Language Models are Few-Shot Butlers

Vincent Micheli
University of Geneva
vincent.micheli@unige.ch

François Fleuret
University of Geneva
francois.fleuret@unige.ch

Abstract

Pretrained language models demonstrate strong performance in most NLP tasks when fine-tuned on small task-specific datasets. Hence, these autoregressive models constitute ideal agents to operate in text-based environments where language understanding and generative capabilities are essential. Nonetheless, collecting expert demonstrations in such environments is a time-consuming endeavour. We introduce a two-stage procedure to learn from a small set of demonstrations and further improve by interacting with an environment. We show that language models fine-tuned with only 1.2% of the expert demonstrations and a simple reinforcement learning algorithm achieve a 51% absolute improvement in success rate over existing methods in the ALFWorld environment.

1 Introduction

Over the past few years, successive generations of language models (Radford et al., 2018, 2019; Brown et al., 2020) have reshaped the way we approach Natural Language Processing problems. These Transformer-based (Vaswani et al., 2017) networks scale to ever-increasing amounts of parameters, data and compute (Kaplan et al., 2020) while demonstrating impressive transfer-learning capabilities across a wide variety of benchmarks (Wang et al., 2018, 2019; Rajpurkar et al., 2018; Reddy et al., 2019).

However, the application of modern language models to action generation in text-based environments (Côté et al., 2018; Hausknecht et al., 2019; Shridhar et al., 2021) remains largely unexplored (Yao et al., 2020). Intuitively, large-scale models leveraging strong linguistic priors should thrive in settings where observations and actions are textual (Luketina et al., 2019), but several issues arise. Indeed, agents must conform to an environment's generative grammar and gameplay specificities making off the shelf transfer fail in most games. A way to overcome this problem is to acquire expert demonstrations and resort to the widely used paradigm of fine-tuning on task-specific data (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019). Nevertheless, collecting demonstrations in text-based environments requires far more time and expert knowledge than for most NLP tasks. A single demonstration includes tens of actions taken over a long time horizon to solve multiple subgoals.

Goal: Rinse the egg to put it in the microwave.
Obs: Looking quickly around you, you see a cabinet, a garbagecan, a coffeemachine, [...], a stoveburner, a sinkbasin and a microwave.
Action: go to sinkbasin
Obs: You arrive at sinkbasin. You see a butterknife, a potato, a spoon and a tomato.
Action: go to garbagecan
Obs: You arrive at garbagecan. You see an egg.
Action: take egg from garbagecan
Obs: You pick up the egg from the garbagecan.
Action: go to sinkbasin
Obs: You arrive at sinkbasin. You see a butterknife, a potato, a spoon and a tomato.
Action: clean egg with sinkbasin
Obs: You clean the egg using the sinkbasin.
Action: go to microwave
Obs: You arrive at microwave. The microwave is closed.
Action: open microwave
Obs: You open the microwave.
Action: put egg in/on microwave

Figure 1: Example of a human-annotated, out-of-distribution task instance solved by $\text{GPT}^{\star}_{\text{partial}}$.

In this work, we propose a two-stage procedure to address these issues and develop language models acting as agents in text-based environments.

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 9312–9318
November 7–11, 2021.

First, we train language models to imitate a few dozens of expert demonstrations in order to respect an environment's grammar and acquire basic gamesense. Second, we let the models interact with the environment and iteratively treat successful trajectories as additional expert demonstrations for further fine-tuning. We demonstrate the effectiveness of our approach in the recently introduced ALFWorld environment (Shridhar et al., 2021)[1], which was designed with an extensive set of tasks and expert demonstrations.

In summary, our contributions are the following:

1. We show that language models fine-tuned on thousands of expert demonstrations considerably outperform current methods in the ALFWorld environment.

2. We achieve strong results with a fraction of the demonstrations by combining imitation and reinforcement learning algorithms.

3. We illustrate the robustness of the models developed to human-annotated goals in realistic scenarios.

2 Methods

2.1 Background: goal-based textual environments

A goal-based textual environment can be represented as a partially observable Markov decision process $P = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{G}, R, T, M)$ where observations, actions and goals are specified in natural language. In state $s_t \in \mathcal{S}$, an agent takes action $a_t \in \mathcal{A}$ conditioned on context $c_t = (g, o_0, a_0, ..., o_t)$. It receives reward $r_t = R(s_t, a_t, g)$, which is an indicator variable for the completion of goal $g \in \mathcal{G}$, and a new observation $o_{t+1} = M(T(s_t, a_t))$, where $M : \mathcal{S} \to \mathcal{O}$ is a mapping from states to observations and $T : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition function.

2.2 Learning from demonstrations

A demonstration d consists of a sequence of observations and actions $(o_0, a_0, o_1, a_1, ..., o_T, a_T)$ for reaching goal g based on contexts $(c_0, c_1, ..., c_T)$. We consider a dataset $\mathcal{D}$ of N demonstrations. A parameterized model p_θ is trained to minimize the mean demonstration loss $\mathcal{L}_\mathcal{D} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{d_i}$ with

$$\mathcal{L}_{d_i} = -\sum_{t=0}^{T}\log p_\theta(a_t|c_t). \quad (1)$$

As noted by Yao et al. (2020),

$$\log p_\theta(a|c) = \sum_{j=1}^{m}\log p_\theta(a^j|a^{<j}, c), \quad (2)$$

where a^j is the j-th token generated in action a of length m.

We use a per-demonstration loss instead of a per-action loss to reduce computational costs. Indeed, with this formulation a Transformer-based autoregressive model can leverage previous computations when considering a new context from the same demonstration. In addition, early experiments suggested that a per-demonstration loss does not harm performance.

We call *action modeling* the process of minimizing the mean demonstration loss, which is conceptually very similar to language modeling, except that we only maximize the likelihood of action tokens instead of maximizing the likelihood of the full trajectory c_T.

2.3 Learning from interactions

While action modeling is a powerful training objective, a model learning from demonstrations is ultimately limited by the size of the training set. To circumvent this issue, we propose an *iterated action modeling* (IAM) algorithm:

1. A language model pretrained on expert demonstrations is tasked to solve a batch of goals in the environment.

2. The language model is further fine-tuned with action modeling on successful trajectories.

The key advantage of this algorithm is that we can easily combine imitation learning with reinforcement learning since we are optimizing the same objective over two distinct sources of data: demonstrations and successful attempts. Moreover, the extensively pretrained language/action modeling head is kept during reinforcement learning instead of initializing a new RL-specific head from scratch, which was shown to lead to better performance in NLP tasks (Gao et al., 2021).

[1] ALFWorld aligns both text and embodied environments, but here we only refer to the text environment.

	ALFWorld goals		Human goals	
	Seen split	Unseen split	Seen split	Unseen split
Seq2Seq (Shridhar et al., 2021)	10	9		
BUTLER (Shridhar et al., 2021)	40	37		
$\text{GPT}_{\text{partial}}$	47	40	17	22
$\text{GPT}^{\star}{}_{\text{partial}}$	69	60	32	37
GPT	91	95	42	57

Table 1: Success percentages per evaluation split (in-distribution and out-of-distribution) with and without human-annotated goals. $\text{GPT}_{\text{partial}}$ and GPT are GPT2-based models fine-tuned with action modeling on 42 and 3553 demonstrations, respectively. $\text{GPT}^{\star}{}_{\text{partial}}$ corresponds to the former model subsequently trained with iterated action modeling in ALFWorld. Our results are averaged over 5 seeds. Standard deviations are upper bounded by 9 for $\text{GPT}_{\text{partial}}$, 8 for $\text{GPT}^{\star}{}_{\text{partial}}$, and 3 for GPT.

3 Experiments

Experiments were implemented with the Transformers (Wolf et al., 2020) and PyTorch (Paszke et al., 2019) libraries and were conducted on an NVidia RTX 3090.[2]

3.1 Environment and dataset

ALFWorld (Shridhar et al., 2021) is a goal-based textual environment mirroring the embodied ALFRED benchmark (Shridhar et al., 2020) with the TextWorld game engine (Côté et al., 2018). The environment was created with the aim of learning high-level language policies inside of it and transferring them to the embodied setting. ALFWorld inludes 6 tasks that are compositional and require multiple sub-goals to be solved over various time horizons. Any string of words constitute a valid action making the action space unbounded and the training of a policy consequently difficult. In total there are 3553 training task instances {task-type, object, receptacle, room}, 140 in-distribution evaluation task instances (seen split) and 134 out-of-distribution evaluation task instances (unseen split). A task instance specifies the type of the task to solve, the object to interact with, the receptacle where the object should be put and the room layout (e.g. {heat and place, egg, countertop, kitchen 12}). Besides, each training task instance in ALFWorld comes with an expert demonstration, enabling the development of imitation learning agents.

[2]Source code and links to models available at: https://github.com/vmicheli/lm-butlers

3.2 Training

We train two GPT2-medium (345M parameters) (Radford et al., 2019) models with action modeling on the set of demonstrations. The first model, GPT, has access to the full set of demonstrations while the second model, $\text{GPT}_{\text{partial}}$, only has access to 42 demonstrations. $\text{GPT}_{\text{partial}}$ is subsequently trained with iterated action modeling in the environment and is then denoted as $\text{GPT}^{\star}{}_{\text{partial}}$. When interacting with the environment, models greedily decode actions token-per-token until an end of action token is reached. See Appendix A for training details.

3.3 Evaluation

We select model checkpoints according to their evaluation performance on the seen split and further evaluate them on the unseen split. During evaluation, we employ greedy action decoding and a sliding context window which depends on the maximum number of tokens the language models can handle. This implies that the contexts given to the models consist of the goal, the first observation and as many of the previous observations and actions as possible. We compare our models with the ones developed by Shridhar et al. (2021):

- BUTLER: trained with Dagger (Ross et al., 2011) for 50k episodes and handling failed actions with beam search.
- Seq2Seq: trained with the full set of demonstrations.

Contrary to our approach, these models do not encapsulate prior linguistic knowledge except from pretrained word embeddings.

3.4 Robustness

In ALFWorld, goals follow a generative grammar specific to the environment, e.g. "put a hot apple in fridge". However, when interacting with autonomous agents, humans may formulate goals that deviate from this grammar, e.g. "warm up apple to put in fridge". The ability to generalize to human-annotated goals is quantitatively assessed with crowd-sourced goal annotations (Shridhar et al., 2020, 2021). We evaluate the best performing models from Section 3.3 on the human-annotated seen and unseen splits.

4 Results

4.1 Language models strongly outperform existing methods in ALFWorld

We report the entirety of the results in Table 1. GPT achieves success rates of 91% and 95%, respectively, on the seen and unseen splits. That is, absolute improvements of 81% and 86% over the Seq2Seq model trained on the same data. Even when compared to BUTLER, trained with 14 times more expert-guided demonstrations and manually handling failed actions, we observe absolute improvements of 51% and 58%. $\text{GPT}_{\text{partial}}$ is also competitive with BUTLER and outperforms the Seq2Seq model with only 0.07% and 1.2% of the expert demonstrations available. However, there remains a large performance gap between the two GPT2-based models.

4.2 Iterated action modeling retains most of the performance with few demonstrations

With iterated action modeling, $\text{GPT}_{\text{partial}}$'s performance improves by 22% and 20%, respectively, on the seen and unseen splits. In other words, $\text{GPT}^{\star}_{\text{partial}}$ retains 76% and 63% of GPT's results with only 1.2% of the expert demonstrations available.

4.3 Agents with linguistic priors are robust to human-annotated goals

Evaluation on the seen and unseen splits with human-annotated goals reveals that language models fine-tuned with action modeling on expert demonstrations and successful trajectories are capable of solving a large proportion of goals formulated in open-ended natural language. For example, GPT and $\text{GPT}^{\star}_{\text{partial}}$ solve respectively 57% and 37% of human-annotated, out-of-distribution task instances. Figure 1 illustrates $\text{GPT}^{\star}_{\text{partial}}$ solving one of these tasks.

5 Related work

Yao et al. (2020) used language models to prune the action space in text-based games. The authors introduced the ClubFloyd dataset, which contains gameplay transcripts collected over a multitude of games, and fine-tuned a GPT2-small (117M parameters) (Radford et al., 2019) model on that dataset for action generation. This contextual action language model (CALM) was then queried to generate a small list of action candidates based on the last few observations and actions. CALM was combined with game-specific models trained with reinforcement learning (He et al., 2016) to pick the best action candidate among CALM's generations. This approach aims to transfer a general-purpose language model across multiple new environments without game-specific imitation or reinforcement learning. In our work, we optimize for performance instead of generalization by training language models with game-specific demonstrations and interactions. In fact, preliminary experiments with CALM in ALFWorld reveal that the model is unable to produce valid actions both in terms of grammar and task completion.

Goal-conditioned supervised learning (Ghosh et al., 2021) treats every trajectory as an expert demonstration for reaching the final state encountered in that same trajectory. This hindsight goal-relabeling is possible because there exists a straightforward mapping between goals and states in the environments considered (i.e. the identity map). In ALFWorld, learning such a mapping is highly non-trivial and constitutes another research direction for extending existing methods (Cideron et al., 2019) to this environment. Therefore, during iterated action modeling we only consider successful trajectories as expert demonstrations and initialize the agent with a few demonstrations in order to start the RL procedure with a non-zero success rate.

6 Conclusion

We developed new agents for text-based environments with pretrained language models. These agents acquired game knowledge through demonstrations and interactions to drastically outperform current methods in the ALFWorld environment. While we investigated learning under the standard fine-tuning paradigm, more sophisticated ap-

proaches could be explored (Schick and Schütze, 2020) and recent works (Brown et al., 2020; Zhao et al., 2021) even suggest that scaled-up and carefully calibrated models achieve great downstream results without requiring any parameter updates. Thus, in the near future one can imagine language models solving text-based environments with only a few demonstrations for priming.

Acknowledgments

We thank the ALFWorld team for their technical support regarding the environment.

Vincent Micheli was supported by the Swiss National Science Foundation under grant number FNS-187494 "Meaningful Human Control of Security Systems – Aligning International Humanitarian Law with Human Psychology".

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.

Geoffrey Cideron, Mathieu Seurin, Florian Strub, and Olivier Pietquin. 2019. Self-educated language agent with hindsight experience replay for instruction following. *CoRR*, abs/1910.09451.

Marc-Alexandre Côté, Ákos Kádár, Xingdi (Eric) Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *Computer Games Workshop at ICML/IJCAI 2018*, pages 1–29.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Manon Devin, Benjamin Eysenbach, and Sergey Levine. 2021. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations*.

Matthew J. Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. 2019. Interactive fiction games: A colossal adventure. *CoRR*, abs/1909.05398.

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, Berlin, Germany. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. 2019. A survey of reinforcement learning informed by natural language. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6309–6317. International Joint Conferences on Artificial Intelligence Organization.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *CoRR*, abs/2012.11926.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Cote, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. {ALFW}orld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations*.

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.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280.

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, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Shunyu Yao, Rohan Rao, Matthew Hausknecht, and Karthik Narasimhan. 2020. Keep CALM and explore: Language models for action generation in text-based games. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8736–8754, Online. Association for Computational Linguistics.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

A Hyperparameters and sample selection

We do not leverage any (potentially large) held-out set of demonstrations to tune hyperparameters or learning objectives. As mentioned in 3.3, we solely optimize the success rate over a small set of validation task instances that we can freely query rather than a validation loss on held-out examples. Hyperparameters for the action modeling and iterated action modeling experiments are displayed in Table 2 and Table 3. For the action modeling experiments with $\text{GPT}_{\text{partial}}$, we randomly select 7 demonstrations per task-type from the pool of 3553 demonstrations.

Hyperparameter	GPT	$\text{GPT}_{\text{partial}}$
Epochs	{**10**, 20}	100
Batch size	1	1
Gradient acc. steps	8	7
Learning rate	5e-5	{1e-5, **5e-5**}
LR schedule	Linear	Constant
Adam β_1	0.9	0.9
Adam β_2	0.999	0.999
Max gradient norm	1.0	1.0
Dropout	0.1	0.1
Max sequence length	1000	1000

Table 2: Action modeling hyperparameters.

Hyperparameter	$\text{GPT}^{\star}_{\text{partial}}$
Iterations	20
Episodes per iteration	{100, 200, **400**}
Batch size	1
Gradient acc. steps	8
Learning rate	{1e-6, **1e-5**, 5e-5}
LR schedule	Constant
Adam β_1	0.9
Adam β_2	0.999
Max gradient norm	1.0
Dropout	0.1
Max action length	20
Max sequence length	1000
Action selection	Sampling

Table 3: Iterated action modeling hyperparameters.

B Performance as a function of the number of training demonstrations

In Figure 2, we provide a curve of model performance as a function of the number of training demonstrations for the action modeling stage.

Around 168 demonstrations are necessary to achieve a success rate equivalent to that of $\text{GPT}^{\star}_{\text{partial}}$. In other words, adding the iterated action modeling procedure brings improvements similar to those we would get if we multiplied the number of demonstrations by 4.

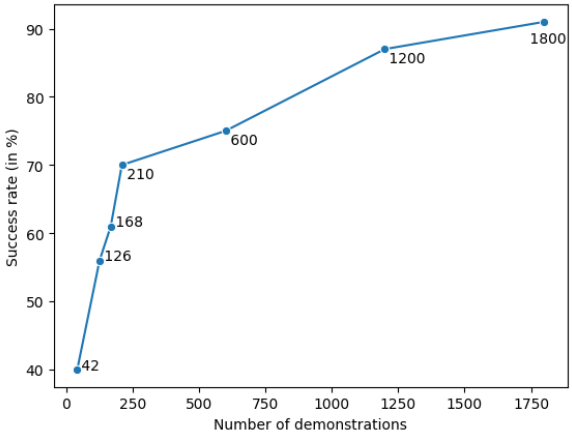

Figure 2: Unseen split performance as a function of the number of training demonstrations.

C Input representation

In practice, a context is formed in the following way:

1. Append the goal to the first observation.
2. Preprend modality strings "[STATE]" and "[ACTION]" to observations and actions.
3. Concatenate past observations and actions in a single string of text.