258	0.055 ± 0.0023	0.450 ± 0.0416	0.420 ± 0.0432	0.047 ± 0.0082
4	0.440 ± 0.0163	0.580 ± 0.0365	0.056 ± 0.0040	0.430 ± 0.0775	**0.445 ± 0.0191**	0.051 ± 0.0071
5	**0.525 ± 0.0526**	0.560 ± 0.0163	0.055 ± 0.0019	0.470 ± 0.0825	0.410 ± 0.0200	0.050 ± 0.0026
Sampling	0.455 ± 0.0681	0.545 ± 0.0300	0.052 ± 0.0100	0.490 ± 0.0200	0.415 ± 0.0100	0.044 ± 0.0042
Max	0.420 ± 0.0283	0.560 ± 0.0163	0.051 ± 0.0052	0.470 ± 0.0476	0.410 ± 0.0383	0.054 ± 0.0102
Existence						
	Fixed			**Random**		
Initial RTG	**BERT-10K**	**DT-10K**	**SI**	**BERT-10K**	**DT-10K**	**SI**
1	0.595 ± 0.0574	0.590 ± 0.0258	0.165 ± 0.0148	0.740 ± 0.0400	0.705 ± 0.0300	0.165 ± 0.0143
2	0.575 ± 0.0412	0.625 ± 0.0252	0.195 ± 0.0256	0.640 ± 0.0566	0.680 ± 0.0283	0.219 ± 0.0318
3	0.610 ± 0.0258	0.645 ± 0.0342	0.232 ± 0.0235	0.620 ± 0.1007	0.690 ± 0.0600	0.233 ± 0.0303
4	**0.650 ± 0.0346**	0.640 ± 0.0365	0.251 ± 0.0538	**0.685 ± 0.0252**	0.670 ± 0.0346	0.240 ± 0.0175
5	0.560 ± 0.0283	**0.690 ± 0.0663**	0.286 ± 0.0421	0.645 ± 0.0661	**0.685 ± 0.0473**	0.253 ± 0.0362
Sampling	0.645 ± 0.0551	0.655 ± 0.0300	0.214 ± 0.0305	0.660 ± 0.0283	0.725 ± 0.0252	0.179 ± 0.0372
Max	0.635 ± 0.0823	0.630 ± 0.0702	0.229 ± 0.0471	0.630 ± 0.0577	0.675 ± 0.0379	0.243 ± 0.0328
Location						
	Fixed			**Random**		
Initial RTG	**BERT-10K**	**DT-10K**	**SI**	**BERT-10K**	**DT-10K**	**SI**
1	0.150 ± 0.0346	0.130 ± 0.0115	0.195 ± 0.0300	0.130 ± 0.0258	**0.130 ± 0.0115**	0.170 ± 0.0476
2	0.135 ± 0.0342	0.155 ± 0.0500	0.190 ± 0.0115	0.130 ± 0.0258	0.105 ± 0.0300	0.165 ± 0.0100
3	0.155 ± 0.0300	0.150 ± 0.0383	0.220 ± 0.0432	0.155 ± 0.0526	0.105 ± 0.0342	0.135 ± 0.0300
4	0.135 ± 0.0500	0.130 ± 0.0258	0.230 ± 0.0200	0.145 ± 0.0300	0.120 ± 0.0432	0.150 ± 0.0346
5	0.135 ± 0.0300	0.135 ± 0.0473	0.230 ± 0.0258	**0.160 ± 0.0432**	0.110 ± 0.0383	0.170 ± 0.0200
Sampling	0.150 ± 0.0383	0.145 ± 0.0100	0.180 ± 0.0163	0.125 ± 0.0412	0.105 ± 0.0100	0.160 ± 0.0163
Max	**0.145 ± 0.0252**	**0.150 ± 0.0115**	0.240 ± 0.0327	0.145 ± 0.0300	0.120 ± 0.0432	0.150 ± 0.0346

Table 6: Question-answering accuracy of the 10K variation BERT model's and Decision Transformer's answer prediction head as well as the Decision Transformer's average sufficient information (SI) score on validation set at different initial return-to-go (RTG) values. Bold values indicate the combined highest QA and sufficient information score with the associated initial RTG value also bolded. Both models were trained on only 10 thousand episodes of data.

Fixed						
Model	**Location**		**Existence**		**Attribute**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Random	-	0.027	-	0.497	-	0.496
500 games						
DQN	0.430 (0.430)	0.224 (0.244)	0.742 (0.136)	0.674 (0.279)	0.700 (0.015)	0.534 (0.014)
DDQN	0.406 (0.406)	0.218 (0.228)	0.734 (0.173)	0.626 (0.213)	0.714 (0.021)	0.508 (0.026)
Rainbow	0.358 (0.358)	0.190 (0.196)	0.768 (0.187)	0.656 (0.207)	0.736 (0.032)	0.496 (0.029)
DT	-	0.168 (0.232)	-	0.668 (0.254)	-	0.504 (0.057)
DT-BERT	-	**0.232 (0.232)**	-	0.626 (0.258)	-	0.524 (0.058)
DT - 10K	-	0.146 (0.302)	-	**0.688 (0.240)**	-	0.488 (0.058)
DT-BERT - 10K	-	0.124 (0.302)	-	0.612 (0.241)	-	**0.552 (0.060)**

Table 7: Results of Fixed Map Experiments

Random						
Model	**Location**		**Existence**		**Attribute**	
	Train	**Test**	**Train**	**Test**	**Train**	**Test**
Random	-	0.034	-	0.5	-	0.499
500 games						
DQN	0.430 (0.430)	0.204 (0.216)	0.752 (0.162)	0.678 (0.214)	0.678 (0.019)	0.530 (0.017)
DDQN	0.458 (0.458)	0.222 (0.246)	0.754 (0.158)	0.656 (0.188)	0.716 (0.024)	0.486 (0.023)
Rainbow	0.370 (0.370)	0.172 (0.178)	0.748 (0.275)	0.678 (0.191)	0.636 (0.020)	0.494 (0.017)
DT	-	0.104 (0.264)	-	**0.722 (0.277)**	-	0.526 (0.058)
DT-BERT	-	**0.270 (0.264)**	-	0.654 (0.277)	-	**0.538 (0.060)**
DT - 10K	-	0.102 (0.220)	-	0.618 (0.255)	-	0.490 (0.048)
DT-BERT - 10K	-	0.076 (0.204)	-	0.676 (0.223)	-	0.518 (0.049)

Table 8: Results of Random Map Experiments