Learn to Read to Read to Learn

Michelangelo Conserva **Alessia Eletti** **Pauline de Lavallade** **Jaeik Jeon**

Abstract

The aim of this project is to study the benefits of informing Reinforcement Learning (RL) agents using Natural Language (NL). The setting in which we chose to explore this is that commonly referred to as Contextual Multi-Armed Bandit. In this framework, the agent must make a sequence of decisions on which action to take. Before making each decision the bandit is shown some context. After the action is chosen a stochastic reward is revealed for the chosen action only. We propose to give a Natural Language description of the environment as context and argue that this will yield better performance than that of the non-informed agent, as is true in general when comparing contextual and non-contextual bandits, and, more importantly, near to same performance as the agent to which exact knowledge of the environment is given. The relevance of this work is twofold. On one hand, contextual multi-armed bandit problems arise frequently in important industrial applications. On the other giving language the role of context is both a very natural way of modelling the learning of human-like tasks as well as a cheap way to feed typically data avid Reinforcement Learning algorithms.

1 Introduction

Since their inception RL algorithms have been successful in applications such as marketing (Collier and Llorens, 2018), games (Tesauro, 1995), dialogue systems (Singh et al., 2002), continuous control (White and Sofge, 1992) and so on. Further, in the past few years, we have seen a surge in the use of RL algorithms in Natural Language Processing (NLP) models. There are many reasons for this recent increase in interest, including the fact that language is one of the most natural ways a human user can interact with artificial agents. Both when provided directly from the user under the form of short instructions as well as from commonly available game manuals or wiki tutorials, we have an abundance of NL information on how to achieve various tasks or describing the dynamics of a certain game. The main reason of this being that humans are themselves habitual users of this type of information. Further, the wide and cheap availability of this data is a valuable quality that should not be ignored, as in research data collection is often expensive. Put in different words, humans learn to read and read to learn and we believe that artificial agents should too. In this paper, we consider a contextual bandit setting in which the agent must choose among a prefixed number of actions and that, at the moment of choice, it is given context on the current environment under the form of a NL description of the environment itself. Many of the efforts made in the direction of integrating NL in RL agents do so in an instruction following setting or more generally speaking in the form of instructive and specific textual information. This is due to the fact that using unstructured and descriptive text requires first retrieving useful information for a given context and then grounding the information with respect to observations. Nonetheless, unstructured textual information is more abundant as it can be found in wikis, manuals, book or can be easily provided by a human user. In this paper, we consider this second type of NL text. The descriptions given as context, in fact, are obtained by combining human written sentences and are very similar to what an actual game manual could contain. This can be seen as an element of novelty also in consideration of the fact that to constrain the difficulty of problems considered, the majority of the works currently published use synthetic language (automatically generated from simple grammar and limited vocabulary). Furthermore, the language text we provide only aids the decision making process of the agent indirectly, by giving an insight on the type of the current environment which in turn influences the way a specific action is connected to the reward. We believe, in fact, that a more general type of dependence from NL could be more interesting and have more potential than a very specific study case. The performance of the agent described is compared to that of other two agents, which represent the limit cases of our base agent. The first is an agent which disposes of exact knowledge on the environment when making its choice. The second is an agent with no knowledge of the environment at all. We expect to see the performance of the base agent upper bounded by the former and lower bounded by the latter. We then consider many interesting variations of the base agent which help draw further insight on the potential of this type of integrated model. We conclude this introduction by describing more in detail the structure of this paper. In Section 2 we give a brief overview of the current state of the field and highlight the main points of reference for this work, including how it differs from them. In Section 3 we introduce our model, describing how we fused an NLP architecture together with an RL one, and the dynamics of the contextual bandit setting, namely how the action depends on the context and how these jointly result in a positive or negative reward. In Section 4 we comment on a number of variations of the base model highlighting how and why changes in the architecture of our model influence overall performance. In Section 5 we describe the results observed overall and comment on them with respect to our initial expectations. In particular, we focus our attention to critically analysing our findings and understanding why these conformed or did not conform to our expectations. Finally, in Section 6 we summarize our findings and expand on potential future works and extensions of our model.

2 Literature review

This paper stems from the picture given in (Luketina et al., 2019) of the current state of two ever-growing fields, NLP and RL, and how they are being tied together. There are many reasons for which much effort is being made in the direction of integrating them. One of these is that RL methods tend to be inefficient, requiring millions or billions of interactions, as well as generalise poorly to new tasks, even when they are only slightly different from those seen during training. In this context, language provides a natural way of aiding the learning process of an agent. In fact, common techniques to learn word representations involve co-occurrence statistics, as in (Deerwester et al., 1990) and in (Mikolov et al., 2013), and contextual word-representations using pseudo-language model objectives as in (Peters et al., 2018) and in (Devlin et al., 2018). The latter can transfer knowledge to downstream tasks that have to deal with language. Similarly, word and task-specific knowledge communicated in NL could also be transferred to sequential decision-making problems, as found in RL settings. We indeed believe that the time is right to investigate a tight integration of NL understanding into RL. In order to contribute to this movement though, one must first recognize that there are many ways of doing this. The survey done by (Luketina et al., 2019), thus serves, among others, also the purpose of introducing a useful taxonomy of the efforts made so far in integrating language in RL. One important distinction is that of separating the literature into *language-conditional RL* and *language-assisted RL*. The former refers to experiments in which the interaction with language is necessitated by the problem formulation itself, the latter to those in which language is used to facilitate learning. These two categories are not mutually exclusive, in that for some language conditional RL tasks, NLP methods or additional textual corpora are used to assist learning (Bahdanau et al., 2018) and (Goyal et al., 2019). A recent example of the first category is (Zhong et al., 2019). Here the authors demonstrate that language understanding via a reading policy learner is a promising vehicle for generalisation to new environments, which is often a challenging problem in RL. In particular, in a setting in which they procedurally generate environment dynamics and corresponding language descriptions of the dynamics, they show that the agents are able to generalise to new environments with dynamics not seen during training via reading of the NL descriptions. This first category can further be distinguished in two subcategories *instruction following* and *rewards from instruction*. In the former, the agents are presented with tasks defined by high-level sequences of NL instructions. Effective instruction following agents execute the low-level actions corresponding to the optimal policy or reach the goal specified by their instructions and can generalize to unseen instructions during testing. In the latter, NL instructions induce a reward function for RL agents or planners to optimize. This is relevant when the environment reward is not available to the agent at test time but is either given during training (Tellex et al., 2011) or can be inferred from parts of expert trajectories. With regards to the second category, instead, we can further distinguish between *language for communicating domain knowledge* and *language for structuring policies*. Examples of the former are (Eisenstein et al., 2009) and (Branavan et al., 2012). What is meant by this is we are dealing with task-relevant text which contains advice regarding the policy an agent should follow or information about the environment dynamics. Examples of the latter are instead (Andreas et al., 2016) and (Das et al., 2018). In this second type of literature, language is used to communicate information about the state and/or dynamics of an environment and thus to construct priors on the model structure or representation of an agent. The utility and potential of integrating NL in RL thus begin to become clear. In general in fact, learning is severely constrained by data efficiency due to limited or expensive environment interactions. In this context, NL provides an interesting and significantly efficient way to render RL agents more data-efficient. Further, often we dispose of human priors that would help to solve the task, these can be expressed easily and cheaply in NL. Our paper falls under the category of language assisted RL and in particular in the subcategory of language for structuring policies. Further, as mentioned in Section 1, our RL setting is that of multi-armed contextual bandit (Collier and Llorens, 2018). This is done both because we believe that including language as context is a very natural way of integrating text in an RL setting and because it allows us to consider a simpler setting than that of an MDP, in which we have to account for more dynamics. We will now spend a few words on the deep contextual multi-armed bandit setting, as in e.g. (Collier and Llorens, 2018), since this will occupy an important role in our paper. Here the authors present a deep learning framework for contextual multi-armed bandits that is both non-linear and enables principled exploration at the same time. Further, the exploration versus exploitation trade-off is tackled through Thompson sampling by exploiting the connection between inference time dropout and sampling for the posterior over the weights of a Bayesian Neural Network (Gal and Ghahramani, 2016). Interestingly, the dropout rate is learned rather than considered a hyperparameter in order to adjust the level of exploration automatically as more data is made available. (Riquelme et al., 2018) provide in-depth analysis to understand the impact of using approximate Bayesian Neural Networks in a Thompson sampling framework. This is done by benchmarking well-established and recently developed methods for approximate posterior sampling combined with Thompson sampling over a series of contextual bandit problems.Note that recent efforts have been made specifically in the direction of giving context under the form of NL, for instance in (Karampatziakis et al., 2019) and in (Misra et al., 2017). In conclusion, this review provides a clear idea that there are many different ways of integrating NL into RL. These possibilities grow quickly when considering the extent of different RL and NLP models which can be used and how they can be combined. This means that there is a fertile ground with ample space for experimentation. We restrict our application to the one described above. In the following section we begin by describing more in detail the main components of our proposed model.

3 Methods

As mentioned in the introduction, we will focus our attention on the contextual bandit setting. The contextual bandit problem is a variant of the extensively studied multi-armed bandit problem. The bandit problem models a situation in which the agent is constrained to choose one action from a possible set of actions, i.e. the *action space* A, with the intention of maximising the expected gain. In fact, after an action is taken, the agent receives a reward that represents the gain of having selected the action taken. The reward, in most cases, is not deterministic and therefore poses an extra challenge to the agent due to the intrinsic stochasticity of the problem. As the reward is only revealed for the chosen action, bandit problems involve trading off exploration, i.e. to try potentially better actions at the cost of facing uncertain rewards, and exploitation, i.e. to select actions which are known to be good. Further in a contextual bandit setting, before making each decision the bandit is shown some context $x \in X$, where X is usually referred to as the *context space*. In this case, the reward is a function of both the action and the context, i.e. the same action can lead to different rewards given a different context. The role of the context is to give some degree of information about the current dynamics of the rewards and, therefore, inform the agent. The degree of information depends on the nature of the problem. The full range of possible scenarios that it is possible to model using RL is presented in Figure 1.

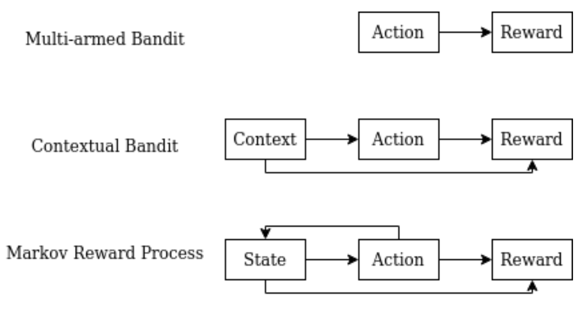

Figure 1: Different kinds of RL problems.

In particular, we imagined a simple setting in which an agent must choose one among seven weapons in order to survive the current dungeon. Note that the choice is made *before* entering the dungeon. There are five such dungeons: `desert`, `swamp`, `mountain`, `rocky plains`, `forest`. In other terms, the choice of the weapon is the action $a \in A = \{a_1, a_2, ..., a_7\}$ and the dungeon is the environment $e \in E = \{e_1, e_2, ..., e_5\}$. The context *x* is instead given by a short description of the given dungeon which is generated by randomly combining human-written sentences. The final text is designed so as to include a few dungeon-specific elements in the middle of numerous non-informative words. To avoid positional bias, the order of the sentences is also randomized within the text. The space of all contexts *X*, which is thus combinatorially large, is represented by all the possible descriptions generated in this way. It is in this sense that our agent is informed by NL. The generation of the context space is loosely inspired by the language templates for the goals and dynamics in (Zhong et al., 2019). We will now go in more detail with regards to the environment dynamics and, in particular, to the dependence of the rewards on the actions and the environments. For every given environment *e* there is an action a_e^* that will yield positive reward with certainty. Every action has at least a 0.01 probability of yielding a positive reward even if it is selected in the *wrong* context, i.e. even if it is not the optimal action a_e^* for the given environment. Moreover, there is an action that performs poorly regardless of the current environment and another one which gives a decent performance regardless of the context. In fact, the latter action is such that, given that the agent is completely agnostic of the environment, the expected reward resulting from selecting it is higher than the expected reward of selecting any other action. Note that we designed the experiment so that for every given context there exists a deterministic optimal policy. Table 1 summarises the dynamics of the environment.

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
e_1	1.0	0.01	0.01	0.01	0.01	0.01	0.3
e_2	0.01	1.0	0.01	0.01	0.01	0.01	0.3
e_3	0.01	0.01	1.0	0.01	0.01	0.01	0.3
e_4	0.01	0.01	0.01	1.0	0.01	0.01	0.3
e_5	0.01	0.01	0.01	0.01	1.0	0.01	0.3

Table 1: Probability of survival given the action choice and the context.

So far we have explained the setting we will be testing our models in. We now proceed to introduce the model we will be using. It is the union of two models that are widely used in the NLP and the RL literature. In fact, we combine the Convolutional Neural Network (CNN) model (Kim, 2014) with a modified version of the Sample Efficient Actor-Critic with Experience Replay (ACER) (Wang et al., 2016) algorithm in order to use it in the contextual bandit setting. Both of these models have shown great success in their respective fields. We call their combination NLP ActorCritic (NLPAC). The idea of using several Neural Network architectures combined is inspired by Modular Neural Networks. In particular, (Devin et al., 2017) address the renown issue of data inefficiency in Deep Reinfocement Learning through the use of modular Neural Network policies. In fact, the authors exploit this modular decomposition to train mix-and-match modules that can solve new robot-task combinations that were not seen during training. We strongly believe that this approach will be further studied in the future and will mitigate the data-inefficiency problem while allowing for more flexible RL algorithms.

We will now describe the base form of our model. In Figure 2 we report a schematic representation of this architecture. Variations of the base model will be described in the following section. The CNN is composed of three 2D convolu-

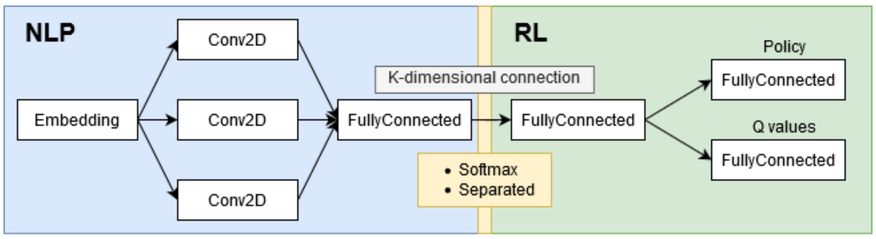

Figure 2: Graph representation of NLPAC

tional layers and one linear layer, which is characterised by $K = 5$ output neurons. In fact, given the dungeon description as an input, the CNN will output a vector $s \in \mathbb{R}^5$ of which the elements represent scores, using a softmax activation function, given to the likelihood (or not) of being in a given dungeon. The ActorCritic is instead composed of a shared series of linear layers and two final and separated linear layers for the policy and the q-values. In the following, in particular, we will consider an end-to-end model in which the output neurons of the linear layer of the CNN are the input neurons of the first linear layer in the ActorCritic, i.e. the context given to the RL agent is that of the dungeon in which it finds itself. With regards to the training process, we designed two different procedures. The first one is an end-to-end training using the RL loss to update both the NLP model as well as the RL model. In other terms, the two parts are updated using the same information. The second one is a two-step training procedure. The NLP architecture is trained using a cross-entropy loss as if this were a Supervised Learning task and the same is done for the RL architecture. It is important to underline that the optimisation step happens at the *end* of every batch of episodes. Note that in this way we are perfectly reproducing the contextual bandit setting as the training is happening simultaneously for both the NLP and the RL architectures with the label, i.e. the true current environment, becoming available only once the action is chosen. We are specifying this as the training process should not be confused with a simple SL problem plugged into an RL framework. Note that the downside of this is that since the NLP model is training *during* this process it yields a context which is not very informative in the first epochs. Further, it is possible and correct to use this second procedure as well since we designed the experiment in such a way that the true environment is revealed only after taking the action. These two training procedures correspond to the two main agents we will comment on in the following. Note that the performance of these agents is not only compared to that of the random agent but more importantly, to that of the *non-informed agent*. We expect to see the performance lower bounded by this agent and upper bounded by the *deterministically informed agent*. The latter is what we call the agent to which the context given is not a score vector $s \in [0, 1]^5$ but a one-hot-encoding of the current dungeon, i.e. the agent is given exact knowledge of the environment instead of the dungeon description from which it has to extract information on the current environment. Our metric of success thus consists in observing the performance curve of the base agents upper and lower bounded as described above and, in particular, to find that it is much closer to that of the deterministically informed agent than it is to the non-informed agent. This, in fact, would show that providing context under the form of NL, which is much easier to provide for a human user than a vector of scores, we obtain performance close to that in which exact context is given. Note that by *performance curve* we mean the curve of the rewards obtained across the training epochs.

With regards to the hyperparameters of the model, these are reported in Table 2. We empirically tested different values without experiencing significant changes in the performance, with the exception of the learning rate. The main result for the rewards during training is reported in Figure 3. The curves present in the plot were obtained in the following way. The training consists of five trials taking about 1h 30m of training for each algorithm. At the beginning of each trial, the Neural Networks are reset. This is done so we can have an unbiased estimate of the performance. We then take the average value every 320 rewards and, finally, smooth the curve.

We now comment on our findings. ACER_NLP_JustRL represents the agent that is trained with RL end-to-end, i.e. the opti-

reward (victory)	reward (defeat)	Learning rate	Gamma	Optimizer	Batch size
10	-10	0.0002	0.98	*Adam*	256

Table 2: Hyper-Parameters

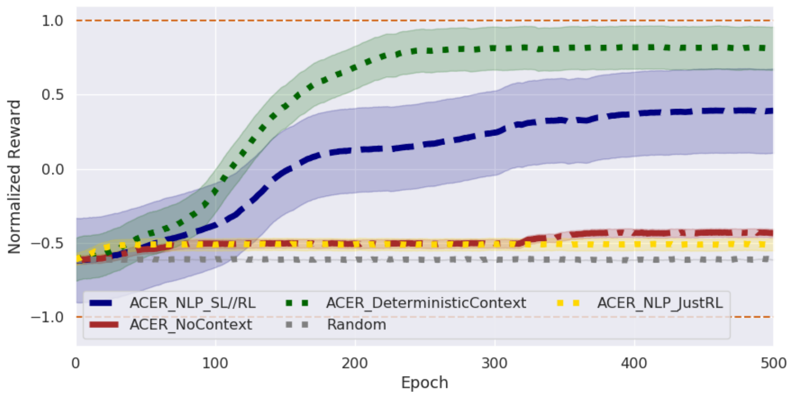

Figure 3: Rewards during training.

mizer backpropagates the information given by the RL loss through the entire architecture. ACER_NLP_SL//RL represents the agent in which the NLP and RL parts of the model are trained separately. In particular, the CNN is trained by backpropagating the information coming from the classification loss, i.e. the error made on the scores assigned to the dungeons, while the RL is trained by backpropagating the information coming from the reward observed. The activation function between the two parts of the model is the softmax. In other terms, we train the CNN as if it were just a prediction task and the ACER as if it were a pure RL task but in the contextual bandit setting, i.e. at the same time. Finally, we indicate with ACER_NoContext and ACER_DeterministicContext respectively the agent to which no context is given and that with exact knowledge of the current dungeon. These can thus be trained using only the ACER architecture as there is no NL involved. Note also that these two differ only in that in the NoContext case the input vector of zeros, whereas for the DeterministicContext the input is a one-hot encoding that deterministically identifies the type of the current environment. As we can see from Figure 3, the performance of ACER_NLP_JustRL is initially comparable to the one of ACER_NoContext. However, after a few epochs we see that ACER_NoContext manages to achieve a better performance, i.e. at that time the agent always chooses the suboptimal action. This is the action which is not optimal but which, with positive probability, yields positive reward and is the best possible choice when the agent is given no information whatsoever on the context and thus is the optimal policy for ACER_NoContext. Conversely, ACER_NLP_JustRL is not even able to reach this suboptimal policy. The problem is, in particular, that it always chooses one of the actions which is optimal in only one of the five existing environments. It is interesting to notice that the gap in performance for ACER_NLP_SL//RL when compared to ACER_DeterministicContext is quite narrow. This is a desirable finding as it means that NL can improve the learning process of an agent almost as much as when exact knowledge on the environment is given. Nonetheless, it is also important to note that this algorithm is characterised by very high variability. In fact, it is not infrequent to find that the agent is not able to reach the optimal deterministic policy. This is due to the fact that the RL part tends to overfit to the initial prediction of the NLP part and it is not always able to recover. The interesting finding is that, even for a simple setting like the one we are studying, the end-to-end model fails to learn. This is a strong call to develop specifically designed architectures that are able to better integrate the NLP part into the RL pipeline as it proves that the connection recent efforts are attempting to make requires careful exploration. The main problem is that in the initial phase of training the agent experiences many negative rewards and the information that passes between the two architectures is therefore flawed and causes the NLP architecture to begin an erroneous learning process. We also tried a Supervised Learning warm-up approach as used by (Silver et al., 2016) and (Das et al., 2017), yet the result is not significantly better. This was quite surprising given the level of success achieved in the cited studies, yet it highlights the difficulty of the task and the concrete necessity of further studies on the topic.

4 Experiments

As mentioned above we consider two variants of the base model. The first is changing the *training* procedure, i.e. we consider a variant of the ACER_NLP_SL//RL model, called ACER_NLP_SL&RL, in which we train using both the end-to-end RL procedure and the Supervised Learning one at the same time. The learning rate for the NLP architecture of the end-to-end training is set to be of one order of magnitude smaller to give more weight to the Supervised training and allowing a seamless integration of the two parts of the end-to-end pipeline, as in Table 2. Moreover, we believe that this can avoid that the initial errors of the agent are not backpropagated by the RL loss. The second is including *dropout* in our Neural Networks. We report the result for the ablations in Figure 5. Note that we maintained in this new plot also the findings plotted in figure Figure 3 so as to make it easier to compare the main model with its variations. We now comment on the result. With regards to the ACER_NLP_SL&RL variation we can observe that, although it does a relatively good job at learning in the initial phase of the training, it ends up being dominated, with a sensible gap, by the performance of ACER_NLP_SL//RL. More specifically, looking at the first part of the training we could have expected ACER_NLP_SL&RL to have a better performance than ACER_NLP_SL//RL. As the training progresses though, the performance doesn't increase enough to maintain this ordering.

This is due to the fact that the backpropagation from the RL to the NLP only results in a higher degree of exploration. This is beneficial in the first part of the training but, in the long run, it weighs down the performance of the agent.

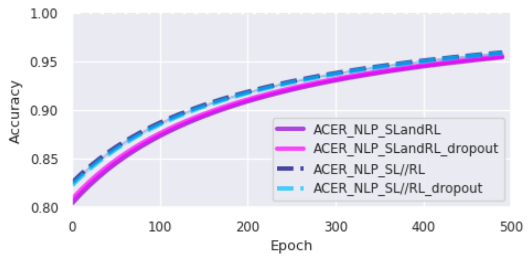

Figure 4: Accuracy of the CNN during training.

To support this we can also examine Figure 4, in which the accuracy obtained during training is reported for the two models. What we find is that ACER_NLP_SL//RL has higher accuracy than ACER_NLP_SL&RL even though the algorithms have the same NLP architecture.

The takeaway from this is thus that a more sophisticated way of integrating of NLP models in RL algorithms is required. Of all the models presented so far, we will now discuss the effect of introducing dropout. The first observation we must make is that the introduction of dropout is in general greatly beneficial for the algorithms both in terms of variability and of increase in the total reward achieved. This is due to the fact that the higher degree of exploration that the use of this technique allows the agent to explore the action space very quickly while not compromising the learning process. This is true with the exception of ACER_NLP_SL&RL, for which the introduction of dropout does not result in a significant reduction of variability or in a significant increase in performance. Beside this exception we, for instance, observe that the ACER_NLP_SL//RL with dropout presents a performance which is much closer to that of the deterministically informed agent than the model without dropout. In other terms, we find noticeably better performance for the ACER_NLP_SL//RL_dropout model than for the ACER_NLP_SL//RL model. The variability in the performance of the dropout version is also much smaller than the one without dropout. This is clearly a desirable finding, as high variability in the performance estimates is a synonym of instability in the learning process, in particular in the simple case that we are examining. However, the use of dropout is not enough to make the agent trained end-to-end with only RL achieve better performance. Finally, note that we have also tried to use the pre-trained (for classification) BERT model instead of the CNN described above, find-

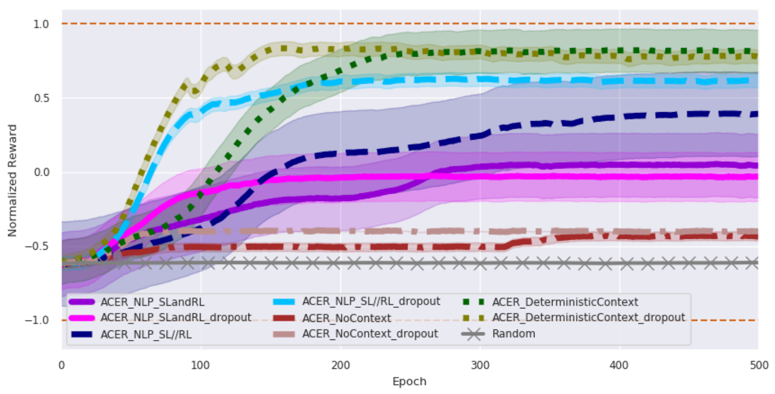

Figure 5: Rewards during training.

ing significantly worse performance. We think that this was due to the fact that the bigger the model is the slower it is to adapt to changes. This is relevant, as in control tasks fast adaptation to changes in the environment is of great importance, in particular in the initial training phase.

5 Results

In this work, we have put much effort into understanding the complicated but useful link between NLP and RL. Taking inspiration from Modular Neural Networks we designed an architecture which fuses a CNN with and ActorCritic model. We found that an end-to-end RL training process does not yield good performances, highlighting that trivially connecting two models is not enough to guarantee that the model will work well. The correct functioning of this simpler end-to-end RL would have been the desirable finding. Nonetheless, we were able to design a hybrid training process in which the unveiling of the true current environment happens once the actions is taken, thus resulting in correct implementation of the contextual bandit setting while distancing itself from a simpler but less interesting SL model plugged into an RL algorithm. With this model, we have found that giving NL text yields near to same performance as that obtained with deterministic information on the environment and can thus be considered a very good substitute. Given the cheap availability of NL this is an important finding and it means that further exploration of this new field could bring very interesting results.

6 Discussions and future work

Our model offers an insight into how an agent can be informed from both environment dynamics and NL guidelines. The integration of NL in this model is essential as it offers a significant advantage in the learning curve compared to non-informed models. RL models can be implemented with models using NLP but the combination of these two pipelines can have the effect of rendering the model significantly slower. In future work, it would be interesting to focus on the different possible techniques to combine NLP and RL in this model and thus to find which way is the most efficient. Recall that the CNN and ActorCritic models are connected through a common set of K neurons. The base model is such that $K = 5$, i.e. the number of dungeons. This means that the vector of weights corresponding to these neurons is a score vector that encodes the likelihood of being in each dungeon. This relies on the hypothesis that the number of environments is known a priori. An interesting extension that could be of wider applicability is to the case in which this information is not available. In this case, the model could be characterised by $K \neq 5$. The idea behind this is that we are not giving the model knowledge of the true number of environments and, therefore, in addition to learning the scores to assign to each dungeon once it is given the description, it should also learn the number dungeons (when $K > 5$) or to project the dungeons down to a lower dimensional space (when $K < 5$).

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48.

Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. 2018. Learning to follow language instructions with adversarial reward induction. *arXiv preprint arXiv:1806.01946*.

SRK Branavan, David Silver, and Regina Barzilay. 2012. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704.

Mark Collier and Hector Urdiales Llorens. 2018. Deep contextual multi-armed bandits. *arXiv preprint arXiv:1807.09809*.

Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Neural modular control for embodied question answering. *arXiv preprint arXiv:1810.11181*.

Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2951–2960.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine. 2017. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jacob Eisenstein, James Clarke, Dan Goldwasser, and Dan Roth. 2009. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 958–967. Association for Computational Linguistics.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Prasoon Goyal, Scott Niekum, and Raymond J Mooney. 2019. Using natural language for reward shaping in reinforcement learning. *arXiv preprint arXiv:1903.02020*.

Nikos Karampatziakis, Sebastian Kochman, Jade Huang, Paul Mineiro, Kathy Osborne, and Weizhu Chen. 2019. Lessons from real-world reinforcement learning in a customer support bot. *arXiv preprint arXiv:1905.02219*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

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. *arXiv preprint arXiv:1906.03926*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Dipendra Misra, John Langford, and Yoav Artzi. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *arXiv preprint arXiv:1704.08795*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*.

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

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

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Twenty-fifth AAAI conference on artificial intelligence*.

Gerald Tesauro. 1995. Temporal difference learning and td-gammon.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. 2016. Sample efficient actor-critic with experience replay. *arXiv preprint arXiv:1611.01224*.

David A. White and Donald A. Sofge. 1992. Handbook of intelligent control: Neural, fuzzy, and adaptive approaches.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. 2019. Rtfm: Generalising to novel environment dynamics via reading. *arXiv preprint arXiv:1910.08210*.