

Practical Models for Sequential Decision Making in Natural Language Processing and Reinforcement Learning

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Rajkumar Ramamurthy
aus
Chennai

Bonn, 2023

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen
Friedrich-Wilhelms-Universität Bonn

1. Gutachter: Prof. Dr. Christian Bauckhage
2. Gutachter: Prof. Dr. Stefan Wrobel
Tag der Promotion: 09.11.2023
Erscheinungsjahr: 2023

Abstract

This thesis focuses on *sequential decision and prediction* (SDP) tasks, comprising structured prediction (SP) and reinforcement learning (RL) tasks. These tasks are characterized by generation of sequential outputs that exhibit interdependencies among them. Notable examples include machine translation (MT), where the objective is to map variable-length input sequences to variable-length output sequences and robotic RL tasks like object grasping, where a sequence of actions must be generated to accomplish the task, with each action influencing future actions.

Contextualized representations play a vital role in making decisions at each step. Modern architectures like Recurrent Neural Networks (RNNs) have become standard tools for obtaining contextualized inputs and achieving state-of-the-art results in SDP tasks. However, RNNs demand high-compute GPU resources and are impractical for low-resource settings. Consequently, the first part of the thesis explores the application of random context encoders, specifically to investigate the performance gap compared to fully trained RNNs. It begins by showcasing the capabilities of echo state networks (ESNs), a type of RNNs, to memorize and reproduce arbitrary sequences of data, such as text, images, and videos, without requiring full training. Next, ESNs are applied to Named Entity Recognition (NER) and learning control policies using RL, highlighting their effectiveness as randomized contextualized encoders.

The thesis then shifts its focus to address the challenges of exploration and high sample complexity in RL. Specifically, two approaches that incorporate additional knowledge into the learning process are introduced. The first approach considers Novelty Search (NS), a method designed to enhance exploration and sample complexity, is considered, and a novel approach utilizing auto-encoders to learn sparse representations of agent behaviors is proposed. This approach outperforms traditional NS methods and provides a solution to promote exploration in RL tasks. Furthermore, the thesis introduces a method to construct policy networks by leveraging domain knowledge to improve transparency, modularity, and data efficiency. This decomposition of policy networks into adaptable and hand-designed components significantly reduces the number of interactions required for learning when compared to fully trained end-to-end recurrent networks.

In the final part of the thesis, we closely look at the relation between SP and RL tasks. In particular, we consider the formulation of SP as an RL task to overcome the problems of data and metric mismatch associated with training SP using supervised learning. Despite the successful application of RL in these settings, research in this direction is hindered by a lack of open-source toolkits. To mitigate this, we first propose *NLPGym*, a modular toolkit that casts typical NLP tasks as RL tasks, allowing RL algorithms to directly optimize any application-specific metric. Building upon this and utilizing large-language models (LLMs), we present an open-source library *RLALMs* that can fine-tune LLMs on arbitrary reward functions, including learned reward models based on automated metrics and human preferences. Additionally, a comprehensive benchmark to evaluate LLMs on various text generation tasks and a new algorithm has been introduced to fine-tune LLMs efficiently.

Acknowledgements

This thesis journey has been a long and delightful adventure and it would not have been as enjoyable without the support of many people including supervisors, advisors, mentors, colleagues, family and friends.

First and foremost, I would like to thank my supervisors Prof. Christian Bauckhage and Prof. Stefan Wrobel for providing the freedom for pursuing research topics that suited my interests and guiding me through this journey whenever I needed support.

I am deeply grateful for the support and encouragement of my family throughout this journey. My parents have consistently stood by my side, offering their support and encouraging me in my choices. I extend my heartfelt thanks to my wife Varsha for her support and care, without which I would not have been able to accomplish this significant achievement in my life. I would also like to express my special appreciation to my daughter Arya¹. Their belief in me and their patience during the times when I had to prioritize my studies over family moments mean the world to me, and I am truly grateful for their understanding.

I would like to thank all my colleagues at Fraunhofer who have provided me with an amazing friendly research environment. Special thanks to Thiago Bell, Maren Pielka, Lars Hillebrand, Max Lübbering and Kostadin Cvejoski for their extensive feedback on parts of this thesis.

And finally, I would like to extend my special thanks to my mentors. Firstly, I am deeply grateful to my advisor and close friend, Rafet Sifa, whose unwavering support and motivation have been invaluable throughout my PhD journey. Additionally, I am incredibly fortunate to have had the opportunity to collaborate with two remarkable researchers, Prithviraj Ammabrolu and Kiante Brantley, during the final stages of my doctoral studies. Their guidance, mentorship, and friendship have surpassed the boundaries of co-authors.

¹ As I write this section of the thesis, my daughter is sitting on my lap, playfully pressing random buttons, while I undo her keystrokes

Contents

1	Introduction	1
1.1	Exploring Randomized Models for Sequential Decision and Prediction	2
1.2	Improving Efficiency of RL using Domain Knowledge	4
1.3	Towards Practical SP using RL	5
1.4	Thesis Outline	7
1.5	Publications	8
I	Echo State Networks for Structured Prediction	9
2	Introduction to Echo State Networks	13
2.1	Basic Architecture	13
2.2	Design Choices	14
3	Echo State Networks for Cryptography	17
3.1	Motivation	18
3.2	Echo State Networks as Memories	19
3.3	ESN-Based Encryption and Decryption	19
3.3.1	Representing Data	20
3.3.2	Memorizing Data	20
3.3.3	Recalling Data	20
3.3.4	Working with “Data Chunks”	21
3.4	Experiments and Results	21
3.4.1	Security analysis	21
3.4.2	Performance	24
3.5	Conclusion	24
4	Echo State Networks for Named Entity Recognition	25
4.1	Motivation	26
4.1.1	Related Work	27
4.2	Named Entity Recognition	28
4.3	ESNs as Random Context Encoders	29
4.4	Experiments and Results	29
4.4.1	ESN Design	30
4.4.2	Evaluation on Different Embeddings	31
4.5	Conclusion	32

5	Echo State Networks for Partial Observability	33
5.1	Motivation	34
5.2	Simultaneous Perturbation Stochastic Approximation	35
5.3	Echo State Networks as Policies	36
5.3.1	Partial Observability	36
5.3.2	Policy Learning Using Echo State Networks	36
5.3.3	Tackling Exploration	37
5.3.4	Training Variants	38
5.4	Experiments and Results	38
5.4.1	Implementation details	38
5.4.2	Results	39
5.5	Conclusion	41
II	Improving Exploration and Sample Complexity in RL	43
6	Guiding Reinforcement Learning via Encoded Behaviors	47
6.1	Motivation	48
6.2	Motivating Experiments	50
6.3	Preliminaries	51
6.3.1	Markov Decision Process	51
6.3.2	Evolution Strategies (ES)	52
6.3.3	Novelty Search with Nearest Neighbors	53
6.3.4	Combining with RL	53
6.4	Novelty Bonuses via Encoded Behaviors	54
6.4.1	Training	54
6.4.2	Sparse Encoding:	55
6.4.3	Novelty Scores	55
6.5	Experimental Results	55
6.5.1	Sparsity Levels	59
6.5.2	Ablation analysis	60
6.6	Conclusion	62
7	Designing Policy Architecture with Domain Knowledge	63
7.1	Motivation	64
7.2	Background	66
7.2.1	Reacher Task	66
7.2.2	Linear MMC Networks	66
7.2.3	Non-linear MMC Networks	68
7.3	Learning a Modular Policy Network	69
7.3.1	Approach For Solving Reacher Tasks	69
7.3.2	Learning Method	70
7.4	Experimental Results	70
7.4.1	Learning Performance	71
7.4.2	Behaviors	72

7.4.3	Sample Complexity	73
7.5	Conclusion and Future Work	73
III	Practical Structured Prediction using Reinforcement Learning	75
8	NLPGym - A toolkit for training RL agents on Natural Language Processing Tasks	79
8.1	Motivation	80
8.2	NLPGym Toolkit	82
8.2.1	Tasks	83
8.2.2	Towards Interactive Learning	83
8.3	Demo Experiments	84
8.4	Conclusion	86
9	RL4LMs - Building Blocks, Baselines and Benchmark for NLG using RL	87
9.1	Motivation	88
9.1.1	Related Work	90
9.2	RL4LMs - A library to train language models using RL	91
9.2.1	Environments: Generation as a Token-level MDP	91
9.2.2	Reward Functions and Evaluation Metrics	92
9.2.3	On-policy Actor-critic Algorithms	92
9.3	NLPO: Natural Language Policy Optimization	93
9.4	General Reinforced-language Understanding Eval - GRUE Benchmark	94
9.4.1	Results on GRUE	97
9.4.2	Preference Reward Learning, Selection, and Hacking	98
9.4.3	Data Budget: Improve your Reward or Gather More Demonstration?	99
9.4.4	Practical Considerations: Which Implementation Details Matter Most?	100
9.5	Conclusion	100
10	Conclusion	101
10.1	Outlook	102
A	ESN for Partial Observability	105
B	NLPGym	107
B.1	Demo Scripts	107
B.2	Default Featurizers	109
B.3	Custom Components	109
B.4	Qualitative Analysis	112
C	RL4LMs	117
C.1	Experimental Details	117
C.1.1	Crowdworking Details	117
C.1.2	GRUE Experiment Setup	117
C.1.3	IMDB	117
C.1.4	CommonGen	126

C.1.5	CNN/DM	133
C.1.6	ToTTo	143
C.1.7	NarrativeQA	150
C.1.8	Neural Machine Translation	154
C.1.9	Dialy Dialog	158
	Bibliography	165
	List of Figures	187
	List of Tables	191
	Publications	197

Introduction

Machine learning (ML) systems have become a critical component of many facets of modern society, from powering web searches and filtering social media content to recommending products on e-commerce websites, enabling object identification and speech-to-text transcription on consumer products like smartphones and laptops. Deep learning (DL) methods have become increasingly prevalent in these applications due to their ability to learn directly from raw data without requiring hand-engineered feature extractors, which are commonly required in traditional ML systems. DL achieves this by performing a series of non-linear transformations on the raw data, with the transformation of each layer learning a more abstract representation than the previous layer. Crucially, these transformation layers are learned from the data rather than designed by human engineers. While the theoretical foundations of DL research date back to the 1980s, the widespread adoption of DL has been driven by the availability of large-scale datasets, the emergence of Graphical Processing Units (GPUs), and the development of open-source automatic differentiation libraries like Tensorflow [1] and Pytorch [2], which make training deep neural networks more accessible.

Deep learning has made significant advancements in many previously unsolved problems, thanks to its ability to identify complex non-linear patterns in high-dimensional data. While it initially gained success in computer vision, achieving impressive accuracy in image classification, object detection, and segmentation [3–6], and in speech recognition [7, 8], it has also outperformed traditional ML methods in various other fields. For instance, it has proven effective in understanding particle accelerator data [9], predicting the effects of DNA mutations on gene expression and disease [10, 11], predicting the activity of potential drug molecules [12, 13], and predicting protein structures [14, 15]. Moreover, DL has led to breakthroughs in generative models such as Generative Adversarial Networks (GANs) [16], Variational AutoEncoders (VAEs) [17] and diffusion models [18], which can create realistic images, videos, and even 3D objects [19]. It has also enabled generative artificial intelligence, where models can generate new content in various domains such as music, art, and text. For instance, Generative Pre-trained Transformer (GPT) models [20, 21] can produce high-quality natural language text, making them useful for several NLP tasks. In contrast, other generative models can create images [18, 22], videos [23], and music [24] with remarkable quality and creativity.

Based on this brief overview of the current state of ML, this thesis focuses on tasks that are inherently sequential in nature, requiring the generation of sequential data. For instance, text summarization involves generating a sequence of words that concisely summarizes a longer body of text. Similarly, in speech recognition, the objective is to convert voice data into a sequence of words. These tasks extend

beyond Natural Language Processing (NLP) and are also prevalent in other domains. For example, time-series prediction, such as weather forecasting or stock price prediction, involves forecasting values for a fixed time horizon. It is important to note that in some of these tasks, the inputs themselves are sequences. Broadly, these tasks fall under the category of *structured prediction* (SP) in which the objective is to map variable-length input sequences to variable-length output sequences. Classic examples of these include Machine Translation (MT), where a sentence from the source language is mapped to a corresponding sentence from the target language. Similarly, in sequence labeling tasks such as Named Entity Recognition (NER) or Part of Speech (POS) tagging, the input is a sentence, and the output is a label sequence.

SP possesses two distinctive characteristics. First, there exists a dependency among the output elements. For example, in NER, the outputs must adhere to specific tagging schemes such as the BIO tagging scheme [25] to identify entities spanning over multiple words. Similarly, in MT, the individual words in the output sentence must adhere to the grammar rules of the target language. In other words, SP entails generating structured outputs. Second, to incorporate this output dependence, each output at each step becomes the input at subsequent steps, meaning that the output at each step is conditioned on both the input and the partially predicted output.

Interestingly, sequential decision-making in the context of RL [26] shares similar aspects with SP. In RL, an agent interacts with an unknown environment by observing its current state and decides on actions to perform. The objective is to train an agent policy that maps states to actions, maximizing the expected cumulative reward. Similar to SP, the output in RL is also a sequence — a series of actions that can effectively solve the task at hand. For example, a robotic hand grasping an object performs a series of actions in the form of joint movements until it successfully holds the object. Moreover, since actions performed at one step impact future observations, a correlation exists between actions within a specific trajectory, similar to the structure between outputs in SP.

This thesis focuses on SP and RL tasks, collectively referred to as *sequential decision and prediction tasks* (SDP). A common theme among these tasks is that they require a model that can efficiently process sequential inputs and generate outputs one at a time to solve the given task.

1.1 Exploring Randomized Models for Sequential Decision and Prediction

SP tasks in NLP deal with the mapping of sequential inputs to sequential outputs. Traditionally, these tasks were addressed using search techniques [27]. However, the combinatorial nature of the problem made these techniques intractable, resulting in incremental solutions [28]. As a practical approach, models for SP tasks now generate outputs one element at a time while parsing the input sequence. Contextualized inputs play a crucial role in making decisions at each step. For example, in NER, the contextualized input comprises the current word, neighboring words, and optionally their predicted labels. In MT, it includes sentences from the source language and partially generated output in the target language.

A key component of these tasks is the need for effective word representations. Classic methods relied on high-dimensional features based on n-grams and suffered from generalizability issues [29, 30]. Accordingly, the notion of distributed representations [31] was proposed, leading to popular Word2Vec [32], Fasttext [33] and Glove [34] which spawned a promising area of language modeling using neural networks so called as *neural language modeling*. Although these representations captured

some global context, they did not consider the local context within a given sentence. Tackling these, Recurrent Neural Networks (RNNs) were used to learn contextual representations, leading to popular methods of word embeddings such as Flair [35], ELMo [36] and ULMFit [37] which were applied to several downstream tasks, including NER and POS tagging.

For tasks involving mapping sequences to variable-length sequences, sequence-to-sequence models [38] were proposed, also leveraging RNNs. These models employ an encoder RNN to map input sequences to a fixed-length context vector, which is then passed to a decoder RNN that generates output sequences. However, the process of encoding long sequences into a single vector and decoding from it poses challenges [39]. To overcome this, the attention mechanism was introduced to sequence-to-sequence models [39], revolutionizing NLP. It enables the decoder to selectively focus on specific parts of the encoder’s hidden states, overcoming issues with long sequences. Various forms of attention [40], extending beyond sequences [41], are widely used for selecting relevant input parts.

Such attention mechanisms paved the way for the transformer architecture [42], which is proposed as an alternative to RNNs that are computationally slower due to their sequential nature. Instead, the non-recurrent transformer architecture that employs self-attention mechanisms and fully connected networks, allows the model to attend to all inputs simultaneously, rather than processing them sequentially. This parallelization significantly speeds up the training process and allows for longer-range dependencies to be learned. The transformer architecture serves as the backbone of many state-of-the-art NLP models, such as BERT (Bidirectional Encoder Representations from Transformers) [43], GPT (Generative Pretrained Transformer) [20, 44], T5 (Text-to-Text Transfer Transformer) [45] including the recent breakthroughs in chatbot like instruction following models [46–48]. These models have achieved remarkable results in various NLP tasks, including language modeling, text classification, and machine translation, among others.

Interestingly, RL also requires contextualized inputs in some settings. Typically, the interactions between the agent and the environment are formalized using Markov Decision Processes (MDP). MDP defines transition dynamics, reward formulations, and other components. Most importantly, it assumes that the observation provided by the environment at each step satisfies *Markov property*, meaning that the current observation fully captures the entire state of the environment. In other words, basing the action solely on the current observation is sufficient for making informed decisions. However, in partially observed settings such as Partially Observed Markov Decision Processes (POMDP), the observation might not fully capture the state of the environment. In such cases, the agents must consolidate the past observations into contextualized inputs to obtain a Markov state representation which can be used for decision-making. Here too, RNNs are generally used to consolidate such states but are often more difficult to train than non-recurrent architectures [49, 50].

While these modern architectures, including RNNs, achieve state-of-the-art results in SDP tasks, they require high-compute GPU resources, limiting their applicability in low-resource settings. Consequently, this raises the question whether such computationally expensive tools are truly necessary for solving SP and RL tasks. Specifically, we would like to examine if random context encoders that do not explicitly train their recurrent connections can solve SP tasks (**RQ1**)¹.

Addressing this question, this thesis explores the application of random context encoders to SDP tasks. To achieve this objective, Echo State Networks (ESNs), a type of RNNs is considered. ESNs were initially developed as a simple alternative to conventional RNNs. This operates under *reservoir computing* paradigm, which involves connecting a randomly connected RNN known as a reservoir,

¹ RQ - Research Question

with fixed connection weights. Similarly, the connections from the input to the reservoir are also fixed. However, the output weights connecting the reservoir to the outputs are adapted for the specific task. ESNs are known for their simplicity, efficiency and ease of training, although they have been primarily successful in time-series prediction.

First, to highlight the context-capturing capabilities of ESNs, we apply ESNs to memorize sequences of various data types such as images and text. Leveraging this capability, a novel application of ESNs [51] for securing data in private cryptographic systems is proposed. Next, we apply ESNs to another SP task namely NER [52] where contextualized word representations are essential. Here, we show that reservoir states act as random contextualized representations that can practically solve the NER task. Building on these successes, in the following contribution [53], we employ ESNs to capture the context in sequential decision-making problems. Mainly, ESNs are used to learn policies for POMDP control tasks and an alternative training is proposed to train them which does not rely on backpropagation. Through these contributions, we show that ESNs are robust random context encoders that are very competitive in SDP tasks.

1.2 Improving Efficiency of RL using Domain Knowledge

In RL, the ultimate goal is to teach the agent to make the best possible decisions in order to maximize the cumulative reward over time. Unlike traditional supervised machine learning techniques, RL incorporates a feedback loop where the agent receives feedback on its actions, learns from that feedback, and adjusts its decision-making process accordingly. This allows the agent to improve its performance over time and adapt to changes in the environment.

While traditional RL using tabular methods has had some successes [54–56] in the past, it has been limited by scalability issues and has primarily been applied to low-dimensional problems. However, the adoption of DL in RL has overcome these limitations and has significantly accelerated research in this field. Specifically, DNNs have been used to approximate value functions or policy functions of agents, which are essential components of several RL algorithms. This integration of DL and RL, known as Deep Reinforcement Learning (DRL), has led to significant breakthroughs. One major breakthrough was the creation of an RL agent capable of playing Atari 2600 video games at a superhuman level [57] using only reward signals. Another significant success was the development of AlphaGO [58], a deep RL agent that defeated a human world champion in the game of Go. These achievements are remarkable and represent major milestones in the development of artificial intelligence (AI), following the historical successes of IBM’s DeepBlue in Chess [59] and Watson in Jeopardy [60] by several decades. Further successes in the game-playing RL agents include AlphaZero [58], an RL agent that can play games of Chess, Shogi and OpenAI’s Five that defeated human players in Dota 2 video game [61].

Although DRL initially gained recognition for its success in game-playing agents, it has since expanded its scope to include applications in robotics [62, 63]. By learning control policies directly from camera inputs, DRL outperforms traditional hand-engineered or learned policies from low-dimensional robot states. One notable success in this area is Dactyl, a human-like robotic hand [64] that can solve Rubik’s Cube one-handed using only tactile feedback. This significant achievement demonstrated the ability of DRL to learn complex and dexterous movements in the physical world, not just in simulation. Moreover, DRL has been applied to real-world problems in various domains that require sequential decision-making. For instance, DRL has been used in communications [65],

autonomous vehicles [66], recommender systems [67], and optimization of energy consumption [68].

Despite the successes of RL in various domains, RL encounters several challenges that hinder its application. Many sequential decision-making tasks heavily rely on sparse reward signals, which are only accessible to agents when they reach the goal state. Mainly when dealing with long-horizon scenarios, assigning temporal credit to past actions becomes a daunting task. Moreover, agents in RL environments confront the exploration-exploitation dilemma, wherein prioritizing greedy actions that offer high rewards may impede their ability to discover alternative actions that could ultimately lead to better overall outcomes. These challenges make exploration difficult, and acquiring a suitable policy necessitates a substantial number of interactions, resulting in the problem of high sample complexity in RL. Consequently, the central research question in RL arises: how can exploration and sample complexity be improved? (**RQ2**)

The second part of this thesis focuses on the research question of encouraging exploration and improving the sample complexity of RL. To address this problem, two approaches that incorporate domain knowledge into the learning process are introduced. The first approach [69, 70] considers Novelty Search (NS), a class of methods aimed at encouraging exploration. These methods define a domain-specific Behavior Characteristic (BC) that captures the behavior of agents and then encourages them to exhibit novelty behaviors. For example, BC is defined as the distance traveled for a bi-pedal locomotion task. In general, BCs are defined by domain experts, thereby using their expertise in designing the tasks. However, to compute novelty scores, these methods use neighborhood methods which do not scale well when storing a large number of BCs. Therefore, we propose learning representations of agent behaviors using auto-encoders instead of neighborhood methods. By leveraging the intuition that novel behaviors tend to produce higher reconstruction errors, novelty scores can be computed on these errors without having to look up on a huge set of observed BCs. In the subsequent work [71], intending to improve sample efficiency, we present an approach that incorporates domain knowledge to design parameter-efficient and modular policy networks. Such a policy network reduces the number of interactions required to learn the policy by a factor of ten.

1.3 Towards Practical SP using RL

SP tasks are commonly trained using supervised learning (SL) methods such as maximum-likelihood estimation (MLE). MLE aims to maximize the log-likelihood of correct sequences by training models to predict the next output element based on previously seen ground truth elements. While this approach is straightforward, it has certain limitations. Firstly, during inference, models may encounter contexts that differ from those seen during training, leading to a well-known issue called as *exposure bias* or *data mismatch*. This occurs when an incorrect output is predicted and the predicted output at each step is added to the context in subsequent steps. Essentially, there is a discrepancy in the data that the model encounters during training and inference. Secondly, SP tasks trained using this objective face a *metric mismatch* problem. When deployed to real systems, these tasks are evaluated on different metrics other than what they are trained on. For example, the generated summary is assessed for its consistency and factual correctness in text summarization systems.

One approach to tackle these mismatches is to reframe SP as sequential decision-making tasks [28, 72] and apply RL techniques to maximize application-specific metrics directly. For instance, the MT task using RL can be approached by directly maximizing ROUGE scores [73]. RL enables models to learn from their own generated sequences during training, bridging the gap between the training

and testing distributions. This exposure also allows the model to learn from mistakes and adapt its behavior accordingly.

Nowadays, DRL has been extensively applied to solving SP tasks, [74–79] in NLP. However, there is currently a lack of open-source frameworks for training RL agents and consistently benchmarking their performance on SP tasks. While the development of open-source frameworks [80–84] which have greatly accelerated research in robotic tasks, game-playing agents, similar frameworks are not available for SP tasks in NLP.

DRL has also been applied to fine-tuning large language models (LLMs) in the modern age of NLP. LLMs are trained with a *next-token prediction* objective, where they predict the next token given previous tokens. Having trained on massive amounts of data with millions of parameters, LLMs are powerful text generators that can take prompts as inputs and generate textual outputs that solve most NLP tasks with few-shot capabilities including text generation, summarization, machine translation, question answering and dialogue generation. However, text generated by LLMs may sometimes contain bias, toxicity, or harmfulness due to not being trained with direct signals of human preferences. To address this, LLMs undergo a fine-tuning step where they are trained to maximize human preferences using DRL.

Consequently, recent methods have focussed on gathering pairwise preferences from humans and training reward models [46, 85, 86], which serve as proxies for human feedback. This paradigm is referred to as *Reinforcement Learning from Human Feedback* (RLHF). RLHF is a fundamental technique for chat-based assistants like ChatGPT and GPT-4, that aligns LLMs with human preferences. Despite its achievements, RL is perceived as a very challenging technique [87, 88] and this is mainly due to the lack of availability of open-source libraries containing RL algorithms suited for NLP. This leads to a practical research problem: lack of availability of open-source frameworks in the intersection of SP and RL (**RQ3**).

Besides the lack of open-source frameworks, there are several pitfalls that contribute to the perception that RL is challenging for NLP. Firstly, training LLMs involve dealing with large action spaces. Secondly, models might exploit flaws in reward functions and learn to generate high-reward sentences that compromise fluency. Thirdly, critical implementation details that affect performance are undocumented. All these lead to another research question: can we develop stable RL algorithms for fine-tuning LLMs? (**RQ4**)

In the final part of the thesis, the focus shifts to addressing **RQ3**, which concerns the lack of open-source libraries at the intersection of SP, NLP, and RL. To tackle this issue, two toolkits *NLPGym* and *RL4LMs*, which bridge the gap between SP and RL have been presented. *NLPGym* [89] is the first toolkit that casts typical SP tasks in NLP as RL environments. The initial version of the toolkit provides environments for sequence tagging, question answering, and multi-label sequence classification. These environments adhere to standard RL interfaces, making them compatible with most RL frameworks. The toolkit has been thoroughly tested on six tasks and includes a benchmark that serves as a baseline for further research in this field. Building upon this toolkit, the subsequent work [90] presents *RL4LMs*, a modular library that allows fine-tuning of LLMs on any arbitrary reward function using popular on-policy RL algorithms. These reward functions can be based on automated metrics like BLEU [91] or learned reward models trained using human preferences. Alongside this library, we introduce a comprehensive benchmark called *Generative Reinforced-language Understanding Evaluation* (GRUE), which consists of seven text generation tasks, including summarization, translation, and dialogue generation. Additionally, to address **RQ4**, an on-policy algorithm specifically designed to handle the large action space associated with LLMs is proposed, making it easier to use in practice.

In summary, this thesis focuses on practical models for SDP tasks. The objectives include developing efficient models for solving SP tasks, improving exploration and sample complexity in RL, implementing practical open-source frameworks for SP and RL, and devising stable RL algorithms for fine-tuning LLMs.

1.4 Thesis Outline

The remainder of the thesis is divided into three parts. The first part, encompassing four chapters, explores the utilization of ESNs for SP tasks. The second part, consisting of two chapters, tackles the challenges of RL and proposes solutions appropriately. Lastly, the third part considers the intersection of SP and RL, introducing two practical frameworks that frame SP as RL. Furthermore, a preliminary benchmark of results is provided for these frameworks.

Part-I

Chapter 2 provides a short introduction to echo state networks (ESNs), delving into the relevant background and discussing design choices that greatly impact their performance.

Building upon this foundation, Chapter 3 explores the application of ESNs in memorizing arbitrary sequences of data, resulting in the development of a novel neural cryptography system. Furthermore, Chapter 4 presents the utilization of ESNs for another SP task namely NER, highlighting its practicality and effectiveness in this specific domain.

Drawing upon these two applications, Chapter 5 delves into the realm of sequential decision-making problems, investigating the use of ESNs for learning control policies in partially observable RL settings.

Part-II

In Chapter 6, the focus lies on exploring Novelty Search (NS) methods, which belong to a class of methods that incentivize the agent to engage in novel behaviors. A novel approach is introduced in this chapter, which utilizes function approximation instead of relying on neighborhood methods to calculate novelty rewards. The chapter concludes with experimental results on benchmark tasks, which suggests that this alternative approach to novelty-guided exploration is a viable option when compared to traditional NS methods.

Chapter 7 presents a methodology to enhance sample efficiency by designing modular policy networks. The chapter argues that policy networks can be divided into adaptable and fixed hand-coded components. Reducing the number of parameters in this manner can significantly reduce the sample complexity.

Part-III

Chapter 8 presents the toolkit *NLPGym* that provides simulated environments for casting typical NLP tasks as sequential decision-making tasks. Experimental results for six tasks using various RL algorithms are presented that serve as baselines for future research.

Chapter 9 focuses on the challenges of RLHF. First, it presents a new framework *RLALMs* which provides easy tools to fine-tune LLMs on arbitrary reward functions. Then, it proposes a novel on-policy RL algorithm that is well-suited for NLP tasks. Finally, a comprehensive benchmark consisting of seven generative NLP tasks is presented, which highlight crucial design choices concerning reward functions, initialization of policies and more.

1.5 Publications

This thesis is based on the following publications (in reverse chronological order) :

1. R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi and Y. Choi, “Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”, *Proceedings of International Conference on Learning Representations*, 2023, URL: <https://openreview.net/forum?id=8aHzds2uUyB>
2. R. Ramamurthy, R. Sifa and C. Bauckhage, “NLP Gym – A Toolkit for Evaluating RL agents on Natural Language Processing Tasks”, *Proceedings of Wordplay Workshop in NeurIPS 2020*, URL: <https://doi.org/10.48550/arXiv.2011.08272>
3. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Guided Reinforcement Learning via Sequence Learning”, *Proceedings of International Conference on Artificial Neural Networks*, 2020, URL: https://doi.org/10.1007/978-3-030-61616-8_27
4. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Novelty-Guided Reinforcement Learning via Encoded Behaviors”, *Proceedings of International Joint Conference on Neural Networks*, 2020, URL: <https://doi.org/10.1109/IJCNN48605.2020.9206982>
5. R. Ramamurthy, C. Bauckhage, R. Sifa, J. Schücker and S. Wrobel, “Leveraging Domain Knowledge for Reinforcement Learning Using MMC Architectures”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30484-3_48
6. R. Ramamurthy, R. Stenzel, R. Sifa, A. Ladi and C. Bauckhage, “Echo State Networks for Named Entity Recognition”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30493-5_11
7. R. Ramamurthy, C. Bauckhage, R. Sifa and S. Wrobel, “Policy Learning Using SPSA”, *Proceedings of International Conference on Artificial Neural Networks*, 2018, URL: https://doi.org/10.1007/978-3-030-01424-7_1
8. R. Ramamurthy, C. Bauckhage, K. Buza and S. Wrobel, “Using Echo State Networks for Cryptography”, *Proceedings of International Conference on Artificial Neural Networks*, 2017, URL: https://doi.org/10.1007/978-3-319-68612-7_75

Part I

Echo State Networks for Structured Prediction

In this part of the thesis, the utilization of randomized context encoders in addressing sequential decision and prediction problems is investigated, with a specific focus on the research question **RQ1**. The exploration centers around a particular type of RNN known as Echo State Networks (ESNs). An introductory overview of ESNs, including the discussion of design choices that have a significant impact on their performance, is provided in Chapter 2.

To expand upon this foundation, a novel method is proposed in Chapter 3, which utilizes ESNs to memorize arbitrary sequences of data, resulting in the creation of a neural cryptography system. Additionally, in Chapter 4, the practical and competitive application of ESNs to the Named Entity Recognition (NER) task is demonstrated.

The exploration of ESNs continues in Chapter 5, delving deeper into their potential in solving Reinforcement Learning (RL) problems with partial observability. Notably, an alternative training method for ESNs in control tasks is presented, which outperforms traditional RL methods.

Introduction to Echo State Networks

Recurrent Neural Networks (RNNs) are powerful tools in NLP for handling sequential data [38, 39, 92, 93]. However, training RNNs is inefficient due to their inherent sequential nature, particularly when dealing with long sequences. They face several challenges in capturing long-term dependencies because small error gradients fail to propagate to earlier time steps, while large gradients can cause unstable learning and hinder convergence. This issue, known as *vanishing and exploding gradients* [94, 95] has been partially addressed with the advancements of RNN variants such as Long Short Term Memory (LSTM) [96] and GRU (Gated Recurrent Unit) networks [97]. An alternative approach to designing and training RNNs has emerged through Echo State Networks (ESNs) [98] and Liquid State Machines (LSMs) [99]. These methods propose that adapting all network weights in an RNN is often unnecessary. Instead, ESNs employ a *reservoir*, a randomly connected RNN attached to an output layer. Only the weights of the connections from the reservoir to the output are adapted, while the recurrent connections in the reservoir remain fixed. This characteristic makes training the network significantly easier and more stable. This paradigm of training RNNs is currently studied under *Reservoir Computing* which is an actively growing research area with many applications and even extends to deep architectures [100].

2.1 Basic Architecture

An ESN consists of three layers: an input layer consisting of n_{in} input nodes, a randomly connected reservoir consisting of n_{res} reservoir nodes and an output layer consisting of n_{out} output nodes. The input nodes are connected to reservoir nodes and their corresponding strengths are captured in the input weight matrix $\mathbf{W}^{in} \in \mathbb{R}^{n_{res} \times n_{in}}$. Similarly, reservoir nodes are connected to each other and weights are captured in the reservoir matrix $\mathbf{W}^{res} \in \mathbb{R}^{n_{res} \times n_{res}}$. Finally, reservoir nodes are connected to output nodes through the output weight matrix $\mathbf{W}^{out} \in \mathbb{R}^{n_{out} \times n_{res}}$. A graphical representation of an ESN is illustrated in Figure 2.1.

Given this architecture, given an input $\mathbf{x}_t \in \mathbb{R}^{n_{in}}$ at time step t , first the reservoir state $\mathbf{r}_t \in \mathbb{R}^{n_{res}}$ is computed. Then, output state is computed as $\mathbf{y}_t \in \mathbb{R}^{n_{out}}$ according to the following update equations:

$$\mathbf{r}_t = (1 - \alpha)\mathbf{r}_{t-1} + \alpha f_{res}(\mathbf{W}^{res}\mathbf{r}_{t-1} + \mathbf{W}^{in}\mathbf{x}_t) \quad (2.1)$$

$$\mathbf{y}_t = f_{out}(\mathbf{W}^{out}\mathbf{r}_t), \quad (2.2)$$

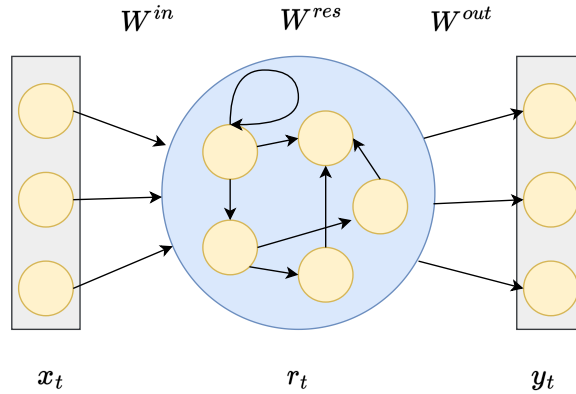


Figure 2.1: **ESN**: A sketch of an echo state network consisting of three layers: input layer, reservoir and output layer. The weights of connections from input to reservoir and inter-connections of the reservoir are generated randomly and kept fixed. Only the weights of connections from the reservoir to the output nodes are trained for the given task.

$f_{res}(\cdot)$ and $f_{out}(\cdot)$ are component-wise activation functions that apply non-linearity to reservoir and output states respectively. For $f_{res}(\cdot)$, typically, a sigmoidal function is chosen for the reservoir neurons, while for the output layer, the selection is task dependent; usually, a linear or softmax function is used. Moreover, α is the leaking rate, which controls the speed of reservoir state updates. As discussed earlier, the generation of the input W^{in} and reservoir weight W^{res} matrices is random and only the output weight matrix W^{out} is trained for a given task.

To train an echo state network, one provides a sequence of input data $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$ and desired output sequence $\mathbf{Y} = [y_1, y_2, \dots, y_T]$. Given this, we can resort to the following process:

1. Generate a randomly connected reservoir that is connected to input nodes (by generating random weights W^{in} and W^{res})
2. Run the input sequence \mathbf{X} through the ESN and collect sequence of output states $\mathbf{Y}' = [y'_1, y'_2, \dots, y'_T]$ using update equations (2.1),(2.2)
3. Find optimal weights W^{out} by minimizing a loss function $\mathcal{L}(\mathbf{Y}, \mathbf{Y}')$

2.2 Design Choices

While ESNs may give the impression of being randomly generated, in practice, their performance is significantly influenced by various design choices. Specifically, the weights W^{in} and W^{res} are not generated entirely random but based on specific hyperparameters, which will be briefly discussed.

Size of reservoir The reservoir in ESNs serves two essential functions. Firstly, it functions as a memory component, enabling the incorporation of temporal context crucial for sequential tasks. Secondly, it acts as a non-linear expansion of inputs, akin to kernel methods, effectively mapping the input to a higher-dimensional space where inputs become more readily separable, thereby improving

classification tasks. Consequently, the choice of reservoir size, denoted as n_{res} , is important. In general, a larger reservoir size allows for a broader temporal context, leading to improved performance.

Sparsity of connections It is recommended to enforce sparsity in both the connections from the input to the reservoir and the interconnections within the reservoir itself. This involves assigning a substantial number of elements in \mathbf{W}^{in} and \mathbf{W}^{res} to zero. Typically, a fixed connectivity ratio is chosen (e.g., 10% of input connections), where each input node is randomly connected to only 10% of the reservoir nodes. In general, the reservoir inter-connections are made sparser compared to the input connections.

Scaling of non-zero elements As discussed earlier, the matrices \mathbf{W}^{in} and \mathbf{W}^{res} are sparsely generated. However, the generation of non-zero elements is crucial for proper learning in ESNs. Generally, these non-zero elements are randomly generated from a symmetric uniform distribution within a specific value range. For example, the elements of \mathbf{W}^{in} are uniformly generated from the interval $[-a, a]$, where a represents another hyperparameter. The input scaling parameter a controls the level of influence the current input has on the reservoir state.

Spectral Radius One of the most crucial hyperparameters in an ESN is the spectral radius of the reservoir matrix \mathbf{W}^{res} . The spectral radius, denoted as $\rho(\mathbf{W}^{res})$, represents the maximum absolute eigenvalue of the matrix. In order for an ESN to function effectively, the reservoir should exhibit a property known as the *echo state property*, which ensures that the reservoir state depends solely on the input history rather than the initial reservoir conditions. To satisfy this property, it is generally recommended to set the spectral radius such that $\rho(\mathbf{W}^{res}) < 1$, as larger values can lead to chaotic reservoir behavior, violating the echo state property. In practice, a random matrix \mathbf{W}^{res} is generated, and its spectral radius $\rho(\mathbf{W}^{res})$ is computed. Subsequently, \mathbf{W}^{res} is divided by $\rho(\mathbf{W}^{res})$, resulting in a matrix with a unit spectral radius. The desired spectral radius can then be achieved by scaling \mathbf{W}^{res} accordingly based on the specific task requirements. As a general guideline, a higher spectral radius is suitable for tasks that require longer memory, while a smaller spectral radius is more appropriate for tasks that only necessitate a shorter history. The spectral radius and input scaling of \mathbf{W}^{in} complement each other and should be tuned in tandem based on the task at hand.

Leaking Rate Lastly, the leaking rate α in equation (2.1) controls the rate at which the reservoir updates occur. A higher leaking rate leads to faster changes in the reservoir states. When determining the appropriate value for α , it is advisable to consider the rate at which the input \mathbf{x}_t or output \mathbf{y}_t changes. Alternatively, setting a very low leaking rate can extend the short-term memory capabilities provided by ESNs.

For a more comprehensive guide on how to set such hyperparameters, readers are encouraged to refer to the review in [101].

Echo State Networks for Cryptography

Given the introduction of echo state networks (ESNs) in Chapter 2, this chapter focuses on *sequential decision and prediction problem*. In particular, a structured prediction task of generating sequences is considered. However, rather than focusing on generating novel sequences, the primary objective is to investigate the efficacy of echo state networks in memorizing and reproducing arbitrary sequences of data, such as images, texts, and videos. While ESNs have been successfully applied to time-series prediction [101, 102], the aspect of sequence memorization remains relatively unexplored.

Hence, in this work, we examine files of various formats, such as images, texts, and videos, which are inherently stored as sequences of bytes. Our focus lies in determining whether echo state networks (ESNs) have the ability to memorize such sequences and accurately generate them in an auto-regressive manner. This research problem holds significant interest for several reasons. Firstly, ESNs are randomized networks wherein recurrent connections are kept fixed and not explicitly trained for the given task. Only the output weights are trained for the task at hand. Consequently, the question of whether this type of training is adequate for sequence memorization is quite intriguing. Secondly, memorizing arbitrary byte sequences is a challenging problem, unlike textual data. Specifically, in textual data, models can exploit the statistical co-occurrences of words to predict the likelihood of certain words following a given context. However, such patterns are harder to obtain for byte sequences in images and videos, as the ordering can also be somewhat arbitrary.

First, we show that ESNs can indeed memorize short sequences of data with appropriate reservoir sizes. To achieve this, we train the output weights of the ESN to predict the subsequent byte in a sequence based on its preceding elements. By employing an auto-regressive generation approach, the ESN can sequentially generate the memorized sequence, reconstructing the original sequence one element at a time. However, as the sequence length increases, the computational demands become impractical when employing a larger reservoir. To address this challenge, we propose a divide-and-conquer strategy wherein the sequence is partitioned into smaller chunks, and each chunk is assigned a separate reservoir to memorize its corresponding chunk of data.

This ability to memorize sequences is subsequently leveraged to introduce a novel application in the field of cryptography. Specifically, we propose utilizing this memorization capability to enhance data security through a private key cryptography system. Given that ESNs can memorize data precisely, we exploit the trained output weights and ESN settings as private keys for encryption. To assess the security of this proposed system, we examine its fundamental properties as a cryptography system and demonstrate its resilience against common security attacks.

In summary, this contribution demonstrates the ability of ESNs to effectively memorize sequences, which is then harnessed for developing a novel and scalable cryptographic system.

This chapter is based on the following publication [51].

- R. Ramamurthy, C. Bauckhage, K. Buza and S. Wrobel, “Using Echo State Networks for Cryptography”, *Proceedings of International Conference on Artificial Neural Networks*, 2017, URL: https://doi.org/10.1007/978-3-319-68612-7_75

The original concept of employing echo state networks (ESNs) for cryptography was proposed and fully implemented by Rajkumar Ramamurthy. Rajkumar Ramamurthy conducted the initial experiments, including implementing, evaluating, and analyzing the security properties. The co-authors of the publication participated in the initial discussions and contributed to shaping the content of the publication. The paper itself was written by Rajkumar Ramamurthy and revised by all co-authors.

The rest of the chapter is organized as follows: First, a brief motivation for the considered problem is presented in Section 3.1. Next, Section 3.2 describes how ESNs can be used as memories. Section 3.3 presents the proposed encrypting scheme describing data representation and other practical aspects. Finally, Section 3.4 presents the experiments confirming that our approach satisfies essential cryptographic properties such as diffusion and confusion.

3.1 Motivation

The field of neural cryptography, a branch of cryptography, focuses on employing artificial neural networks for encryption and cryptanalysis. Initial studies explored cryptographic systems based on recursive autoencoders, demonstrating that feed-forward networks trained using backpropagation can encrypt plaintext messages by utilizing activation patterns of hidden layer neurons [103]. Subsequent research introduced key-exchange systems where coupled neural networks synchronize to establish shared secret keys [104]. Although the original approach had certain vulnerabilities [105], recent studies have shown that modern convolutional interacting neural networks can effectively safeguard communication against eavesdroppers [106]. Another popular concept involves combining chaotic dynamics and neural networks [107–111]. For instance, chaotic neural networks have been utilized for image encryption, demonstrating high security, and chaotic hopfield networks have been shown to generate random binary sequences for text encryption.

Based on this brief survey, the proposed approach in this contribution presents a novel idea for neural cryptography, employing a hybrid strategy that leverages chaotic dynamics and the deterministic outcome of a training procedure. Specifically, the proposal is to utilize echo state networks [98] for both encryption and decryption. In the conventional scenario where Alice and Bob exchange messages and seek to safeguard their communication against eavesdropping by Eve, it is assumed that they possess identical copies of an echo state network whose internal states evolve based on a non-linear dynamical system. To encrypt a message (such as text or an image), Alice inputs it into her copy of the network and trains the output weights such that the input is reconstructed. She then transmits these output weights to Bob, who employs them in his copy of the network to regenerate the message. Eve may intercept the communicated output weights, but she cannot decipher the message without knowledge of the corresponding echo state network’s structure, input weights, and internal weights.

Experiments conducted with this private-key or symmetric cryptography system demonstrate that this approach is easy to use, efficient, scalable and secure.

3.2 Echo State Networks as Memories

As discussed earlier in Chapter 2, ESNs follow the paradigm of reservoir computing where a large reservoir of recurrently interconnected neurons process sequential data with only minimal adaptation of weights. Recalling that the central idea is to randomly generate weights $\mathbf{W}^{in} \in \mathbb{R}^{n_{res} \times n_{in}}$ between input and reservoir neurons as well as weights $\mathbf{W}^{res} \in \mathbb{R}^{n_{res} \times n_{res}}$ between reservoir neurons. Only the weights $\mathbf{W}^{out} \in \mathbb{R}^{n_{out} \times n_{res}}$

Given this setup, we would desire to train ESN to memorize a given sequence of data $X = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_T]$. The main idea is to train ESN to predict the next item at \mathbf{x}_{t+1} in the data given the current item \mathbf{x}_t . To this end, we collect the desired one-step offsetted target sequence as $Y = [\mathbf{y}_1, \mathbf{y}_2, \dots, \mathbf{y}_T]$. In order to train, one first feeds the training sequence X into the reservoir and collects corresponding states at each time step into a sequence as $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \dots, \mathbf{r}_T]$. By gathering the sequences Y and \mathbf{R} into matrix forms $\mathbf{Y} \in \mathbb{R}^{n_{out} \times T}$ and $\mathbf{R} \in \mathbb{R}^{n_{res} \times T}$ respectively, the least squares method allows computing optimal output weight matrix \mathbf{W}^{out} as follows:

$$\mathbf{W}^{out} = \mathbf{Y}\mathbf{R}^T (\mathbf{R}\mathbf{R}^T + \beta\mathbf{I})^{-1} \quad (3.1)$$

where β is a regularization constant. However, for a good practical performance, the scale a of \mathbf{W}^{in} and the spectral radius ρ of \mathbf{W}^{res} have to be chosen carefully. Together with the leaking rate α , these parameters are tuned according to general guidelines discussed in Section 2.2.

An entire input sequence (e.g. a text file) can therefore be stored in- and retrieved from the reservoir, provided the reservoir is large enough. Hence, the idea in this contribution is to produce an echo state network with a large reservoir and to train it to memorize an input sequence. Once the training is complete, the network can run freely to (re)generate the memorized sequence.

In Section 3.3, we delve into further details of representing sequential data so that ESNs can train them and utilize them for cryptography purposes.

3.3 ESN-Based Encryption and Decryption

We consider the traditional cryptographic scenario where Alice and Bob aim to secure their communication against Eve's eavesdropping. Using a *secret key*, Alice converts her messages, known as *plaintexts* into encrypted messages known as *ciphertexts*. She then sends the ciphertexts to Bob who uses the same secret key to convert them back into plaintexts.

In this scenario, the idea is to “memorize” a given message using an echo state network at one end of a communication channel and to “recall” it at the other end using the same network. If Alice and Bob possess an identical copy of the network, Alice can train it to memorize the data and transmits only the resulting weights \mathbf{W}^{out} over the insecure channel. Upon receiving these weights, Bob plugs them into his copy of the network and runs them to reconstruct Alice's message. In essence, the weight matrices \mathbf{W}^{in} and \mathbf{W}^{res} and leaking rate α of the echo state network constitute the secret key of the proposed cryptographic system. Without having access to this key, Eve can not decipher the transmitted ciphertext \mathbf{W}^{out} . In practice, it is sufficient to communicate the random seed and other

global hyperparameters discussed in Section 2.2 as the secret key, using which the weight matrices \mathbf{W}^{in} and \mathbf{W}^{res} can be generated at the other end of the communication channel.

3.3.1 Representing Data

In the practical implementations of the above scheme, byte-level representations of messages are considered. This allows for flexibility and broad applicability because, in the memory of a computer, texts or images are ultimately represented as a byte-stream after all. To further increase flexibility, each byte is represented as a “one hot” encoding of 256 possible values; therefore, each byte is a 256-dimensional binary vector.

3.3.2 Memorizing Data

Given an input data which is represented as a byte-sequence $\mathbf{B} = [b_1, b_2, \dots, b_T]$, an echo state network is trained to memorize it as follows: First, a dummy byte b_0 is appended at the beginning of the original sequence \mathbf{B} to make the later recall process independent of the value of the original first byte in the sequence. Second, the resulting sequence is encoded to obtain $\mathbf{H} = [h_0, h_1, \dots, h_T]$ where each h_i is a binary vector of length 256. With this, input and desired output sequences to train ESN are set as follows:

$$\mathbf{X} = [h_0, h_1, \dots, h_{T-1}] \quad (3.2)$$

$$\mathbf{Y} = [h_1, h_2, \dots, h_T] \quad (3.3)$$

where the indices of the vectors in sequences \mathbf{X} and \mathbf{Y} differ by one time step. Given an echo state network with input weights \mathbf{W}^{in} and reservoir weights \mathbf{W}^{res} , appropriate output weights \mathbf{W}^{out} according to equation (3.1) are computed by iterating on equations (2.1) and (2.2).

3.3.3 Recalling Data

Once \mathbf{W}^{out} has been computed, it can be plugged into an identical copy of the echo state network at the other end of the communication channel. This network can then auto-regressively regenerate the encoded message one element at a time. To this end, the same dummy byte b_0 is considered and “one hot” encoded to obtain $\mathbf{x}_0 = h_0$. Using this as the first input to the network, the system is run according to equations (2.1) and (2.2) to obtain the output of the network \mathbf{y}_0 , which is fed as input again to the network until all the elements are generated.

At each time step, the network output \mathbf{y}_t is obtained, which is not necessarily a binary vector, as a vector of probabilities for different bytes. This vector is converted to a one-hot vector of 1s and 0s based on the byte that has the maximum probability. The resulting binary vector is then used as the input \mathbf{x}_{t+1} for the next iteration of the network. Moreover, the binary vectors obtained in each step are decoded into byte sequences \mathbf{S} , which is precisely the original sequence \mathbf{B} memorized by the echo state network.

parameter	value
chunk size m	200 (reservoir size n_{res} is chosen as $0.95 \times m$)
leaking rate α	0.07
spectral radius ρ of W^{res}	1.0
input scaling a of W^{in}	0.5
random seed	randomly chosen
input connectivity	input neurons are connected to 30 % of reservoir neurons
reservoir connectivity	reservoir neurons are connected to 30% of reservoir neurons
activation function f	logistic for the reservoir and softmax for the output

Table 3.1: **ESN configuration:** An overview of hyperparameters that are used for the security analysis of the proposed cryptography system. The input sequences are split into chunks of $m = 200$ bytes and reservoir size is set to $0.95 \times m$

3.3.4 Working with “Data Chunks”

As the size T of the data sequence increases, the size $n_{res} \in \mathcal{O}(T)$ of a reservoir that can memorize it increases, too. This makes the matrix multiplications $W^{res} \mathbf{r}_t$ required for the network’s state updates expensive. In fact, the total cost for T internal updates amounts to $\mathcal{O}(T^3)$. To reduce this cost, a “divide-and-conquer” strategy is adopted in which the data is split into chunks of size m and a small reservoir is employed to memorize each chunk. In this case, cost of the reservoir updates for each chunk is $\mathcal{O}(m^3)$. Consequently, for an entire sequence, i.e. for $\frac{T}{m}$ chunks, efforts reduce to $\mathcal{O}(\frac{T}{m} \times m^3) = \mathcal{O}(Tm^2)$. For a fixed chunk size m , the complexity of reservoir updates grows linearly in the size of the sequence T .

Also, for longer sequences, it is impractical to compute closed-form least squares as in equation 3.1 and as a result, splitting them into small chunks is inevitable. Finally, with this approach, the ciphertext is plainly just the concatenated matrix of W^{out} of all the small reservoirs used to encrypt each chunk of data.

3.4 Experiments and Results

In our practical experiments, we found that the proposed usage of echo state networks can indeed memorize and perfectly recall different types of data such as texts, images, audio files, videos, archives, etc. In this section, we present results obtained from different kinds of security analysis of our cryptography scheme. For our experiments, we used the same hyperparameters which we have listed in the Table 3.1.

3.4.1 Security analysis

Any cryptography system should be robust against common types of attacks such as brute force attacks, known-plaintext attacks, and ciphertext-only attacks. A brute force attack is an attack in which an

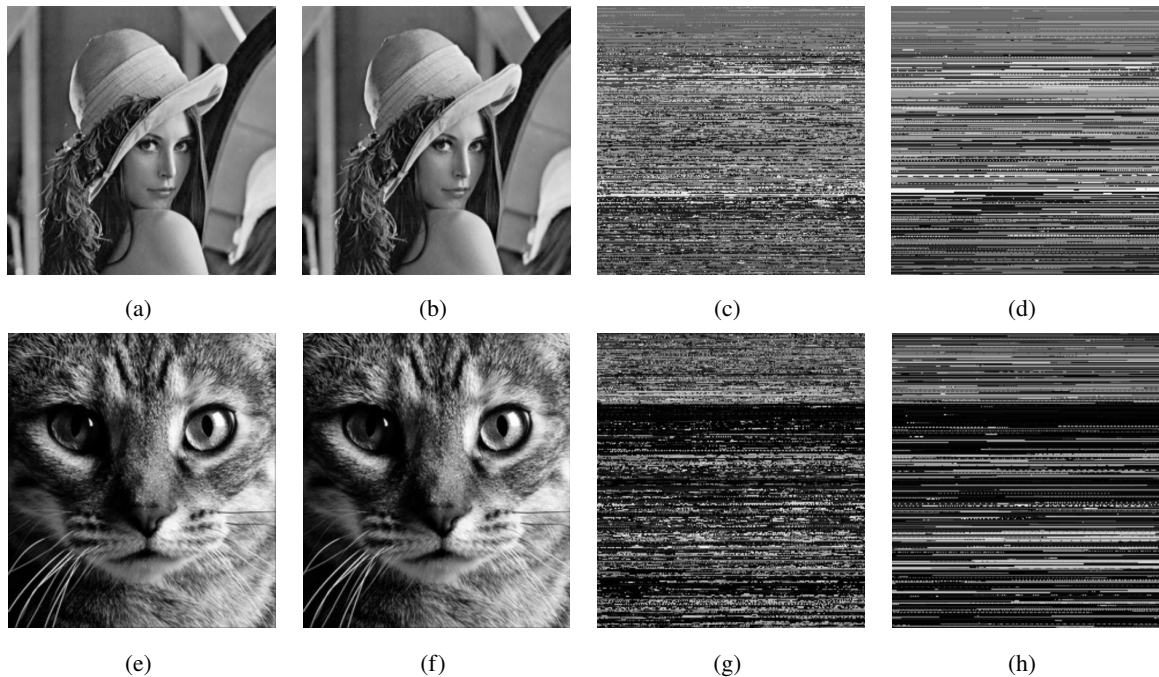


Figure 3.1: **Key sensitivity:** (a),(e) original images; (b),(f) decrypted images using the same key (echo state network) as used for encryption; both decrypted images are identical to the originals; (c),(d),(g),(h) decrypted images using slightly modified keys, i.e. slightly modified echo state networks; here, all decrypted images differ considerably from the original images.

attacker attempts to find the keys of the system through trial and error. It is evident from the Tab. 3.1 that the key space of our proposed system is very large and most of the parameters are unbounded. This renders brute force attacks extremely time-consuming and practically infeasible.

Brute force attacks Figures 3.1(a) and (e) show two original images, one of which (Lena) was given as a tiff file, the other (cat) as a png file. Both were encrypted and decrypted using the same echo state network. Decryption produced the images in Fig. 3.1(b) and (f), which are identical to the original ones. However, when decrypting with networks with slightly modified parameters, i.e. when using slight variations of the secret key, we obtained useless images as shown in Fig. 3.1(c), (d), (g), and (h). These results are prototypical and show that the system is susceptible to the secret key. This makes it robust against brute force attacks because decryption is only possible if all the parameters of the secret key are set precisely.

Known-plaintext attacks Known-plaintext attacks are ones where an attacker has access to an example of a plaintext (a message) and a corresponding ciphertext (a weight matrix \mathbf{W}^{out}) and attempts to crack the secret key via a comparative analysis of changes between them. For instance, by analyzing changes in the ciphertexts of images that differ by just a few pixels, it might be possible to obtain part of the mapping involved in encryption. Figures 3.2(a) and (f) show original images and Figs. 3.2(c) and (h) show slightly distorted versions where 1% of the pixels were randomly changed. The corresponding encrypted images (matrices \mathbf{W}^{out}) are visualized in Fig. 3.2(b), (g), (d), and (i). Only

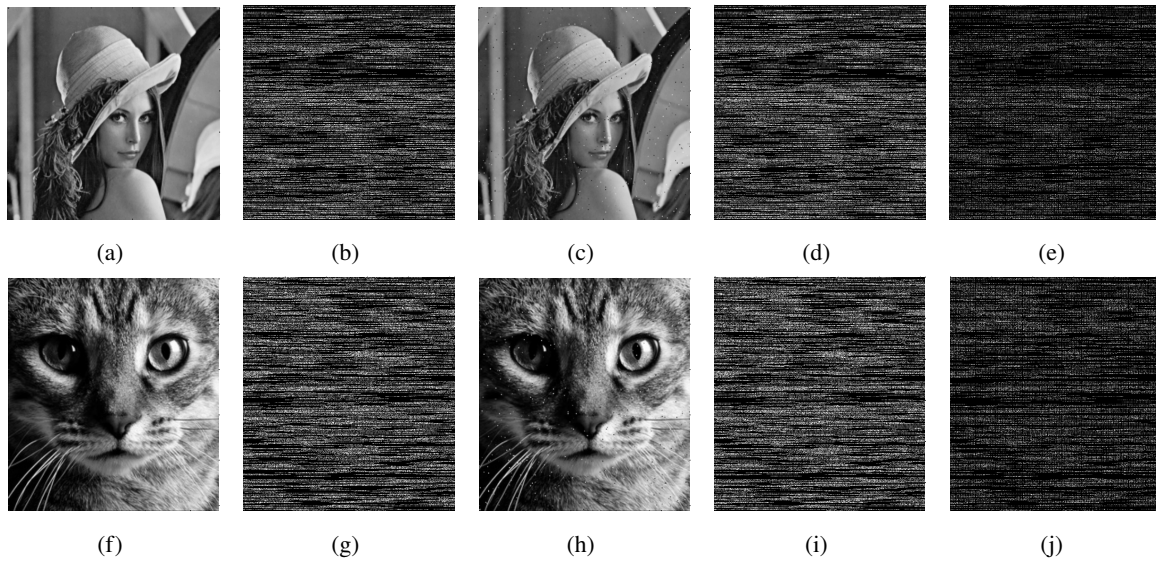


Figure 3.2: **Plaintext sensitivity**: (a),(f) original images; (b),(g) encrypted images; (c),(h) original images with 1% of their pixels randomly distorted; (d),(i) encryptions of the modified images; (e),(j) difference between encrypted original and encrypted modified images (33.22% and 37.78%, respectively).

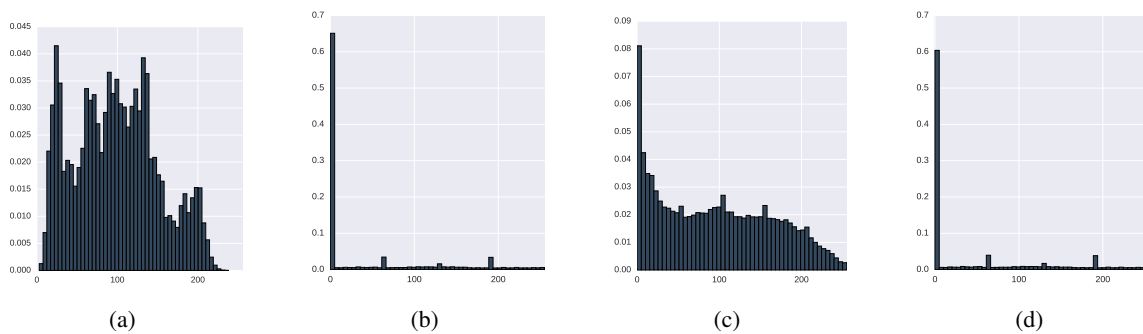


Figure 3.3: **Ciphertext sensitivity**: (a),(b) plaintext distribution of the Lena image and corresponding ciphertext distribution; (c),(d) plaintext distribution of the cat image and corresponding ciphertext distribution. Since the ciphertext distributions are almost identical, frequency analysis is difficult and the system is robust against ciphertext-only attacks.

minor changes in the plaintext led to considerable changes in the ciphertext; these differences are visualized in Fig. 3.2(e) and (j) and amount to about 35%. Thus, our system is sensitive to slight modifications of the plaintext and therefore renders known-plaintext attacks very difficult.

Ciphertext-only attacks In ciphertext-only attacks, an attacker has access to a set of ciphertexts however has some knowledge about the statistical distribution of plaintexts. Using frequency analysis of ciphertexts, for instance, exploiting the fact that “e” is the most frequent character in English texts, one can map the most frequent parts in ciphertext to corresponding plaintexts. Figure 3.3 shows frequency distributions for the plaintexts and ciphertexts of the images “Lena” and “cat”. Although the plaintext distributions of the two images differ, their ciphertext distributions are very similar. From

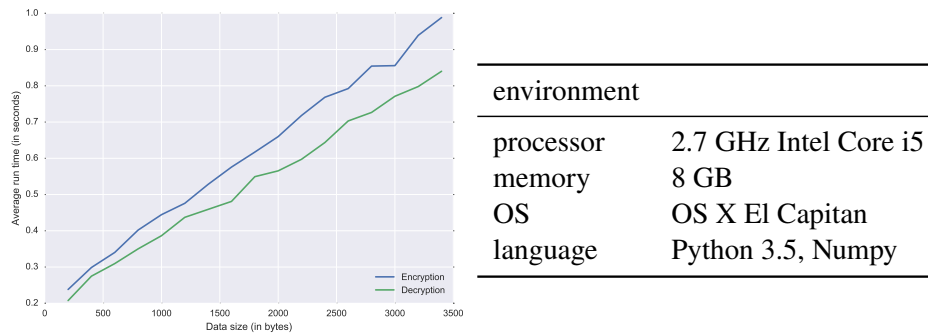


Figure 3.4: **Encryption and Decryption run times:** Run times for encryption and decryption for different message sizes. It is observed that run times grow linearly with data sizes indicating the scheme is both practical and scalable

these distributions, it is evident that most of the elements ($\approx 50\%$) in the ciphertext (W^{out}) are zero and that the non-zero elements are uniformly distributed. Thus, frequency analysis will be ineffective, and the proposed system is robust against ciphertext-only attacks.

Diffusion and Confusion Diffusion and confusion are the two fundamental properties of a good cryptography system, according to Shannon [112]. A system that has the diffusion property is one where a slight change in either plaintext or key causes a significant change in the ciphertext. A system with the confusion property is one where the mapping between plaintext and ciphertext is complex. Our experimental results indicate that the proposed system has both of these properties.

3.4.2 Performance

To assess the runtime performance of our proposed system, we determined average encryption and decryption times for messages of different sizes. Our results are presented in Fig. 3.4. For instance, encrypting and decrypting a 3KB message took less than one second each and runtimes were found to grow linearly with the message size. Our approach, therefore, is scalable and practical for real-time applications.

3.5 Conclusion

In this work, we demonstrated that echo state networks can memorize and reproduce data sequences despite only training a part of the weights. This capability is then utilized for proposing a novel neural cryptography system. Through extensive analysis, the proposed system is found to be robust against common security attacks and satisfies the essential cryptographic properties of diffusion and confusion. Furthermore, our method is simple, scalable, and suitable for real-time applications.

Echo State Networks for Named Entity Recognition

In the broader context of the thesis, this chapter delves into research question **RQ1**, which explores the application of ESNs as random context encoders for structured prediction tasks. In the previous chapter (Chapter 3), we focused on the structured prediction task of memorizing arbitrary data sequences. The fact that the generated output sequence exhibits a structure makes this task fall under the domain of structured prediction. More specifically, for the memorization task, the model needs to precisely generate the original sequence itself. Our findings highlight the efficacy of ESNs, despite their randomized recurrent nature, in utilizing reservoir states as random contexts for accurately recalling the complete data in an autoregressive manner.

Further delving into the subject, we explore the application of ESNs in yet another structured prediction task. Specifically, our focus is on Named Entity Recognition (NER), where the objective is to predict a sequence of labels for a given sequence of words. State-of-the-art approaches for tackling this task typically rely on Recurrent Neural Networks, such as LSTMs with word embeddings as their inputs. In the current era of transformer-based large language models, the general approach involves fine-tuning BERT-like models, wherein the word representations are fine-tuned further specifically for the NER task. Notably, NER benefits from contextualized word representations that encompass the context of the current word.

Nevertheless, RNNs often exhibit slower training times, while transformers require more computational resources. Consequently, these networks may not be practical in scenarios with limited resources, both in terms of hardware and data. This raises an intriguing question: Can ESNs also tackle this task, and if so, what is the performance gap between LSTMs and randomized encoders like ESNs?

With this objective in mind, we employ ESNs solely as random context encoders, utilizing word embeddings as inputs. In essence, we feed the word embeddings as input and extract the corresponding reservoir states as contextualized representations. Our findings demonstrate that these representations derived from ESNs effectively capture the necessary context to perform NER at a competitive level, all while significantly reducing training times.

In conclusion, this contribution strongly emphasizes the practicality and efficacy of ESNs as powerful context encoders, mainly when applied to sequence tagging tasks like NER. The demonstrated success of ESNs in the context of NER further solidifies their standing as versatile and reliable tools for

tackling a wide range of structured prediction problems.

The chapter is based on the publication [52].

- R. Ramamurthy, R. Stenzel, R. Sifa, A. Ladi and C. Bauckhage, “Echo State Networks for Named Entity Recognition”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30493-5_11

The concept of utilizing ESNs as contextual encoders was proposed by Rajkumar Ramamurthy. The core implementation of the ESN is developed by Rajkumar Ramamurthy which was then adapted to the task by a co-author. The set of experiments was outlined by Rajkumar Ramamurthy which was then carried out by the co-author. The resulting paper was written entirely by Rajkumar Ramamurthy and the co-author, which is then revised extensively by all other authors. All the authors participated in the discussion during the course of this work.

The rest of the chapter is organized as follows: First, a brief motivation for using word or sentence representations is presented in Section 4.1, followed by a brief literature review of randomized neural networks. Section 4.2 presents the task of Named Entity Recognition. In the subsequent Section 4.3, we present the approach of applying ESNs as random context encoders. Section 4.4 discusses the detailed experiments on the performance and practicality of ESNs on NER tasks along with discussion on crucial hyperparameters.

4.1 Motivation

Natural Language Processing (NLP) consists of a broad range of tasks, including sentence labeling [113, 114], sentence classification [115], semantic similarity question answering [116], and natural language inference [117, 118]. A crucial precursor to all these tasks is to obtain an appropriate representation of lexical units such as words, sentences and documents. These representations are then considered as features for training machine learning models. For instance, in a sequence tagging/labeling task, each word in a given sentence needs to be assigned a linguistic tag (eg. a named entity or a part of speech). Therefore, to train any ML system for solving this, each word should be converted to a real-valued vector that captures both context and semantics. Similarly, for sequence classification or semantic clustering, the task boils down to obtaining compact representations of sentences.

Classically, approaches to obtain word embeddings deal with latent-semantic analysis (LSI) [119] on term-document matrices using Singular Value Decomposition (SVD) [119] or GloVe [34] that uses word co-occurrences, context window-based word representations such as Word2Vec [32]. To make them robust against morphological variations and misspelled words, FastText [33] extends the idea of Word2Vec to include sub-word information. More recent methods [36, 120] train a bidirectional language model using LSTMs that can predict future or past words/characters conditioned on history. With these language models, the representation of any given word is obtained by gathering activations of hidden layers.

Similarly, traditional approaches to obtain sentence representations use bag-of-words (BOW) or term-frequency-inverse document frequency (TF-IDF) or LSI [119]. Although simple and effective, they ignore word ordering and suffer from high dimensionality. More recent methods focussed on learning compositional operations that map word vectors to sentence vectors using recurrent neural networks [96], recursive neural networks [121] and convolutional networks [122, 123]. However,

most of these methods are supervised in the sense that they produce representations specific to a particular task. However, it is desired to obtain generic sentence representations that work across various downstream tasks. Consequently, this led to Doc2vec [124], Skip-thought [125] models learning sentence vectors by extending on the idea of Word2Vec to sentences. Interestingly, InferSent [126] showed that it is possible to obtain generic representations by training a sentence encoder in a supervised fashion for natural language inference (NLI). These methods fall under the paradigm of *feature-based* methods as they provide general representations of words/sentences which can be used as features for downstream tasks.

In recent years, an alternating line of work has emerged which allows the training of large language models in an unsupervised fashion and then fine-tuning downstream tasks. These methods are referred to as *fine-tuning based* approaches and the most crucial aspect is that only a few parameters need to be learned from scratch. Representative approaches of these include UlmFit [37], which first trains a language model based on bidirectional LSTMs and this language model is then further fine-tuned on a target task. However, these language models are restricted in a way since RNNs such as LSTMs can capture only a short range of dependencies and are inefficient due to recurrent processing. To tackle this, Transformer based architectures which replace recurrent layers with self-attention layers, are now state-of-the-art in learning generic representations. Based on transformer architectures, BERT [43] learns representations by pre-training them on masked language modeling and next-sentence prediction tasks.

Extending on BERT architectures, sentence transformers [127] enrich sentence representations using contrastive loss functions for semantic similarity comparisons. Another crucial aspect of *fine-tuning based* models is that they require very minimal architectural adaptations to downstream tasks; addition of just a linear or multi-layer perceptron to a pre-trained model is sufficient to train further for any given downstream task.

Contrary to all these works on sentence encoders, there is surprising evidence [128, 129] that they are similar to random encoders (e.g. max pooling over pre-trained word embeddings). In other words, the performance gain achieved by using sentence encoders such as InferSent and SkipThought over random parameterized methods such as RNNs is surprisingly little.

Inspired by these random encoders for sentences, we would like to analyze the significance of contextual word encoders. In particular, we focus on a sequence labeling task namely Named-Entity Recognition (NER). Current methods using Recurrent Neural Networks (RNNs) train a bidirectional LSTM which takes pre-trained word embeddings as its inputs to get contextual word representation which is then used for NER classification. In this setting, we are interested in the question *What is the performance gain with contextual encoders over random context encoders?*. To answer this question, we propose to use ESNs to get contextualized word representations which can then be used for classifying the word into named entities. Since reservoirs are generated randomly and their connections are not trained for the task, resulting representations are also random yet capture some context. Therefore, the main contribution of this work is evaluating ESN-based random contextual encoder and analyze how close they can match the performance of trained contextual encoders.

4.1.1 Related Work

The usage of random projection matrices dates back to the origin of neural networks in which the first few layers of multi-layer perceptrons are initialized with random weights [130]. It has been studied under various themes such as extreme learning [131, 132] and reservoir computing [98, 99].

Of particular interest to us is reservoir computing, which uses ESNs as random context encoders. ESNs have been successfully applied in various sequential tasks; sequence generation [51], time series prediction [101, 102]. Extending this to deep architectures, deep ESNs have been applied to detect Parkinson’s disease [133]. However, its application in natural language processing is relatively an unexplored area and is limited to learn grammatical structures [134], and systematicity in natural language [135]. Differing from these methods, our approach focuses on obtaining contextual word representations to be used in NER, which has not been studied with ESNs.

4.2 Named Entity Recognition

Named Entity Recognition (NER) involves identifying named entities within a given context and assigning them appropriate labels such as person (PER), organization (ORG), location (LOC), and so on. NER serves as either an independent task to identify sensitive entities within a document or as a preliminary step in various other NLP text processing pipelines, including relation extraction and co-reference resolution. Typically, NER is approached as a sequence labeling task, wherein the objective is to assign a label to each element in the provided sequence (sentence), as illustrated below:

Raj is living in Düsseldorf since 2018

In a more abstract sense, Named Entity Recognition (NER) can be viewed as a structured prediction (SP) problem, as the labeled sequence possesses inherent structure and differs from standard classification or regression tasks. Other examples of SP problems include natural language text generation (NLG), machine translation, and summarization. One of the primary challenges in solving SP problems lies in searching for the optimal label sequence within a large, combinatorial output space. For instance, in the case of NER, the number of possible output sequences scales exponentially with the number of words in the sequence (N) and the number of potential labels (L), resulting in an output space of $O(N^L)$. Due to this inherent complexity, NER is typically not tackled using search-based methods, where the objective is explicitly finding the optimal label sequences. Instead, an incremental solution is preferred, where the search process involves parsing the given sentence in a left-to-right (or vice versa) direction, incrementally determining the optimal label for each word. This incremental approach reduces the complexity to $O(N * L)$ and is widely adopted by most NER approaches.

Early NER systems focussed heavily on domain-specific knowledge in the form of lexicons and simple hand-crafted rules and features; for instance, the morphology of the word, trigger context words and term frequency. In the later years, these systems were then replaced by supervised machine learning models by including more sophisticated features such as part-of-speech tagging, word embedding and context features. For a detailed overview of these earlier approaches, we refer to the review [136]. In recent years, research has shifted away from these feature-engineered systems towards automatic feature-inferring systems using neural network-based methods as highlighted in the review [137]. Most of these NER systems based on LSTMs consist of the same underlying architecture; a bidirectional LSTM that takes a sequence of word embeddings as its input and returns a sequence of contextual word embeddings embedding into its sentence context. These contextual word embeddings are obtained by concatenating the LSTM’s hidden state of a word in both directions. Often, the output of LSTM is coupled with a Conditional Random Field (CRF), a probabilistic method that can predict labels of sequences by taking label dependencies into account.

4.3 ESNs as Random Context Encoders

This section discusses how ESNs can be used to obtain contextualized word representations. Specifically, we assume each sentence is represented by pre-trained word embeddings. Let us consider a sentence s consisting of L tokens, which can be represented as a sequence of its pre-trained token embeddings i.e. $s = (x_1, x_2, \dots, x_L)$. With this input sequence and an ESN with randomly initialized input and reservoir weights, we feed the given input sentence one token at a time and collect its reservoir states at every time step as $c = (r_1, r_2, \dots, r_L)$ by following equation (2.1).

By computing context in this manner, the representation of a token includes the previous context. Whereas for NER, previous works [126, 128] observed that that it is beneficial to have a bidirectional context. To obtain this, we pass the sequence also in reverse order once and collect its corresponding reservoir states. Therefore, contextual word representation for each token is the concatenated reservoir states obtained from both directions i.e. $c = (r_1, r_2, \dots, r_L)$ where each r_i is $[\vec{r}_i, \overleftarrow{r}_i]$. It is to be noted that the same ESN is used for both directions, unlike LSTM-based methods, which use two separate hidden layers parsing in each direction. This is primarily because W^{res} is randomly initialized.

To solve a sequence labeling task such as NER, we pass this sequence c of contextual-word representations to the readout layer with weights W^{out} and its output can be collected in a sequence $y' = (y'_1, y'_2, \dots, y'_L)$ following equation (2.2) with f_{out} as softmax activation function. Given a ground truth sequence $y = (y_1, y_2, \dots, y_L)$ of NER tags where each y_i is a one-hot encoded vector of possible tags, then it is sufficient to optimize W^{out} by using gradient descent at the rate of η minimizing cross-entropy loss \mathcal{L} between predicted sequence y' and ground truth sequence y .

$$\mathcal{L}(y', y) = - \sum_{i=1}^L \sum_{j=1}^C y_i(j) \log(y'_i(j)) \quad (4.1)$$

$$W^{out} = W^{out} - \eta \nabla_{W^{out}} \mathcal{L}(y', y) \quad (4.2)$$

As input and reservoir weight matrices W^{in} and W^{res} are kept constant throughout this training process, we can break this whole procedure naturally into (1) generating contextual-word representations for all sentences (2) fitting a readout layer. Step (1) can be done just once for each ESN setting and contextual representations can be cached. In Step (2), any classifier of our choice (shallow or deep) can be trained offline with batch gradient descent.

4.4 Experiments and Results

Now, we shift our attention to the quantitative experiments in which the contextual-word representations obtained using ESNs are evaluated on the task of German NER. For this purpose, we test our approach on GermEval Dataset [138] in which the task is to identify named entities belonging to 12 classes in total. These include four main classes: PERson, LOCation, ORGanisation and OTHER. Additionally, each class contains 2 subclasses namely -deriv and -part. One example of a nested entity is “University of Bonn” which is an ORG but at the same time contains a location entity “Bonn” (LOC-part). Similarly, the second subclass includes word derivation, for example, “das Bonner Theater” (translated as “the theater of Bonn”) contains the word “Bonner”(LOC-deriv) derived from the entity “Bonn”. The entities and their spans are encoded using BIO-scheme. With these subclasses and the encoding

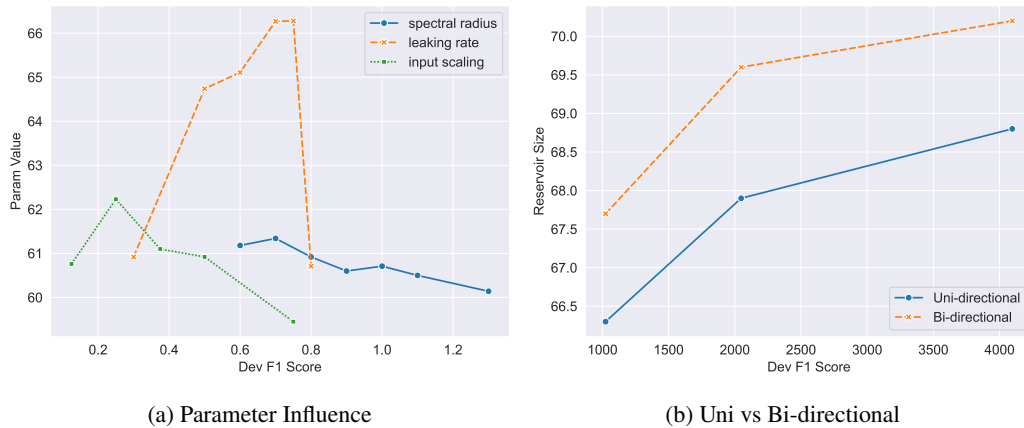


Figure 4.1: **Ablation:** (a) Influence of hyperparameters such as spectral radius, leaking rate and input scaling. (b) Influence of reservoir size on performance; the larger the reservoir, the better the performance. The fact that the bidirectional variant performs better than the uni-directional suggests that the task depends on context from both sides

scheme, each token might belong to one of 25 classes. The dataset has pre-defined splits of training, development and test consisting of 24 000, 2 200 and 5 100 sentences respectively.

4.4.1 ESN Design

Since ESNs have several hyperparameters that must be set beforehand, we tune them by cross-validating on the development set. To that end, we use pre-trained word embeddings from FastText [139] as the inputs. For this purpose, we use the Flair implementation¹, which offers an easy-to-use framework for training and evaluating NER models along with access to different types of embeddings. To keep this tuning process tractable, as discussed before, we first generate contextual-word embeddings for different settings of ESN. Later, we fit a logistic regression model as our readout layer to train it to predict NER tags. We use Ada delta optimizer trained for 150 epochs with a learning rate of 0.1 and weight decay of 10^{-5} to train this readout layer. Next, we briefly explain the settings of ESN hyperparameters:

Spectral radius: The spectral radius $\rho(W^{res})$ which is the maximal absolute eigenvalue of the reservoir weight matrix. As W^{res} is initialized randomly, it is essential to set the spectral radius to a value less than 1.0 to satisfy the echo state property [98]. However, it was shown empirically that sometimes even higher values satisfy this condition and deliver better performance [101]. We varied the spectral radius between 0.6 and 1.3 and measured the resulting F1 score. Figure 4.1 shows the F1 scores for different spectral radii; the performance peaks around 0.7 and then decreases continuously apart from a small local maximum around the unit spectral radius.

Input scaling: Another key parameter of ESNs is the scaling a of input weight matrix W^{in} . These weights can be controlled to influence the non-linearity of the reservoir responses. In our setting, we

¹ <https://github.com/zalando-research/flair>

Model	Embedding Dimension	Run Time	Performance
Logistic regression (LR)	300	33 min	52.75 \pm 0.19
Bi-ESN + LR	4096	37 min	69.04 \pm 0.22
Bi-ESN + NN	4096	47 min	70.54 \pm 0.10
Bi-LSTM	4096	10h 12 min	75.45 \pm 0.10
Bi-LSTM	256	3h 3 min	76.87 \pm 0.21

Table 4.1: **NER on Word Embeddings:** Comparison of performance of different models using non-contextualized word embeddings as inputs. Even with simple word embeddings as inputs, reservoir representations obtained from Bi-ESNs are very competitive to fully trained Bi-LSTMs.

use $\tanh(\cdot)$ activations; for linear tasks, it is thus beneficial to have small weights around 0.0, where the activation is almost linear. On the other side, for a more complex task, it might be better to choose a high scaling to use the non-linearity of the activation function. However, it was shown in previous research [101], which suggests that large weights can make the ESN unstable. In Figure 4.1, the effect of the input scaling is also presented, with the highest f1-score corresponding to input scaling of 0.25.

Reservoir size: Next, we investigate the influence of reservoir size for both uni- and bidirectional ESNs. Since a bidirectional ESN produces an embedding of twice the reservoir size, we instead look at the final embedding sizes to obtain a fair comparison. Figure 4.1(b) shows that the bidirectional embedding leads to a continuously better f1-score of around 1.5%, indicating the NER task indeed benefits from having a bidirectional temporal context.

4.4.2 Evaluation on Different Embeddings

Next, we investigate the performance of contextual-word representations obtained from ESN on the test set by considering two types of word embeddings: FastText and Stacked embedding of FastText (dimension of 300) and Flair (dimension of 4096) [140]. We also consider two variants of ESN, first with a logistic regression (Bi-ESN+LR) readout layer and second with a neural network (Bi-ESN + NN) with a hidden layer consisting of 1000 neurons and 0.5 dropout. In either case, we fix our size of the reservoir to have 2048 neurons amounting to 4096-dimensional embedding (bidirectional). The optimal setting of other hyperparameters is obtained from the analysis presented in the previous section. We compare our approach to several baseline models: (i) logistic regression (LR), which does not use any context (ii) bidirectional LSTM (Bi-LSTM) which learns contextual word-representations in an end-to-end fashion for NER task with hidden size 256 as chosen in [140]. For a fair comparison, we also train a variant with a hidden size that matches the size of the ESN reservoir.

Word Embeddings as Inputs: Table 4.1 summarizes the performance of all methods trained with FastText embeddings as inputs. It is evident that the contextual-word representations generated by ESN lead to a strong improvement over a logistic regression method. Comparing the two ESN variants, it is noticed that a readout with neural networks has $\approx 1.5\%$ improvement over the logistic regression readout layer. The most important result is that the difference between the LSTM and the ESN models

Model	Embedding Dimension	Run Time	Performance
Logistic regression	4396	2h 10 min	73.78 \pm 0.18
Bi-ESN + LR	8792	3h 40 min	76.74 \pm 0.03
Bi-ESN + NN	8792	4h 40 min	78.17 \pm 0.19
Bi-LSTM	8792	75h	81.24 \pm 0.13
Bi-LSTM	256	4h 35 min	83.52 \pm 0.21

Table 4.2: **NER on Flair + Word Embeddings:** Comparison of performance of different models using stacked flair and word embeddings. Although the resulting stacked embedding is contextualized, reservoir representations obtained using Bi-ESNs still provide a considerable performance gain over original stacked embeddings

only amounts to a value of 6%. This suggests that LSTM variants do not improve much over ESN methods but only incur longer training times of \approx 3h. On the other hand, ESN methods can be quickly trained in less than an hour \approx 0.5h. This result suggests that random contextual-word representations obtained from ESN are already competitive and can be used as a baseline while bench-marking LSTM-based NER models.

Contextualized Word Embeddings as Inputs: Table 4.2 presents the same analysis as before, combining Flair and FastText token embeddings as inputs. In these experiments, we choose the reservoir size as 4396 resulting in contextual-word embedding size of 8792 due to bi-direction. One crucial observation is that all methods have a considerable improvement over the results presented in Table 4.1 which shows that Flair embeddings are more powerful in the NER task. As Flair embeddings already encode contextual information as opposed to FastText, one might expect no further improvement by applying a contextual encoding using ESN or LSTM. Nevertheless, both the ESN and the LSTM increase the performance noticeably by 3 to 5% and 8 to 10% respectively. Comparing the LSTM with the ESN models, we observe that the performance gap is just around 5%. These findings concur with our previous analysis that ESNs are capable of achieving competitive performance as LSTMs while requiring only a short period of training time.

4.5 Conclusion

In this work, our focus was on exploring the efficacy of ESNs as random contextual-word encoders. Although these encoders do not train input and recurrent weight connections, they still possess the capability to capture contextual information that can be valuable for the NER task. Through a comprehensive set of experiments, we discovered that these encoders can be trained efficiently and achieve competitive accuracy, comparable to state-of-the-art methods. The slight difference in performance between trained encoders (such as LSTMs) and our ESN-based random context encoders makes our approach an ideal baseline for evaluating alternative methods of contextual representation learning. Several intriguing aspects within ESNs warrant further exploration, including investigating connection sparsity and exploring different reservoir structures (such as random graphs or scale-free networks). Additionally, previous research on multi-layer ESNs [141] suggests that incorporating multiple layers could lead to performance enhancements.

Echo State Networks for Partial Observability

Chapter 3 showed ESNs are capable of performing a challenging structured prediction problem of memorizing and recalling sequences of data. Similarly, the subsequent chapter delved into applying ESNs in Named-Entity Recognition (NER), another task involving structured prediction. Despite their random architectures, ESNs exhibited a type of short-term memory that proved adequate for achieving competitive performance in these tasks. These findings highlight the efficacy of ESNs as random context encoders for structured prediction tasks.

In this chapter, we delve deeper and consider a sequential-decision making problem in the field of RL. The typical setup is that there is an agent interacting with an unknown environment by observing its states, performing an action and receiving rewards. The main goal of the agent is to perform a sequence of actions conditioned on its states such that a long term expected cumulative reward is maximized. In this context, there exists an inherent structure within the outputs that needs to be discovered for the specific problem at hand. Moreover, it is crucial to note that the agent's current actions have lasting effects on future states that it will encounter; in other words, actions carry long-term consequences.

Deep neural networks (DNNs) have replaced tabular-based value and policy methods, particularly when dealing with high-dimensional states and actions that render the latter intractable. However, employing DNNs in RL introduces its own set of challenges. Even simple feed-forward networks can suffer from instability and divergence issues, often referred to as the *deadly triad* which arises from the combination of bootstrapping and off-policy learning. To address these challenges and enhance stability in learning, the utilization of an experience replay buffer [142] has proven effective to a certain extent.

In this work, we are focused on partially observable settings where the observation from the environment does not fully capture the state of the environment. In such scenarios, in order for the agent to make informed decisions, it becomes necessary for the agent to incorporate the history of observations as input in order to select an appropriate action. Typically, an RNN is employed to process the history, and its hidden state serves as a condensed representation of the historical information for subsequent processing.

Considering that training RNNs is inherently more difficult, and when combined with the challenges posed by the *deadly triad* the task becomes even more challenging. Consequently, our focus shifts toward investigating whether randomized networks, such as Echo State Networks (ESNs), can also provide solutions to this problem. Exploring this direction, we aim to discover alternative techniques for

training ESNs. In particular, we propose using Simultaneous Perturbation Stochastic Approximation (SPSA), a gradient approximation technique that can effectively compute gradients with just two function evaluations, regardless of the parameter dimension.

In summary, we explore the usage of ESNs for learning control policies in partially observable RL settings. We propose an alternative approach to train these policies without using backpropagation of gradients. Finally, we compare our methods to classic RL value-based methods.

The chapter is based on the publication [53].

- R. Ramamurthy, C. Bauckhage, R. Sifa and S. Wrobel, “Policy Learning Using SPSA”, *Proceedings of International Conference on Artificial Neural Networks*, 2018, URL: https://doi.org/10.1007/978-3-030-01424-7_1

The idea of using ESNs as contextual encoders for partial observability problem was proposed by Rajkumar Ramamurthy. However, an alternative way of training them using SPSA was proposed by the second author. Rajkumar Ramamurthy did the implementation thoroughly and performed the experiments. The paper is written mainly by Rajkumar and the second author. Later, it was revised by all co-authors.

The rest of the chapter is organized as follows: First, a brief background and motivation of the problem that we considered is presented in Section 5.1, followed by a short literature review. Section 5.2 presents a short introduction to SPSA and its design choices. Section 5.3 proposes the application of ESNs as RL control policies, along with training details and variants. In Section 5.4, we present experimental results on classic control problems and benchmark the proposed method against classic RL methods.

5.1 Motivation

The primary objective of artificial intelligence research is to create systems that can learn to solve complex tasks through interactions with their environment. Considerable progress has been made in this area, primarily through modern reinforcement learning (RL) techniques [143, 144]. Noteworthy successes include surpassing human-level performance in console-based Atari games [145] and navigation of 3D virtual environments [146]. AlphaGo Zero [58] made a groundbreaking achievement by becoming the first program to defeat world-class GO players solely through self-play learning. The implementation of function approximators like deep neural networks, coupled with off-policy and bootstrapping techniques such as Q-learning, which were previously unstable and referred to as the “deadly-triad” [147], has now proven to be a reliable approach. Techniques such as experience replay [142] have played a vital role in stabilizing learning by incorporating a large replay memory.

Motivated by these achievements, recent research has explored alternative approaches to reinforcement learning (RL) that employ black-box optimization methods, eliminating the need for backpropagation of gradient computations. Notable contributions in this regard include systems trained using evolution strategies [148, 149], which have demonstrated competitive performance in playing Atari games. Similarly, comparable results were obtained using genetic algorithms [150], which were found to scale better than evolution strategies. These advancements have revitalized interest in black-box methods for RL, mainly due to their potential for parallelization with modern distributed architectures.

However, many real-world systems face the challenge of limited and noisy state information, leading to partial observability, as encountered in partially observable Markov decision processes (POMDPs). To tackle RL problems in such scenarios, systems need to incorporate internal memory. Consequently, recurrent RL methods have recently been investigated to address partial observability, although they have proven to be challenging to train [151].

This work addresses specific problems using reinforcement learning (RL) techniques in partially observable environments. Our choice of echo state networks (ESNs) for training parameterized control policies is based on their simple architecture and short-term memorization capabilities. We propose adopting Simultaneous Perturbation Stochastic Optimization (SPSA), a gradient approximation technique, as the training algorithm. Notably, SPSA only requires two objective function evaluations per iteration, regardless of the parameter dimension. We devise three variations of ESN training using SPSA, differing in the selection of weight matrices during each iteration. Subsequently, we employ these ESNs to learn policies and evaluate their performance against baselines in classic control problems.

Earlier research on training ESNs using black-box methods explored the combination of genetic algorithms to train internal reservoir weights and stochastic gradient descent for training output weights [152, 153]. Another approach involved evolving the output weights and spectral radii of internal weight matrices [154]. Additionally, recent work [155] focused on adapting reservoir matrices using hebbian learning rules. A hybrid approach [156] that combined hebbian learning and temporal difference learning was proposed to adapt actor-critic ESNs. In contrast to these previous methods, our approach utilizes SPSA to optimize the entire network weights, which offers several notable advantages: (i) it requires only two loss measurements per iteration, (ii) it does not rely on backpropagation of gradients, (iii) it eliminates the need for maintaining candidate solutions like in genetic algorithms, and (iv) it can handle stochastic returns, removing the necessity for averaging over multiple measurements to account for noisy returns.

5.2 Simultaneous Perturbation Stochastic Approximation

In this short section, we briefly recall the main ideas behind SPSA for derivative-free optimization.

Consider the general problem of maximizing a differentiable objective function $f(\vec{\theta}) : \mathbb{R}^d \rightarrow \mathbb{R}$, that is, consider the problem of finding $\vec{\theta}^* = \operatorname{argmax}_{\vec{\theta}} f(\vec{\theta})$.

For many complex systems, the gradient $\partial f / \partial \vec{\theta}$ cannot be computed directly so that $\partial f / \partial \vec{\theta} = \vec{0}$ can often not be solved. It is, however, typically possible to evaluate $f(\vec{\theta})$ at various values of $\vec{\theta}$ which, in turn, allows for computing stochastic approximations of the gradient. One method in this regard is SPSA due to Spall [157] which iteratively updates estimates of the optimal $\vec{\theta}$ as

$$\vec{\theta}_{k+1} = \vec{\theta}_k + l_k \vec{\hat{g}}_k(\vec{\theta}_k) \quad (5.1)$$

where $\vec{\hat{g}}_k(\vec{\theta}_k)$ is an estimator of the gradient at $\vec{\theta}_k$ and l_k is the learning rate in iteration k . To estimate the gradient, two perturbations are generated, namely $(\vec{\theta}_k + c_k \vec{\delta}_k)$ and $(\vec{\theta}_k - c_k \vec{\delta}_k)$ where $\vec{\delta}_k$ is a perturbation vector and c_k is a scaling parameter. Then, the possibly noisy objective function $F(\cdot) = f(\cdot) + \text{noise}$ is measured at $F(\vec{\theta}_k + c_k \vec{\delta}_k)$ and $F(\vec{\theta}_k - c_k \vec{\delta}_k)$ and the gradient is estimated

using a two-sided gradient approximation

$$\vec{g}_k(\vec{\theta}_k) = \frac{F(\vec{\theta}_k + c_k \vec{\delta}_k) - F(\vec{\theta}_k - c_k \vec{\delta}_k)}{2 c_k \vec{\delta}_k}. \quad (5.2)$$

The convergence of the SPSA algorithm critically depends on the choice of its parameters l_k , c_k and $\vec{\delta}_k$. Specifically, the learning rate l_k must meet the Robbins-Monro conditions [158], namely $l_k > 0$ and $\sum_{k=1}^{\infty} l_k = \infty$, and a common choice in practice therefore is $l_k = \frac{l}{(L+k)^\alpha}$ where $l, \alpha, L > 0$. Similarly, the scaling factor c_k must satisfy $\sum_{k=1}^{\infty} \left(\frac{l_k}{c_k}\right)^2 < \infty$ so that a good choice amounts to $c_k = \frac{c}{k^\gamma}$ where $c, \gamma > 0$. And, essentially, each element of the perturbation vector $\vec{\delta}_k$ is sampled from a uniform distribution over the set $\{-1, +1\}$.

5.3 Echo State Networks as Policies

In this section, we will provide a concise overview of policy learning in scenarios involving partial observability. Subsequently, we will introduce our approach, which involves utilizing ESNs trained through the SPSA algorithm for policy learning purposes.

5.3.1 Partial Observability

Suppose there is an agent that interacts with an environment. At each time step t , the agent receives observations of the environment's state \vec{s}_t and takes an action a_t based on a policy $\pi(a_t|\vec{s}_t)$, which maps the state \vec{s}_t to the probability of selecting action a_t . Consequently, the environment provides a reward r_t and transitions to a new state \vec{s}_{t+1} . These interactions are generally modeled as a Markov Decision Process (MDP). For details, refer to Section 6.3.1

However, when dealing with partially observable environments, the sensory inputs of agents are limited, resulting in incomplete knowledge of the state information. Consequently, the state \vec{s}_t fails to satisfy the Markov property as it does not fully capture the past events, making it insufficient for informed decision-making. In the case of such non-Markovian states, it becomes necessary to design a policy that depends on a history of states $h_t = \{\vec{s}_t, \vec{s}_{t-1}, \dots\}$ rather than solely relying on the current state \vec{s}_t . Therefore, the policy is modified to $\pi(a_t|h_t)$.

However, this becomes impractical to compute whenever different tasks require arbitrary lengths of histories. In situations like these, an ESN can be used to integrate the required history in its reservoir states. In this way, we can parameterize the policy with weights of an ESN $\vec{\theta}$ as $\pi_{\vec{\theta}}(a_t|\vec{s}_t)$ which takes the current state \vec{s}_t as the input and returns probabilities of actions by compacting the history of input states in the reservoir memory.

5.3.2 Policy Learning Using Echo State Networks

We briefly recall the notion of ESNs and slightly modify it to use it for learning policies. In our setup, given that the state of the environment $\vec{s}_t \in \mathbb{R}^{n_s}$ is given as the input to the network, the hidden states and output of our policy network are given by $\vec{h}_t \in \mathbb{R}^{n_h}$ and $\vec{\pi}_t \in \mathbb{R}^{n_a}$, respectively. The following,

non-linear dynamical system governs the temporal evolution of such a ESN policy

$$\vec{h}_t = (1 - \beta) \vec{h}_{t-1} + \beta f_h(\mathbf{W}^{res} \vec{h}_{t-1} + \mathbf{W}^{in} \vec{s}_t) \quad (5.3)$$

$$\vec{\pi}_t = f_\pi(\mathbf{W}^{out} \vec{h}_t) \quad (5.4)$$

where $\beta \in [0, 1]$ is called the leaking rate and \mathbf{W}^{in} , \mathbf{W}^{res} , and \mathbf{W}^{out} are the input, reservoir, and output weight matrices, respectively. The functions $f_h(\cdot)$ and $f_\pi(\cdot)$ are activation functions of reservoir and output respectively

At any time, the goal of the agent is to maximize the expected cumulative reward or the return received over a period of time which is defined as $R_T = \sum_{t=1}^T r_t$. In other words, the objective function that is to be maximized is $f(\vec{\theta}) = \mathbb{E}_{\pi_{\vec{\theta}}} [R_T]$ and finding an optimal policy amounts to finding $\vec{\theta}^* = \operatorname{argmax}_{\vec{\theta}} f(\vec{\theta})$ where we now write $\vec{\theta}$ to denote the set of weights of an ESN used to approximate the policy $\pi_{\vec{\theta}}(a_t | \vec{s}_t)$.

According to our discussion in Section 5.2, we can then iteratively learn an optimal $\vec{\theta}$ according to a stochastic gradient ascent rule that follows the gradient $\nabla_{\vec{\theta}} \mathbb{E}_{\pi_{\vec{\theta}}} [R_T]$. We can now resort to SPSA in order to approximate this gradient as

$$\nabla_{\vec{\theta}} \mathbb{E}_{\pi_{\vec{\theta}}} [R_T] \approx \frac{F(\vec{\theta} + \vec{\epsilon}) - F(\vec{\theta} - \vec{\epsilon})}{2\epsilon} \quad (5.5)$$

where $F(\cdot)$ is the stochastic return from the environment by running an episode where, in each step, the agent follows the policy $\pi_{\vec{\theta}}(a_t | \vec{s}_t)$ approximated by the ESN and where $\vec{\epsilon}$ is the perturbation generated by SPSA. A summary of this learning method can be found in Algorithm 1.

Algorithm 1 Learn policies using SPSA

Input: SPSA parameters l, c, L, α, γ and initial weight $\vec{\theta}_0$

for $k = 0$ to $\frac{k_{max}}{l}$ **do**

$$l_k = \frac{l}{(L + k)^\alpha}$$

$$c_k = \frac{c}{k^\gamma}$$

$$\vec{\delta}_k \sim \mathcal{U}(-1, 1)$$

$$\vec{\theta}^+ = \vec{\theta}_k + c_k \vec{\delta}_k$$

$$\vec{\theta}^- = \vec{\theta}_k - c_k \vec{\delta}_k$$

Compute returns $F(\vec{\theta}^+)$ and $F(\vec{\theta}^-)$ by running an episode with weights $\vec{\theta}^+$ and $\vec{\theta}^-$ respectively

$$\vec{\hat{g}}_k(\vec{\theta}_k) = \frac{F(\vec{\theta}^+) - F(\vec{\theta}^-)}{2 c_k \vec{\delta}_k}$$

$$\vec{\theta}_{k+1} = \vec{\theta}_k + l_k \vec{\hat{g}}_k(\vec{\theta}_k)$$

end for

5.3.3 Tackling Exploration

An agent's policy can either be deterministic or stochastic. In a discrete action space, the agent may apply a deterministic, greedy, "winner-takes-all" strategy to select an action, i.e. $a_t = \operatorname{argmax}_a \pi_{\vec{\theta}}(a | \vec{s}_t)$.

However, in order to encourage exploration, the agent can follow a stochastic softmax policy in which actions are sampled based on action probabilities according to the policy $\pi_{\vec{\theta}}(a_t|\vec{s}_t)$, i.e. $a_t \sim f_{\pi}$ where f_{π} is the softmax function. In a continuous action space, the agent's actions are sampled from a Gaussian policy parameterized by mean and variance neurons, that is f_{π} is considered a Gaussian probability distribution. In practice, in case of discrete action space, for each possible action, there is an output neuron, with f_{π} as softmax function applied to the output layer, whereas in continuous action spaces, for each motor action, there are two neurons (one for mean and one for variance) with f_{π} as a gaussian probability distribution applied at the output layer.

5.3.4 Training Variants

Typically, only the output weight matrix \mathbf{W}^{out} is optimized in ESN. However, some tasks require tuning of the input- and reservoir weights \mathbf{W}^{in} and \mathbf{W}^{res} in order to extract relevant information from observations or to construct missing state information. Therefore, we consider three variants of our SPSA algorithm using different choices of $\vec{\theta}$ at each iteration

1. `output_spsa`: at each iteration, we optimize only the output weight matrix, that is we let $\vec{\theta} = \mathbf{W}^{out}$
2. `all_spsa`: at each iteration, all of the weight matrices are updated at once, that is we let $\vec{\theta} = \{\mathbf{W}^{in}, \mathbf{W}^{res}, \mathbf{W}^{out}\}$
3. `alternating_spsa`: at each iteration, we update one of these matrices and alternate in the subsequent iteration.

5.4 Experiments and Results

We assessed the aforementioned SPSA variations using a set of classic control problems provided by OpenAI Gym [80], and we compared their performance to temporal difference and policy gradient learning approaches. Specifically, we focused on two well-known problems: Mountain Car (MC) and Acrobot. Within the Acrobot environment, we examined two variants: one involving stochastic policy training and another involving deterministic policy training. The observation space in Acrobot consists of cosine and sine values representing joint angles, while the available actions are discrete and involve applying torque of +1, 0, or -1 on the joint between two links. Similarly, we tackled both the discrete and continuous versions of the MC task. In the discrete variant, the available actions are either pushing left or right, whereas in the continuous version, the action is a force applied.

5.4.1 Implementation details

We used the same architecture of ESNs consisting of 40 reservoir neurons with \tanh activation functions for our SPSA variants and their RL baselines. The number of input- and output neurons and the output activation function are chosen depending on the task and the type of policy being learned. The weight matrices are initialized according to parameters such as sparsity, scaling and spectral radius which are carefully set as per the guidelines in [101]. The input and reservoir matrices are chosen from a uniform distribution over values $[-0.5, 0.5]$. However, the output scaling is chosen differently for each task. The initial spectral radius of the reservoir matrix and the leaking rate are chosen to be 1.0

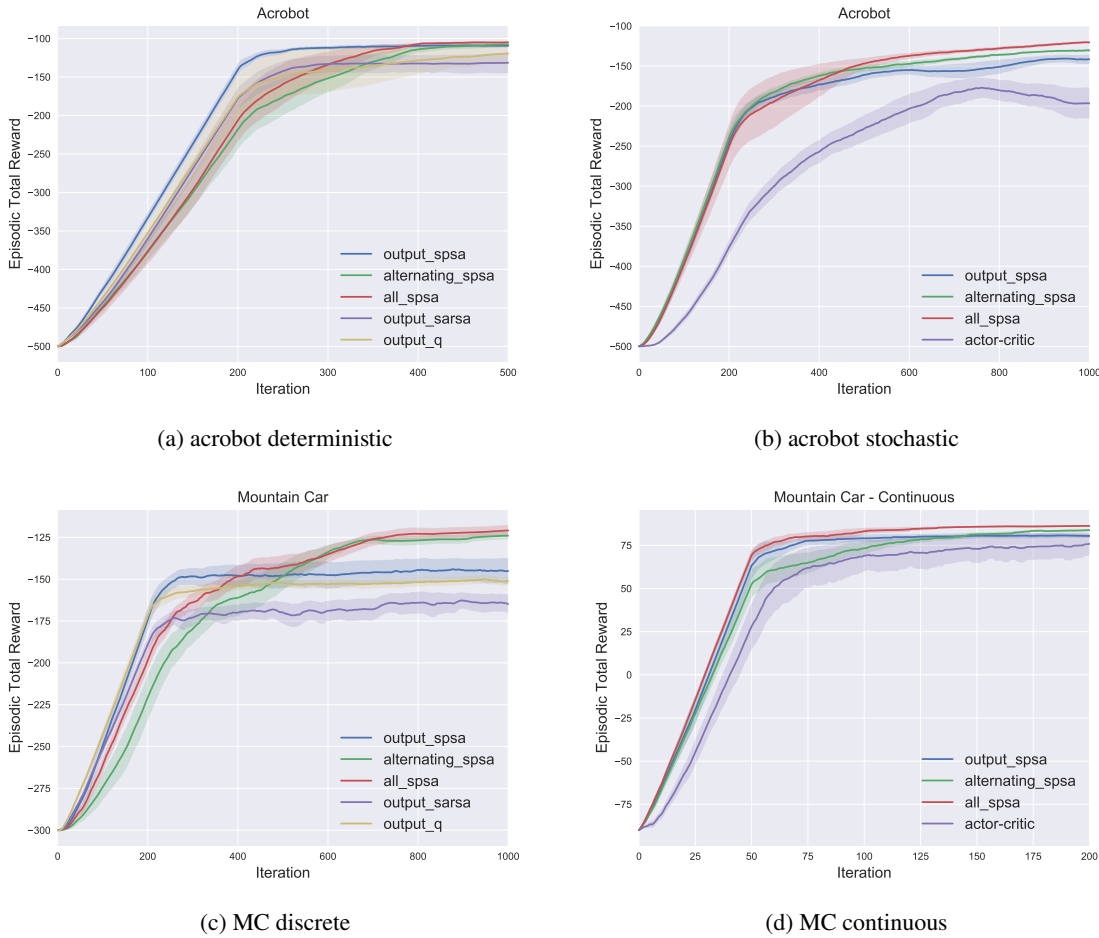


Figure 5.1: **Learning curves:** (a),(c) Evolution of total episodic reward in learning deterministic policies for discrete versions of acrobot and mountain car. (b),(d) Evolution of total episodic reward in the learning of a softmax and Gaussian policy for discrete acrobot and continuous mountain car problems tasks, respectively. It is evident that SPSA variants perform better than RL methods

and 0.3, respectively, for all tasks. The SPSA parameters such as learning rate, scaling factor, decay rates and similarly, parameters concerning reinforcement learning methods such as discount factor and learning rates are tuned for each experiment. Table A.1 lists all hyperparameters and their values.

5.4.2 Results

Initially, we conducted tests on our algorithms to train deterministic greedy policies for the discrete versions of the acrobot and MC tasks. We observed that the SPSA variants successfully solved both of these tasks. To quantitatively evaluate their performance, we generated mean learning curves using 10 different random seeds. These curves were then compared to similar curves obtained from ESNs trained with temporal difference methods such as Q-learning and SARSA learning, utilizing stochastic gradient descent. The learning curves, depicted in Figures 5.1 (a) and (b), represent the

variants	deterministic		stochastic	
	Acrobot (discrete)	Mountain Car (discrete)	Acrobot (discrete)	Mountain Car (continuous)
all_spsa	-105.56	-121.61	-121.83	85.34
alternating_spsa	-110.07	-124.70	-131.01	80.41
output_spsa	-109.16	-144.88	-141.25	80.24
output_q	-123.72	-150.69	-	-
output_sarsa	-132.28	-163.94	-	-
actor_critic	-	-	-193.95	72.64

Table 5.1: **Performance summary:** Evaluation results containing average episodic total reward in the last 100 iterations of policy learning on classic problems for different variants and their baselines (the higher the value, the better the performance)

progression of episodic total reward throughout the learning process (with higher values indicating better performance). Notably, all SPSA variants outperformed Q-learning and SARSA learning in terms of discovering better policies.

Next, we tested our algorithms to learn stochastic policies for a discrete version of the acrobot- and a continuous version of the MC task. We found that the SPSA variants are able to solve these by finding a softmax policy and a Gaussian policy for acrobot and MC, respectively. In a quantitative evaluation, we again computed mean learning curves with 10 different random seeds and compared them to data obtained using actor-critic methods. In the actor-critic method, two ESNs are used, one to learn the policy and one to learn the state value function, both act with limited state information as in our SPSA variants. Figures 5.1 (c) and (d) show the learning curves and it is seen that SPSA variants perform better than actor-critic methods. Next, in order to visualize the learned Gaussian policy for the mountain car task, we plotted action probabilities for selected input states. As we can see in Figure A.1 (b), for the same input states, the resulting action probability distribution is a mixture of Gaussians, meaning that the actions are sampled from appropriate mixture components based on the hidden states of the network which constructs the missing velocity information.

The key findings from our evaluations are summarized in Table 5.1, presenting the average episodic total rewards over the last 100 iterations with 10 different random seeds. (i) In all experiments, training solely the output weight matrix using SPSA outperforms its RL counterparts, indicating the efficacy of SPSA as a powerful alternative to common RL methods. (ii) Updating all weight matrices simultaneously yields the best performance across all tasks. However, training in an alternating fashion also shows promise, warranting further investigation. (iii) For the acrobot tasks, SPSA exhibits superior performance in learning a deterministic policy compared to a stochastic policy. This suggests that introducing stochasticity into the action space may be unnecessary, as exploration already occurs through parameter space perturbations. This finding aligns with the work done in [148], where deterministic policies were preferred when using black-box methods. Nonetheless, our approach demonstrates the feasibility of learning both deterministic and stochastic policies in general.

5.5 Conclusion

In this work, we explored the usage of ESNs as recurrent policies to effectively capture relevant state information in partially observable settings. Specifically, we proposed employing SPSA as the training algorithm, which does not rely on backpropagation. Through experiments conducted on classical problems, we demonstrated that SPSA serves as a robust alternative to conventional value-based methods like Q-learning and SARSA. Remarkably, with SPSA, training only the output weights proved sufficient for learning suitable policies across most tasks. This implies that ESNs can capture a certain degree of random context, which is adequately informative. However, for tasks seeking higher performance, SPSA can still be employed also to train the reservoir and input matrices of the ESN.

Part II

Improving Exploration and Sample Complexity in RL

In this section of the thesis, the focus is on addressing challenges in RL related to research question **RQ2**. Two approaches are presented that aim to improve sample complexity and exploration in RL by incorporating domain knowledge into the learning process.

In the first contribution, described in Chapter 6, Novelty Search (NS) methods are considered, which are a class of methods that encourage the agent to perform novel behaviors. An alternative function approximation approach is introduced for computing novelty rewards based on auto-encoders. The chapter concludes with experimental results on benchmark tasks that suggest the viability of this novelty-guided exploration approach as an alternative to classic novelty search methods.

In the subsequent contribution, presented in Chapter 7, domain knowledge is incorporated in the design of modular policy networks. Specifically, only a part of the policy network is adapted for the task, while other parts are hand-designed using domain knowledge. To achieve this, an architecture from computational biology is revisited to solve a robotic task. Finally, the chapter concludes with experiments that demonstrate how this method reduces the sample complexity by a factor of ten.

Guiding Reinforcement Learning via Encoded Behaviors

In part I of this thesis, we focused on structure prediction tasks such as sequence memorization, named entity recognition and sequential decision-making. We demonstrated that ESNs can solve these tasks without having to fully train the networks since they acted as random context encoders.

In this part of the thesis, we turn our attention to challenges concerning Deep Reinforcement Learning (DRL). DRL has led to significant breakthroughs in several applications, including game-playing agents, robotics, recommender systems and so on. Despite its successful applications, DRL is highly challenging due to several reasons. Many of the sequential decision-making tasks heavily depend on sparse reward signals, using which it is harder to assign credits to a long sequence of actions. Additionally, due to this sparse feedback, which is available only in the goal states, models often need to engage in extensive exploration. In other words, learning with RL in such demanding exploration problems often exhibit high sample complexity.

This chapter focuses on addressing this research question **RQ2** which pertains to improving exploration and sample complexity in RL. To this end, we consider Novelty Search (NS), a class of methods to encourage exploration. Even without exploiting any environment rewards, they are capable of learning skills that yield competitive in several tasks. However, these methods rely on neighborhood models that store behaviors in an archive set to assign novelty scores to policies. Hence, they do not scale and generalize to complex tasks requiring too many policy evaluations. Addressing these challenges, we propose a function approximation method that learns sparse representations of agent behaviors using auto-encoders.

Building on the simple intuition that novel behaviors tend to produce large reconstruction errors; we propose two variants of auto-encoders. First, in the initial work [69], a multi-layer perceptron that encodes the agent behavior into an encoded vector and later this latent vector is decoded back to the original behavior. It is to be noted that agent behaviors are not of fixed length and generally considered as sequences. For instance, the behavior of an agent which is trying to walk can be characterized as a trajectory of its path along a 2D or 3D world. Consequently, a sequence-to-sequence auto-encoder was proposed to handle variable length sequences in [70].

In summary, this chapter presents a scalable alternative to Novelty Search (NS) methods that are based on deep neural networks, along with experimental results which highlight that the the proposed approach works better than classic NS methods.

The chapter is based on the following publications [69, 70]:

1. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Novelty-Guided Reinforcement Learning via Encoded Behaviors”, *Proceedings of International Joint Conference on Neural Networks*, 2020, URL: <https://doi.org/10.1109/IJCNN48605.2020.9206982>
2. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Guided Reinforcement Learning via Sequence Learning”, *Proceedings of International Conference on Artificial Neural Networks*, 2020, URL: https://doi.org/10.1007/978-3-030-61616-8_27

The original idea of leveraging auto-encoders as novelty detectors in the context of NS was proposed by Rajkumar Ramamurthy. Rajkumar Ramamurthy implemented the code entirely and performed the experiments. The resulting paper was also written fully by Rajkumar, which was later revised by all coauthors. All the authors actively participated in the discussion of the entire work.

The rest of the chapter is organized as follows: First, a brief introduction is presented in Section 6.1, followed by a short overview of related work. Section 6.2 presents motivating experiments which highlight the power of NS methods. A short preliminary section is discussed in Section 6.3. Our approach of using AEs for novelty-guided RL is presented in Section 6.4. Finally, experimental results of the proposed approach against classic NS methods are presented in Section 6.5.

6.1 Motivation

Despite the successful application of Reinforcement Learning (RL) in various domains [58, 145, 159–162], RL encounters several challenges that hinder its practical implementation in real-world systems.

Firstly, the learning process heavily relies on sparse reward signals, which are only available when agents reach the desired goal state. Particularly in tasks with long time horizons, effectively propagating reward signals to past actions becomes a significant challenge of temporal credit assignment and demands a large number of roll-outs to develop an appropriate policy. To tackle this challenge, numerous approaches have been proposed. These include the utilization of shaping rewards [163] that offer additional reward signals, employing curriculum learning techniques [164], and employing temporal difference methods that extend bootstrapping through the use of multi-step returns [165, 166].

Secondly, for many real-world problems, the design of reward functions presents a significant challenge and is susceptible to various concerns [167]. These concerns include: (i) *reward hacking*, where agents exploit loopholes in the reward function to maximize rewards without actually accomplishing the intended task. For instance, in the context of a bicycle riding system [168], the agent learned to ride in circles around its starting position because the reward function rewarded progress towards the goal without penalizing failure to reach the actual goal position; (ii) *negative side effects*, where the pursuit of the task inadvertently causes undesired consequences, such as a robot knocking over a vase while performing a task; (iii) *safe exploration*, which ensures that agents do not take exploratory actions with harmful consequences; and (iv) *deceptiveness*, where reward functions can be misleading. For example, in a grid environment, an agent learning to navigate with a reward function that rewards positions close to the goal may get trapped by deceptive walls and fail to reach the actual target.

To address these challenges, various methods have been proposed. Inverse reinforcement learning approaches [169, 170] infer reward functions directly from expert demonstrations while simultaneously learning a policy to solve the given task. Alternatively, approaches involving human interaction [171, 172] rely on obtaining preferences about policies from human trainers to learn the desired optimal behavior.

Third, when agents interact with RL environments, they face the challenge of balancing exploration and exploitation. If the agent exclusively chooses actions based on their high estimated value, it may miss out on discovering alternative actions that could lead to better future rewards. In many RL solutions, exploration is often implemented through random strategies such as epsilon-greedy methods or policies based on Gaussian or Boltzmann distributions. Although recent studies have explored techniques like count-based exploration [173] and intrinsic curiosity [174], these methods either require exhaustive enumeration of state-action spaces or the learning of complex state-transition models, making them impractical for tasks with high-dimensional spaces.

Alternatively, exploration in the parameter space of deep neural networks can be achieved using black-box evolutionary algorithms, such as Evolution Strategies (ES) [148, 175]. ES has shown faster training times and better scalability than traditional RL algorithms. However, deceptive reward signals and sparse rewards still contribute to longer training times in ES, emphasizing the need for more targeted exploration methods rather than relying solely on random exploration.

This work falls within the category of methods known as Novelty Search (NS) [176, 177], which are particularly effective for promoting directed exploration in problems characterized by sparsity or deceptiveness. The central concept behind NS is to incorporate a domain-specific behavioral characteristic (BC) that captures an agent’s behavior, and then encourage the agent to exhibit different and novel behaviors compared to its previous actions. For example, [178] chose BC as the "final position of the agent" while [177] utilize the "distance traveled by the agent" in a bipedal locomotion task, demonstrating that optimizing the novelty of these BCs is more effective than solely optimizing the standard reward objective. In general, BCs can be defined by domain experts, leveraging their expertise in designing tasks. Our approach is based on the method [178], which integrates a novelty objective into the standard reward objective, leading to improved exploration in various benchmark tasks. However, this approach has two significant drawbacks: (i) Typically, a fixed-size archive set stores observed policies and their corresponding behaviors. Computing the novelty of a given policy involves retrieving its nearest neighbors from this archive set, and a simple metric is then used to assign novelty scores based on the distances to these neighbors. As complex tasks require evaluating a large number of policies, the computational burden of storing and finding nearest neighbors in such a large archive set becomes a bottleneck. (ii) BCs are often considered in terms of fixed-length sequences, which need to be more robust to handle temporal variations effectively.

This work addresses the scalability and fixed-length sequence limitations associated with NS methods. Our approach involves utilizing a function approximation technique and employing a sequential auto-encoder to learn representations of agent behaviors. Instead of storing behaviors in an archive set, we encode them into fixed-length representations. To achieve this, we propose two variants. First, a straightforward auto-encoder is implemented using a multi-layer perceptron, which operates on fixed-length behavioral characteristics (BCs). Second, a sequential auto-encoder based on Recurrent Neural Network (RNN), which takes into account of sequential information and learns encodings of variable-length BCs by leveraging sequential dependencies. In both cases, the reconstruction errors can be utilized as novelty bonuses for policies. The underlying concept is that novel behaviors tend to produce more significant errors than previously observed behaviors. By assigning novelty scores to

policies based on their reconstruction error, we can encourage robust exploration towards less-explored areas of the behavioral space. Importantly, we simultaneously learn this model alongside the policy learning process fully online, without the need for a storage buffer or an archive set.

In summary, the main contribution of this work is a simple, scalable and efficient exploration method using novelty bonuses for policies, along with a detailed experimental evaluation against classic novelty search and policy gradient methods.

Related Work: Exploration is an actively studied area in reinforcement learning. Several methods have been proposed to promote directed exploration in RL; The general theme of most approaches is to encourage agents to visit states that are seldom visited. Early work [179] in this regard proposed to learn a curiosity model which predicts future events using a history of interactions with the environment. The arising prediction error is then seen as an intrinsic reward that drives toward creative solutions. Methods [173, 180] that assign novelty scores based on visitation counts either on raw states or encoded states were also proposed. On the other hand, approaches that maximize the information-theoretic objectives [181, 182] to learn exploration strategy by encouraging diversity. Methods that are most related to our approach are (1) convolutional auto-encoders [183] to encode the given observation to hash codes upon which count-based exploration rewards are generated (2) models [174, 184] that learn one-step state dynamics from previous states and actions; and use prediction error as exploration rewards. While most previous work considered generating exploration bonuses at each step using input observation space, our work, in contrast, focuses on encoding episodic agent behaviors in NS methods.

6.2 Motivating Experiments

We consider simple grid environments in which the task is to navigate to a desired goal position. In particular, we chose to use MiniGrid environments [185] which offers multi-room settings. Rooms are separated by a door which must be opened first to gain access to the next room and the target is placed at the last room. The agent is provided with a partially observable symbolic input grid of size $7 \times 7 \times 3$ and available actions are turn left, turn right, move forward, move backward, pick-up/drop objects, toggle doors etc. Rewards are sparse and given only upon reaching the goal, with a penalty for the number of steps taken. Despite its simplicity, this setting poses a hard exploration problem with sparse goal-only rewards, which is difficult to learn with RL alone.

To illustrate the benefits of novelty-guided methods, we consider three variants consisting of two, three, and four rooms demanding increasingly higher levels of exploration (see Fig 6.1(a)). To solve these tasks, we consider Evolution Strategies (ES) using only sparse rewards, a classic novelty-guided ES which computes novelty provided by k -nearest neighbors, and, finally, the proposed novelty-guided ES using novelty bonuses provided by a behavior auto-encoder.

Fig 6.1(b) shows the learning progress for the simplest variant with just two rooms. All three methods can solve this task with ease. When increasing the difficulty to three rooms in Fig 6.1(c), we observe the benefits of novelty-guided methods, which accelerates learning and, at the same time, ES guided by the behavior model, performs slightly better. Next, we consider the hardest variant with four rooms. As seen in Fig 6.1(d), ES, which takes only the environment rewards, completely fails to discover viable policies; due to the sparse returns, there are no gradients for it to follow most of the time. While both novelty-guided methods can solve this task, novelty gradients obtained through

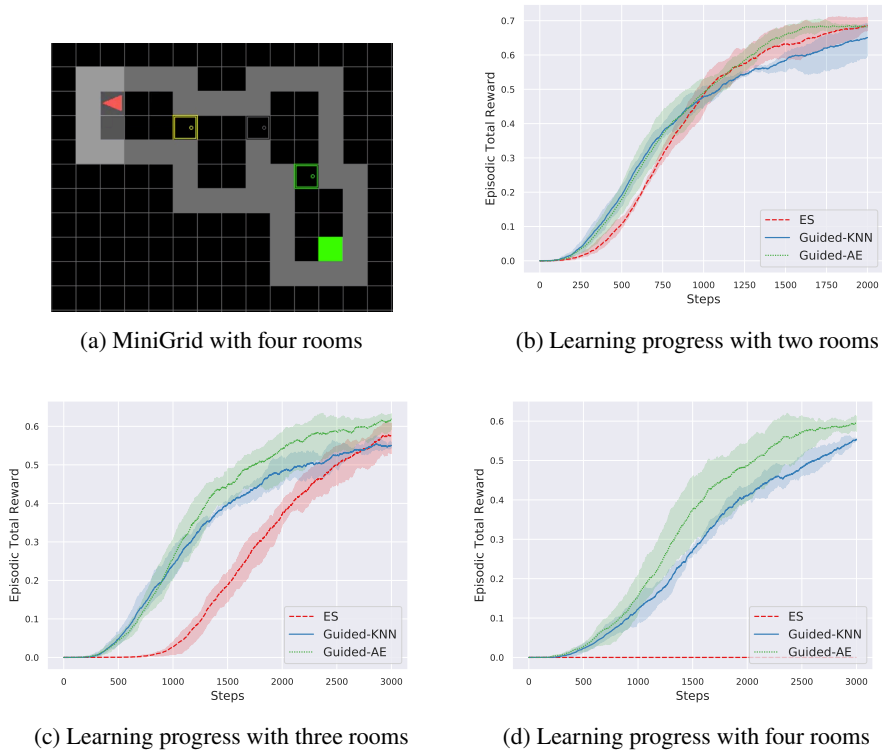


Figure 6.1: **Motivating Experiments:** Figure shows learning curves of classic ES and ES with novelty rewards such as Guided KNN and Guided AE (our approach) on different levels of MiniGrid environment. All of the methods solve the simplest variant with two rooms. However, With increased difficulty, we see that guided methods accelerate learning. With the hardest variant, ES fails to solve the task whereas guided methods can solve this task due to additional rewards

behavior models achieve higher returns than classic novelty search, which can only remember limited number of policies. We further investigate their performances in Sec 6.5 concerning continuous control tasks.

6.3 Preliminaries

6.3.1 Markov Decision Process

We consider a standard reinforcement learning setting where an agent interacts with an environment. At each time step t , the agent receives an observation o_t about the state s_t of the environment, performs an action a_t , and receives a scalar reward r_t . In fully observable environments, observations and states coincide, i.e, $s_t = o_t$. In partially observable environments, however, a history $s_t = (o_t, a_t, r_t, o_{t-1}, a_{t-1} \dots)$ of observations and agent’s interactions may be required to decide for the next action.

Typically, the environment is stochastic and formulated as a Markov Decision Process (MDP) defined by a tuple: $\langle \mathbf{S}, \mathbf{A}, \mathbf{T}, \mathbf{R} \rangle$ where \mathbf{S} is a set of states, \mathbf{A} is a set of actions an agent can perform, \mathbf{R}

is the reward function, and T is the transition probability. Upon an action a_t , the environment moves to a new state s_{t+1} according to the transition function $T(s_{t+1}, s_t, a_t)$ and responds with a scalar reward $r_t = \mathbf{R}(s_{t+1}, s_t, a_t)$. The agent's behavior is characterized by a policy function $\pi(s_t, a_t)$, which maps each state-action pair (s_t, a_t) to the probability of selecting the action in the particular state. In RL, the goal of the agent is to maximize the return discounted by $\lambda \in (0, 1)$ over a period of time T given as

$$G_T = \sum_{t=1}^T \lambda^{t-1} r_t. \quad (6.1)$$

The goal is to learn a policy function that maps states to actions to maximize the expected cumulative reward. For high-dimensional state and action spaces, tabulating the probabilities for each state-action pair is not feasible. Therefore, the policy π is often represented as a deep neural network π_θ with weights θ . The goal is to determine optimal weights θ^* that maximize the expected cumulative reward

$$\theta^* = \operatorname{argmax}_{\theta} \mathbb{E}_{\pi_\theta} [G_T]. \quad (6.2)$$

Optimization is typically achieved via stochastic gradient ascent where the gradient $\nabla_{\theta} \mathbb{E}_{\pi_\theta} [G_T]$ is obtained using sampled sequences $(s_t, a_t, r_t, s_{t+1} \dots)$ of interactions with the environment and can be computed via policy gradient methods [148, 162, 186] depending on whether a stochastic or a deterministic policy is to be found.

6.3.2 Evolution Strategies (ES)

Evolution Strategies (ES) [175, 187] are heuristic search procedures inspired by evolution. In each iteration, a population of perturbed parameters is generated and an objective function is evaluated. Using a process akin to natural selection, parameter vectors are combined to create the next population and this process continues until the population reaches a satisfactory performance. There exist several flavors of ES w.r.t. the representation of parameters or the selection process. The version we use here belongs to the class of Natural Evolution Strategies (NES). Let f be the objective function acting upon parameters θ . In a reinforcement learning setting, the stochastic return is obtained from the environment. NES algorithms maintain the population as a distribution p_ψ over parameters θ . Typically, the distribution p_ψ corresponds to a multivariate Gaussian centered around the current parameter with co-variance $\sigma^2 I$. Given this, NES seeks to maximize the average objective of the population $\mathbb{E}_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)} [F(\theta + \epsilon)]$ by optimizing the parameters θ . Generally, NES also updates the co-variance of the population distribution, but as in other RL approaches, we use a static co-variance σ by fixing it throughout the training.

To estimate the gradient of the expected cumulative reward in iteration k , n perturbations are sampled from the distribution by adding Gaussian noise to the current parameter vector θ (i.e., $\theta_k = \theta_k + \sigma \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, I)$). The gradient is then approximated by a sum of sampled perturbations weighted by their sum

$$\nabla_{\theta_k} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} [F(\theta_k + \sigma \epsilon)] \approx \frac{1}{n\sigma} \sum_{i=1}^n F(\theta_k^i) \epsilon_i \quad (6.3)$$

Typically, the objective function evaluations $F(\theta_k^i)$ are subject to rank-normalization before computing the gradients to ensure that the reward scales between different tasks do not affect the optimization process.

6.3.3 Novelty Search with Nearest Neighbors

One of the reasons why RL can not cope well with sparse/deceptive problems is that the reward function usually needs to take into account of intermediate stepping stones that would allow for learning target skills. For instance, in the grid navigation tasks in Sec 6.2, rewards do not encourage the agent to learn to start navigating the room without getting stuck in walls. Hence, solving such tasks purely with goal-only rewards raises considerable challenges and demands numerous interactions with the environment. Novelty Search (NS), inspired by nature’s tendency to evolve increasingly complex behaviors, tackles this by using novelty as a stepping-stone proxy. In other words, NS aims to drive the search process towards policies with higher novelty rather than ones with higher cumulative rewards.

In order to differentiate behaviors, each policy π_θ (parameterized by a DNN with weights θ) is assigned a domain-dependent Behavior Characteristic (BC) denoted as $b(\pi_\theta)$. For instance, in the case of grid navigation or bi-pedal walking domains, it simply could be the final two-dimensional position of the agent at the end of an episode. However, considering only the final position may not be sufficient to distinguish different behaviors terminating at the same position; therefore, it is necessary to capture the trajectory of agent positions concatenated as a sequence to form BC [178].

In the classical sense, a set of fixed size typically known as archive set \mathbf{A} is maintained to store observed policies and their behaviors. Given this set, we desire a metric $N(\theta, \mathbf{A})$ that allows us to measure the novelty of a given parameterized policy π_θ . Typically, it is defined as the average distance to its nearest neighbors \mathbf{K} . The higher the distance to its neighbors, the higher the novelty and vice versa.

$$N(\theta, \mathbf{A}) = \frac{1}{|\mathbf{K}|} \sum_{i \in \mathbf{K}} \|b(\pi_\theta) - b(i)\|^2 \quad (6.4)$$

$$\nabla_{\theta_k} \mathbb{E}_{\epsilon \sim \mathcal{N}(0, I)} [N(\theta_k + \sigma \epsilon)] \approx \frac{1}{n\sigma} \sum_{i=1}^n N(\theta_k^i, \mathbf{A}) \epsilon_i \quad (6.5)$$

We can then resort to the ES framework from Sec 6.3.2 to estimate the novelty gradient concerning current policy parameters θ_k and take a step towards parameters that produce novel behaviors. Initially, this search scheme may start with idle behaviors or ones that fail immediately. As the optimization progresses, these behaviors will become less novel paving the way for “stepping stones” to be discovered such as behaviors that walk for a few time steps. Further along the optimization, behaviors of increasing complexity are identified, and, eventually, agents learning to walk far or to navigate the grid will be discovered.

6.3.4 Combining with RL

Novelty Search can be incorporated in both settings; in sparse/deceptive settings where RL is not just enough (as discussed in MiniGrid experiments) and in general to speed up RL by promoting exploration even with dense rewards. In either case, NS and RL objectives can be combined [178] as

their weighted combination and the update rule is given as follows:

$$\theta_{k+1} = \theta_k + \alpha \frac{1}{n\sigma} \sum_{i=1}^n w F(\theta_k^i) \epsilon_i + (1 - w) N(\theta_k^i, \mathbf{A}) \epsilon_i \quad (6.6)$$

where w is the importance given to the reward objective, referred as “reward pressure”. By following gradients to this objective function, policies that are both novel and achieve higher rewards can be searched. Further w can be adapted based on learning progress. Initially, w is set to 1.0 to purely pursue environment reward signals. However, if the performance is not improved in a few iterations k_{max} , then w is decreased by δ_w . At this point, gradients slightly start to follow novelty until the performance has improved, then w is incremented by δ_w .

6.4 Novelty Bonuses via Encoded Behaviors

Classic novelty search has several drawbacks. First, the policies are stored in an archive set; therefore, to compute the novelty of a policy, its nearest neighbors must be retrieved. Since complex tasks demand a large dimensional behavior characteristic and evaluate many policies, finding the nearest neighbors becomes a bottleneck computationally. Second, it may not generalize well to unseen behaviors as only a limited number of policies can be stored. Therefore, our aim is to come up with a general mechanism for computing the novelty of a given policy rather than relying on neighborhood methods to store and retrieve closest agent behaviors.

To that end, we propose to learn representations of agent behaviors such that the prediction error in behavior space provides a good novelty bonus. These representations can be learned using a deep neural network consisting of two components: the first component maps the behavior into an encoded vector and the second component takes the encoded vector as inputs and reconstructs the behavior back. As novel behaviors are unseen inputs to this model, the arising prediction error can be seen as a measure of novelty. In the same way, frequently occurring behaviors tend to have lesser prediction errors and will be assigned a lower novelty score. Aligning with other novelty-search works, we also consider an episodic setting of reinforcement learning (i.e.) novelty bonuses are assigned to a policy at the end of an episode. In addition to novelty rewards, agents also receive the cumulative reward from the environment.

6.4.1 Training

Next, we describe the model of the sequential auto-encoder to learn behavior representations. The model consists of two components: the encoder network E with weights ϕ_e and the decoder network D with weights ϕ_d . The input to the model is a behavior sequence $\mathbf{b} = (\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_T)$ observed in an episode. The encoder network processes and encodes this sequence into a fixed-length context vector, which acts as a representation of the behavior sequence. Given this representation as input, the decoder network reconstructs the whole sequence $\mathbf{b}' = (\mathbf{b}'_1, \mathbf{b}'_2, \dots, \mathbf{b}'_T)$. The two components E and D are jointly trained to minimize the reconstruction loss L between the actual and the predicted sequences:

$$L(\mathbf{b}, \mathbf{b}') = \frac{1}{T} \sum_{i=1}^T \|\mathbf{b}_i - \mathbf{b}'_i\|^2 \quad (6.7)$$

We propose two auto-encoder variants which differ in the capability to process sequential information namely a MLP-based auto-encoder and a sequential auto-encoder based on recurrent neural networks.

First, MLP AE consists of an encoder that projects the given behavior sequence into an embedding, which is then given to another MLP decoder to reconstruct it. Since MLPs cannot handle variable-length sequences, inputs to these are fixed-length padded BC sequences subsampled at specific intervals. However, since BC might contain sequential information which might be crucial for detecting novel behaviors, we propose to use Seq2Seq AE based on recent advancements in natural language processing [38] and computer vision [188]. Seq2Seq AE contains an RNN encoder and an RNN decoder. In order to produce behavior representation, RNN encoder first processes the given BC sequence one element at a time and once the final element has been processed, the final hidden state of the RNN acts as the representation. Given this representation, the RNN decoder generates the sequence one at a time, in reverse order to keep the optimization tractable so that the network learns easily to predict low-order correlations.

6.4.2 Sparse Encoding:

Further, in order to encode diverse behaviors, we consider k-sparse auto-encoders [189], a simple variant of auto-encoders to enforce sparse representations. During the feed-forward phase, the hidden activations of the encoder are sorted and only the top k hidden units are retained while the rest of the units are set to zero. By back-propagating only through these active hidden units, the decoder learns to reconstruct the given input by using very few units, thereby also acting as a regularizer. To compare sparsity in networks with different number of hidden units, we define the sparsity level *sparsity* as the ratio of hidden units which are retained for each sample of input. We summarize the training of sparse auto-encoders in the **Algorithm 2**.

6.4.3 Novelty Scores

Given a policy π_θ and its behavior characterized by $b(\pi_\theta)$ obtained by rolling out an episode with π_θ . Then the novelty bonus is obtained by passing the behavior sequence to encoder-decoder networks to obtain the reconstruction error $N(\theta) = L(b(\pi_\theta), b'(\pi_\theta))$. In summary, our agent is composed of two sub-systems: a behavior auto-encoder model that outputs a novelty bonus for the given policy and a policy network that outputs a sequence of actions to maximize the joint objective function of cumulative reward and novelty. We summarize the entire training approach in the algorithm presented in Alg 3.

6.5 Experimental Results

We test our methods on the Mujoco continuous control tasks, which provide a standard set of benchmark tasks. In particular, we consider 4 tasks: Inverted Pendulum, Inverted Double Pendulum, Half-Cheetah

Algorithm 2 Fit Sparse Behavior Auto-encoders

Input: Learning rate β , epochs E , batch size m , behavior auto-encoder parameters ϕ_e, ϕ_d , behavior buffer A , Sparsity level $spare$

for $i = 0$ to E **do**

Generate *batches* of size m from the behavior buffer

for b in *batches* **do**

Compute encoded vectors h by forwarding b to the encoder

Find indices of largest activations of h according to specified sparsity level $ind = top_{spare}(h)$

Set activation of other units to zero $h(ind^c) = 0$

Compute decoded values b'

Compute the reconstruction error $L(b, b')$ using Eq. 6.7

Update ϕ_e and ϕ_d by taking a gradient descent at the rate β

end for

end for

Algorithm 3 Novelty-Guided RL via Encoded Behaviors

Input: Learning rate α , initial reward pressure w , iterations K , ES parameters n and σ

Initialize: Policy parameters θ , behavior auto-encoder parameters ϕ_e, ϕ_d

for $k = 0$ to K **do**

for $i = 1$ to n **do**

Sample $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$

Compute $\theta_k^i = \theta_k + \sigma \epsilon_i$

Perform a roll-out with policy $\pi_{\theta_k^i}$ and obtain behaviors $b(\theta_k^i)$

Compute novelty bonus $N(\theta_k^i)$ using behavior auto-encoder's reconstruction error

Compute cumulative reward $F(\theta_k^i)$

end for

Update policy network: $\theta_{k+1} = \theta_k + \alpha \frac{1}{n\sigma} \sum_{i=1}^n w F(\theta_k^i) \epsilon_i + (1-w) N(\theta_k^i) \epsilon_i$

Update behavior auto-encoder: Fit sampled behaviors $b(\pi_{\theta_k^i})$, $i = 1 \dots n$ to minimize Eq:6.7

Adapt the reward pressure w based on learning progress as discussed in Sec 6.3.4

end for

and Hopper. For all our experiments, the policy network is a multi-layer perceptron with two hidden layers containing 64 tanh neurons each. The input to the network is the observation vector from the environment and the output is an action vector of motor commands. ES is trained with a learning rate $\alpha = 0.01$ and a noise standard deviation of $\sigma = 0.1$. For inverted pendulum, we use $\sigma = 0.01$ as its solutions are sensitive to perturbations. To keep the experiments tractable, we limit the number of samples drawn from the population distribution in each generation to $n = 50$. For the baseline novelty method using nearest neighbors, we fix $k = 10$ and use an archive set of size 1000 which is implemented as a FIFO so that only the recent behaviors are kept. ES implementation is based on our open-sourced black-box pytorch optimization framework [190], which allows the integration of multiple objective functions (reward and novelty).

Behavior Characteristic As discussed in Sec 6.3.3, NS methods require a domain-specific behavior characteristic (BC). For the pendulum tasks, BC is chosen as a trajectory of cart and pole positions. And for other locomotive tasks, it is chosen to be the 2-D trajectory of agent positions relative to the start position when the episode begins. Since agents might learn behaviors that move in backward positions, it is necessary to align this BC concerning the task of moving forward. For this reason, we clip the behavior space such that all behaviors with negative offsets (with respect to the initial position) collapse to zero offsets.

Auto-Encoder Setup The MLP-based auto-encoder is composed of feed-forward multi-layer perceptrons whose configurations and their sparsity levels are chosen based on a formal grid search. It is trained with Adam optimizer with a learning rate of $\beta = 0.001$ and batch size of $m = 100$. The autoencoder consists of encoder and decoder RNNs with Gated Recurrent Units (GRUs)[97] consisting of several layers. Additionally, the context vector from the encoder is subject to sparsity constraints. The encoder and decoder networks are jointly trained using Adam optimizer with a batch size of 100. For stable gradient updates, a behavior buffer of size 1000 is used. The behavior sequences are sampled based on the task to have the maximum sequence length of 50, and padding is not used so that they are variable-length sequences. Other hyper-parameters such as learning rate, number of GRU units, and layers are obtained through formal search.

Novelty Search First, we evaluate our method on a pure novelty search scenario by setting the reward pressure w to 0 and compare against the classic method using k-nearest neighbors. Fig 6.2 shows the learning curves of agents trained using only novelty gradients averaged over several runs. The first main result is that our method of novelty-guided by plain auto-encoders outperforms classic novelty methods in the tasks of Half-Cheetah, Hopper and Inverted Pendulum. In the task of Inverted Double Pendulum, classic novelty search performed slightly better than plain autoencoders, yet both methods cannot solve the task while pursuing the novelty alone. However, sequential auto-encoders show considerable gains over plain auto-encoders and classic novelty search methods indicating that sequential information is quite crucial to these tasks.

Novelty-Guided RL Although novelty search methods are able to teach the necessary tasks and skills without having access to reward functions, they still ignore some other aspects of reward functions, such as energy efficiency, performance, etc. To that end, they must be combined with reward gradients by adapting reward pressure accordingly based on learning progress as discussed in Sec 6.3.4. To

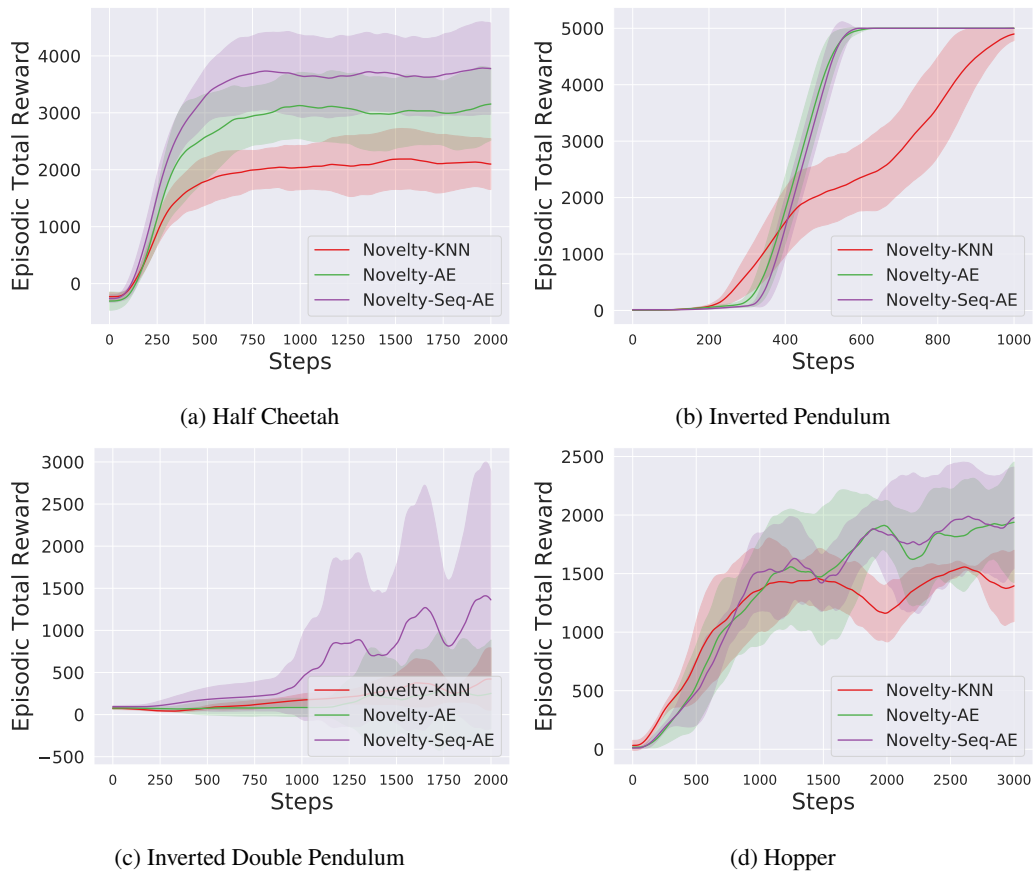


Figure 6.2: **Novelty only learning curves:** Average learning curves of agents that are trained using only novelty gradients

benchmark our methods, we consider the policy gradient method, namely Evolution Strategies (ES), which considers only reward signals. For the adaptive methods that combine with RL, we initially set the reward pressure to $w = 1.0$. The maximum stagnation steps is $k_{max} = 50$ and $\delta_w = 0.05$. Fig 6.3 shows the learning curves averaged over several runs. The main results in these experiments are summarized as (i) novelty guided methods which use both novelty and reward signals perform better than ES and speed up learning in almost tasks (ii) Novelty guided by behavior models outperformed classic novelty search in three out of four tasks and performed equally well in the other tasks. (iii) The benefit of novelty-guided methods is observed in difficult tasks such as Half-Cheetah, Hopper etc. Also, it can be seen that scores of pure novelty search (Fig 6.2 (a), (d)) are higher than that of novelty-guided methods (Fig 6.3 (a), (d)) in some tasks. This could be because novelty-guided methods also use reward signals, which penalize agents for the cost of motor actions, which might hinder learning, whereas pure novelty search ignores these aspects, thereby not impeding the learning process. (iv) benefits of using sequential autoencoders are seen in three out of four tasks indicating the learning of sequential dependencies indeed help to encourage exploration.

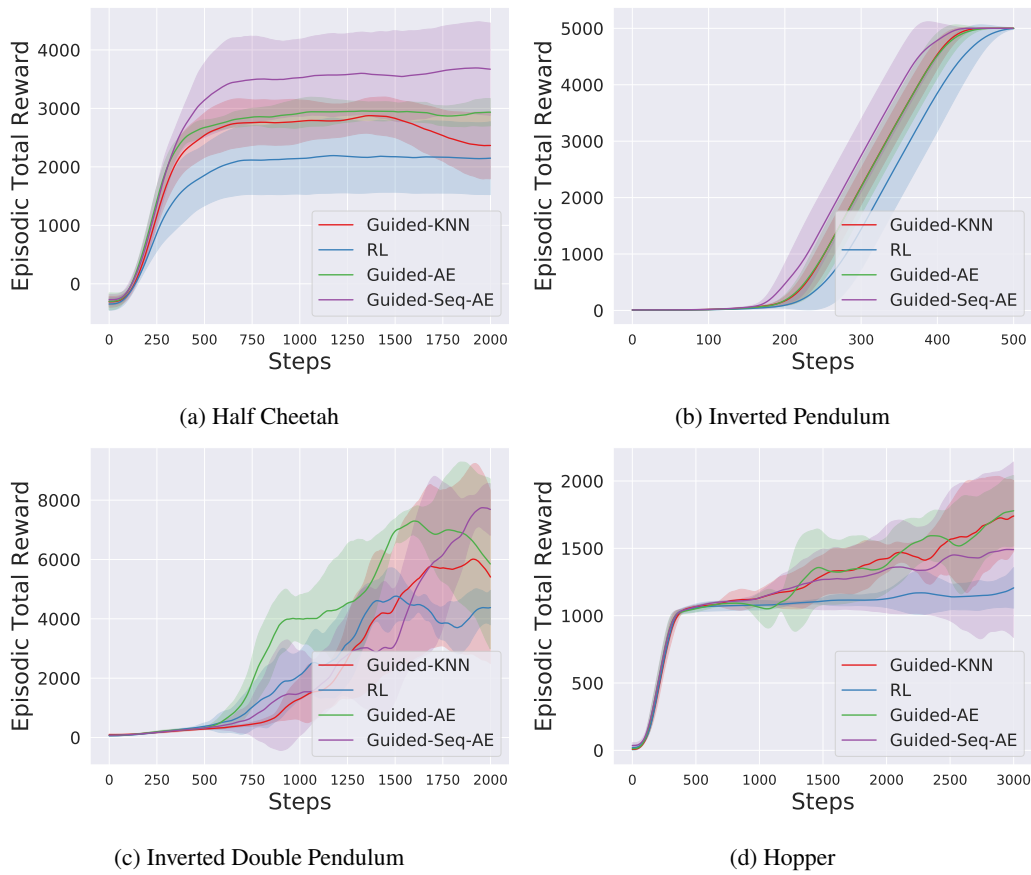


Figure 6.3: **Novelty-guided learning curves:** Average learning curves of agents that are trained using adaptive novelty-guided methods

6.5.1 Sparsity Levels

To understand the effect of enforced sparsity constraint, we considered plain behavior auto-encoders with different hidden layer configurations and sparsity levels of 0.25, 0.5 and 1.0. Note that the sparsity level of 1.0 corresponds to the classic auto-encoder in which all the hidden units are used to reconstruct the given input. For this analysis, we consider the final performance of these variants over several runs. Fig 6.4 captures the effects of sparsity when using only the novelty gradients, with the help of a box plot showing the distribution of final performance. As observed, in most settings, the sparse encodings with levels of 0.25 and 0.5 perform better than the dense encoding of 1.0. Next, we performed the same analysis by considering the final performance when both novelty scores and rewards were used for learning. The plots in Fig 6.5 also suggest that the sparse encodings performed better in most cases. In a nutshell, these results indicate that sparse representations are preferred over dense; however, this is another hyper-parameter that has to be tuned for the given task at hand.

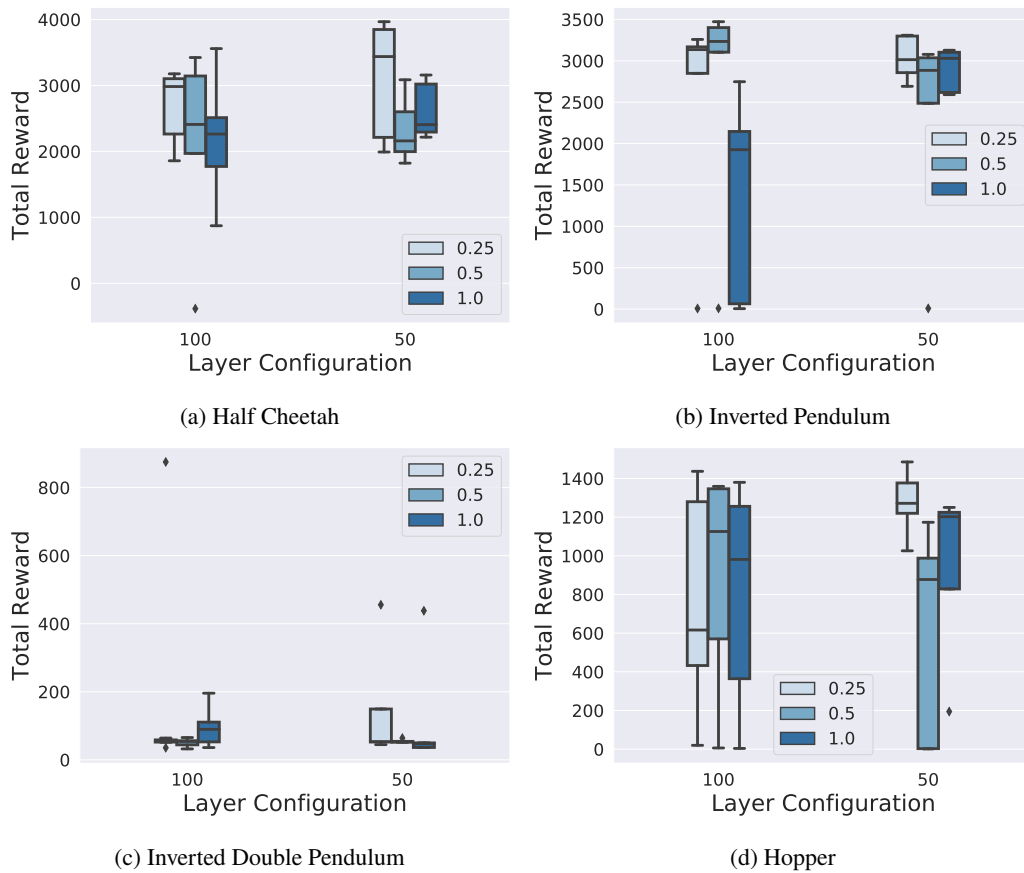


Figure 6.4: **Sparsity in NS methods:** Effects of sparsity levels in pure novelty search methods. It is evident that sparse representations perform better than dense encodings in most settings.

6.5.2 Ablation analysis

Effect of Sequence Length As discussed earlier, the chosen BC is a sequence of 2D agent positions at specific intervals. For instance, in the task of hopper, the sequence length is fixed to be 50. However, we would like to vary the sequence length and compare its influence on the final performance for both novelty-guided methods. We chose the Hopper task as a testbed for this analysis as it is one of the difficult tasks. Fig 6.6 (a), (b) shows the box plot showing the performance using different sequence lengths. As seen in Fig 6.6 (a), the performance of neighborhood methods is affected by the sequence length; more importantly, it does not scale well to longer sequence lengths. As seen in Fig 6.6 (b), for auto-encoders, the increased sequence length yields better results than shorter ones which show the scalability of function approximators using autoencoders.

Effect of archive size The neighborhood models rely on a fixed-sized archive set, using which the novelty scores are computed. Similarly, a fixed-sized behavior buffer is used to sample behaviors to fit the model for autoencoders. To understand the influence of the size of the archive set and the behavior buffer, we analyzed the Hopper task again. Fig 6.6 (c), (d) shows the box plot showing the performance using different sizes for the neighborhood models and autoencoders, respectively.

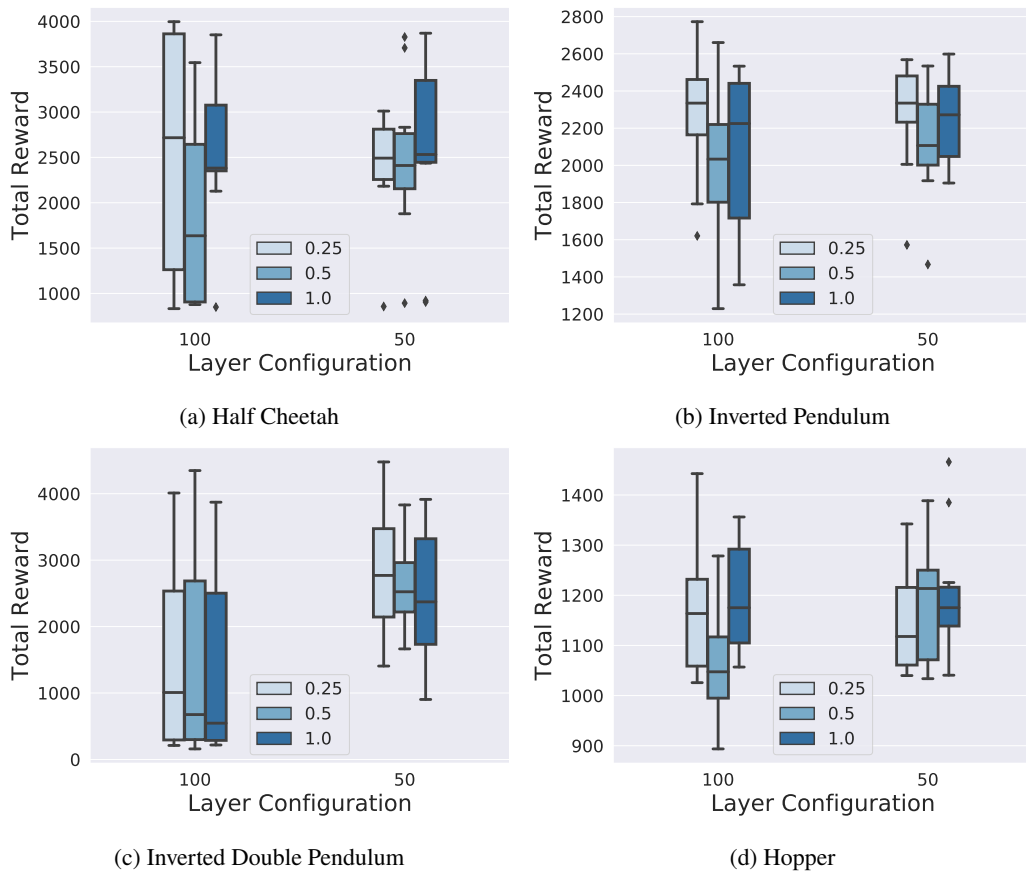


Figure 6.5: **Sparsity in Novelty-guided methods:** Effects of sparsity levels in novelty guided methods. It is evident that sparse representations perform better than dense encodings in most settings.

Clearly, the size influences the performance of neighborhood models, as seen in 6.6 (c). The larger the size, the better the performance indicating that accurate novelty scores can be computed using a larger-sized archive set. On the other hand, as it is observed in 6.6 (d), the size of the buffer does not influence the performance. In fact, even with a smaller buffer, the results are still competent. This shows that the autoencoders rely on encoding the behaviors in the network weights and rely less on stored behaviors in the buffer; therefore, it can scale better compared to neighborhood models. With further investigation and analysis, the behavior buffer can also be dropped entirely while the learning can be performed online using the sampled behaviors at each iteration. Regarding computation costs, the cost of computing novelty score is independent of buffer size K when using auto-encoders. In contrast, for the classic novelty search, it is $O(K)$, which we have avoided with the auto-encoder approach. Yet, this scalability and generalizability come at the cost of additional training time for auto-encoders.

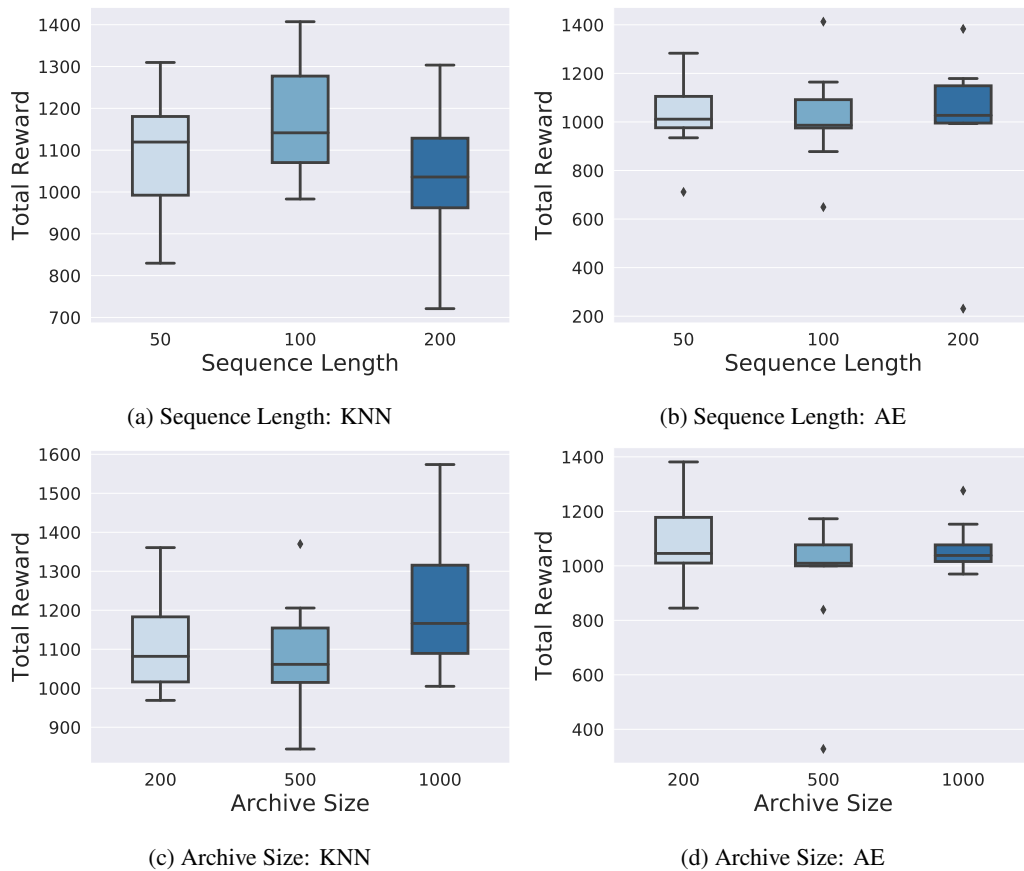


Figure 6.6: **Ablation analysis:** The plots (a),(b) show the effect of sequence length on the performance for KNN and AE methods. Similarly, plots (c),(d) show the effect of archive size on the performance of KNN and AE methods. It is observed that AE scales well to longer sequences and it requires only a small archive set

6.6 Conclusion

In this work, we aimed to overcome the limitations of NS methods in RL, which often suffer from scalability and generalizability issues. Our proposed approach, based on sparse behavior auto-encoders, offers a simple and scalable solution by assigning exploration bonuses to novel policies. Experimental results from continuous control tasks indicate that our approach presents a promising alternative to traditional NS methods that rely on nearest neighbors among known policies.

Our work opens up several avenues for future research. Firstly, employing BCs instead of reward functions could be a viable option in RL settings. Defining BCs is relatively straightforward compared to designing reward functions. However, the modeling and learning of appropriate BCs remain largely unexplored. Exploring the concept of informed reinforcement, which involves leveraging domain knowledge in the learning process, appears promising in this regard.

Secondly, the field of learning representations for sequences is actively studied in various domains of machine learning, such as natural language processing and speech recognition. Our work takes an initial step towards applying such methods in the context of NS approaches. We hope our findings inspire further research in learning efficient behavior representations.

Designing Policy Architecture with Domain Knowledge

In the broader context of this thesis, this chapter focuses on **RQ2**, concerning improving and exploration in RL. As discussed in Chapter 6, DRL has high complexity, especially in hard exploration problems. The previous chapter proposed accelerating RL by augmenting with novelty rewards that encouraged the agent to explore novel behaviors. In particular, we replaced classic novelty detectors using k-nearest neighbors with auto-encoders, resulting in better performance and generalization.

In addition to facing challenges related to high sample complexity, RL systems often lack interpretability. While deep neural networks excel at approximating value and policy networks, they tend to lack interpretability. As a result, there has been a growing interest in developing RL methods that offer interpretability [191, 192].

In this work, we focus on addressing these critical challenges in RL, precisely the high sample complexity and lack of transparency. We propose using domain knowledge to enhance transparency and construct policy networks that possess characteristics such as modularity, transparency, and efficiency in terms of data requirements. More specifically, by leveraging domain knowledge, we suggest decomposing the policy into components that are both learnable and non-learnable.

Intending to achieve this objective, we examine a robotic task called "reacher," wherein the goal is to train a 2D multi-joint arm to reach an unknown target using feedback. To address this task, we employ a computational biology-inspired model called Mean of Multiple Computations (MMC), which is a recurrent neural network capable of generating trajectories that lead to the target. Leveraging the dynamics of this network, we design our policy network in an informed manner to tackle the reacher task. By dividing the policy network into learnable and non-learnable components, we reduce the number of parameters that require RL training. The results indicate that learning such a modular network necessitates only one-tenth of the interactions compared to fully trained end-to-end networks.

In summary, our contributions in this chapter can be outlined as follows: firstly, we introduce the concept of using the Mean of Multiple Computations (MMCs) to design policy networks; secondly, we expand the application of MMCs by incorporating RL to train recurrent connections; and finally, we empirically validate the performance and sample complexity of MMC nets.

The chapter is based on the following publication [71]:

- R. Ramamurthy, C. Bauckhage, R. Sifa, J. Schücker and S. Wrobel, "Leveraging Domain Knowledge for Reinforcement Learning Using MMC Architectures", *Proceedings of International*

Conference on Artificial Neural Networks, 2019, URL: https://doi.org/10.1007/978-3-030-30484-3_48

The proposal of employing MMC networks to tackle the reacher task originated from the second author. Rajkumar Ramamurthy introduced the concept of decomposing the policy network using MMC and utilizing RL to train the parameters of MMC. The implementation of the code and execution of the experiments were carried out by Rajkumar. The initial draft of the paper was written by Rajkumar, which was subsequently subjected to thorough revisions by all coauthors to refine its content and structure.

The subsequent sections of this chapter are structured as follows: In Section 7.1, we delve into the challenges specific to RL that are pertinent to this research. Section 7.2 provides a concise overview of the reacher task and MMC networks. The concept of decomposing policy networks using MMC and their RL training is detailed in Section 7.3. Lastly, Section 7.4 presents the experimental results, comparing the performance of the proposed design with fully trained policy baselines.

7.1 Motivation

Reinforcement learning (RL) faces several limitations in addition to the challenges discussed earlier in Sections 5.1, 6.1. One major limitation is the high sample complexity involved in the training process. Even simple tasks require millions of interactions with the environment, and this issue becomes more pronounced for high-dimensional control tasks. To address this, modern RL approaches employ offline learning, where state-action transitions are stored in an extensive experience replay memory. This technique, exemplified in the work by [145] allows agents to learn from experiences and reduces the number of interactions needed for effective learning. Moreover, frameworks such as the Arcade Learning Environment [193], OpenAI Gym [80], and DeepMind Control Suite [194] offer high-quality simulated environments that enable RL agents to gather interactions on a large scale.

However, in the field of robotics, relying solely on simulation-based training poses challenges when deploying RL policies to real systems. Mismatches often arise between the simulated environment and the robot, necessitating adaptation. Some works focus on directly transferring learned policies from simulation to reality, but this approach is only sometimes successful due to the disparities between the two domains. Others explore domain adaptation techniques [195, 196] which aims to bridge the gap between simulation and reality. Additionally, specialized architectures [197, 198], have been developed to transfer low-level features and high-level control policies effectively. These endeavors provide frameworks that facilitate the transfer of learned behaviors from simulated environments to real-world robotics applications.

Another limitation of current RL systems is their limited interpretability regarding decision-making. Deep reinforcement learning employs function approximators like deep neural networks to acquire policy functions that establish the mapping between states and actions. While this approach is highly efficient, the resulting networks are predominantly black-boxes and cannot be employed in safety-critical systems, such as autonomous vehicles, where decision transparency is crucial for legal reasons. Consequently, there has been a growing interest in the development of interpretable deep reinforcement learning methods and the formulation of human-readable policies [191, 192].

This work addresses two critical challenges in RL systems: high sample complexity and the lack of transparency. Our approach involves incorporating expert knowledge into the models to enhance

transparency and create more reliable systems. Specifically, we integrate domain knowledge into the architecture of policy networks to develop policy networks that are transparent, modular, and efficient regarding data requirements.

To illustrate our approach, we focus on a reacher task where an agent, a 2D multi-joint arm, needs to reach an unknown target using feedback and rewards. Initially, we revisit a model based on the principle of the "mean of multiple computations" (MMC) [199], which has previously shown success in modeling biological systems such as analyzing walking behaviors of six-legged insects [200] and landmark-based navigation [201]. The MMC Net offers several appealing features. Firstly, it is a simple recurrent neural network capable of generating geometrically accurate solutions by predicting target-reaching trajectories. Secondly, it can be easily extended to other tasks, such as a 3D 6-DoF arm, with minimal domain-specific knowledge. Thirdly, it comprises two components: a linear part that can be numerically optimized based on rewards and a readily available non-linear component. This modular design facilitates the transfer of learned policies.

Our contributions in this work are threefold: (i) we introduce the idea of MMCs for constructing modular policy networks using expert knowledge, (ii) we enhance MMC nets by adapting recurrent connections through reward signals and apply them to solve the reacher task, and (iii) we empirically compare the performance and sample complexity of MMC nets against end-to-end policy networks.

Related Work Previous work on incorporating prior knowledge into RL has primarily centered on designing reward signals that consider multiple sub-goals and intermediate rewards based on relative progress [168, 202, 203]. These supplementary rewards, in addition to sparse environmental rewards, accelerate the learning process. More recent research [176, 178, 204, 205] explores the integration of hand-designed behavioral characteristics (BCs) to introduce a novelty objective alongside the existing reward objective.

Another approach to leverage domain expertise is through learning from expert demonstrations. Imitation Learning (IL) focuses on developing agents that replicate expert behavior, ranging from object manipulation using video demonstrations [206, 207] to complex tasks like operating doors or utilizing tools [208]. Alternatively, learning reward functions from human-generated demonstrations using reward shaping and inverse reinforcement learning (IRL) has also been pursued [170]. Recent studies [209, 210] combine transfer learning and human demonstrations, utilizing prior task knowledge to bootstrap related tasks for adaptation. Approaches involving human interaction [171, 172] rely on obtaining positive and negative feedback from human trainers to learn desired optimal behavior. Another popular approach to incorporate knowledge is QS learning [211], which combines Q-learning with expert-specified state-action similarities to update state-value estimates and has shown better performance compared to standard temporal difference methods.

Previous work on building modular policy networks has focused on transfer learning by sharing network layers across different tasks [212, 213]. Notably, interesting research in autonomous systems [214] utilizes decomposed networks with perception and controller modules, demonstrating deployability to various vehicles and environmental conditions.

In contrast to these approaches, our focus is on designing policy architectures using domain knowledge. Specifically, we concentrate on modular networks with hand-designed components rather than learning them end-to-end. Importantly, by designing components as MMC networks, we propose a simple method that learns from a reduced number of interactions.

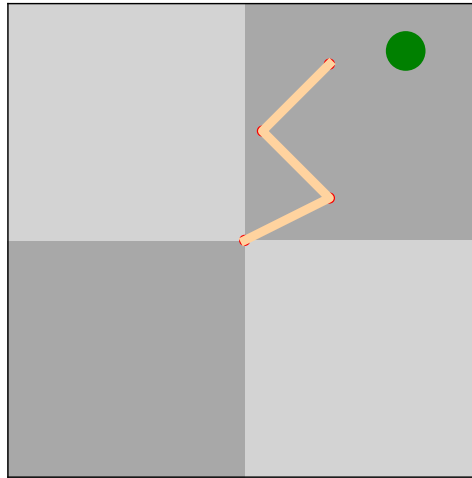


Figure 7.1: **Reacher environment:** A simple three-segmented robot arm with a randomized target. The goal is to plan a sequence of actions using feedback and rewards so that the tip of the arm touches the target.

7.2 Background

In this section, we first present a short introduction to the reacher task, followed by MMC nets that were initially proposed in [199].

7.2.1 Reacher Task

We will provide a brief overview of the reacher task, which serves as the test bed for our methodology. In this task, we utilize a planar three-segmented robotic arm with an end effector, with the objective of reaching a target position. The target position is randomly generated at the beginning of each episode, and its location remains unknown to the agent throughout the task. The agent’s observation consists of joint angles and a feedback signal, providing information about the arm’s current configuration. The agent’s actions correspond to desired joint angles for the arm’s trajectory. The negative norm of the distance between the end effector and the target position determines the reward. Although our setup shares similarities with the simulation environments offered by OpenAI Gym [80] and DeepMind’s control suite [194], there are slight differences regarding observations and actions.

7.2.2 Linear MMC Networks

Consider a simple manipulator (a robotic arm) with three joints operating in a 2D space as shown in Figure 7.2. Orientation and length of the three segments of the arm are denoted as vectors \vec{L}_1 , \vec{L}_2 and \vec{L}_3 in a Cartesian coordinate system with the origin located at the shoulder joint. Two additional vectors \vec{D}_1 , \vec{D}_2 connect the shoulder and the second joint and the first joint and the end effector; a vector \vec{R} connects the shoulder and the end effector. This setting provides the following over-determined

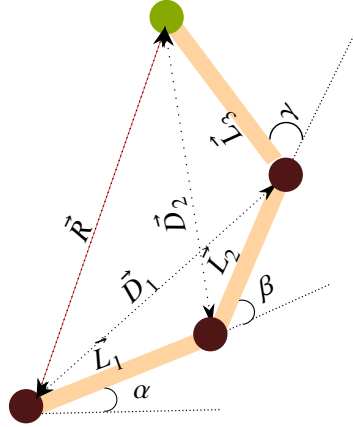


Figure 7.2: **Robotic arm:** A robotic manipulator consisting of three segments denoted as \vec{L}_1 , \vec{L}_2 and \vec{L}_3 . The relative angles at the joints are denoted as α , β , and γ . The vector pointing to the end-effector (in green) is described as \vec{R} .

system of equations

$$\begin{aligned}
 \vec{L}_1 + \vec{L}_2 + \vec{L}_3 - \vec{R} &= 0 \\
 \vec{L}_2 + \vec{R} - \vec{D}_2 - \vec{D}_1 &= 0 \\
 \vec{L}_1 - \vec{L}_3 + \vec{D}_2 - \vec{D}_1 &= 0 \\
 \vec{L}_1 + \vec{L}_2 - \vec{D}_1 &= 0 \\
 \vec{L}_2 + \vec{L}_3 - \vec{D}_2 &= 0 \\
 \vec{L}_3 - \vec{R} + \vec{D}_1 &= 0 \\
 \vec{L}_1 - \vec{R} + \vec{D}_2 &= 0
 \end{aligned} \tag{7.1}$$

In the system of equations 7.1, each vector appears exactly four times and according to the MMC principle, we can write every vector as a mean of the corresponding entries in the four equations. For instance, \vec{L}_1 can be computed as

$$\vec{L}_1 = \frac{1}{4}(-2\vec{L}_2 + 2\vec{R} - 2\vec{D}_2 + 2\vec{D}_1) \tag{7.2}$$

Given a desired target position \vec{R} , this setting is an instance of inverse kinematics in which the goal is to solve for the values $\vec{L}_{1,2,3}$ in an iterative manner. Thus, the mean values are fed back as input to the system for the next iteration until the system relaxes to the desired position. Note that \vec{R} is kept constant throughout and its mean value is suppressed during the feedback. Furthermore, self-excitations are introduced via damping factors d_1, d_2, d_3 , with the goal of suppressing oscillations. In summary, the system represents a simple form of a recurrent neural network whose dynamics can be written as

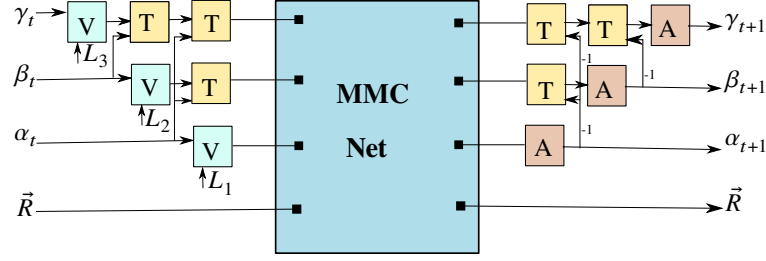


Figure 7.3: **MMC architecture**: A sketch of MMC architecture described in this paper; it consists of components for MMC Net sandwiched by forward and inverse transformation components. The computed joint angles are fed back into the system until the end effector relaxes to the desired target

$$\begin{aligned}\vec{L}_1(t+1) &= \frac{1}{4+d_1} [d_1 \vec{L}_1(t) - 2\vec{L}_2(t) + 2\vec{R} - 2\vec{D}_2(t) + 2\vec{D}_1(t)] \\ \vec{L}_2(t+1) &= \frac{1}{4+d_2} [-2\vec{L}_1(t) + d_2 \vec{L}_2 - 2\vec{L}_3(t) + 2\vec{D}_2(t) + 2\vec{D}_1(t)] \\ \vec{L}_3(t+1) &= \frac{1}{4+d_3} [-2\vec{L}_2(t) + d_3 \vec{L}_3 + 2\vec{R} + 2\vec{D}_2(t) - 2\vec{D}_1(t)]\end{aligned}$$

The coefficients of this system of equations can be collected in a matrix θ (scaling factors are ignored for readability)

$$\theta = \begin{bmatrix} d_1 & -2 & 0 & 2 & -2 & 2 \\ -2 & d_2 & -2 & 0 & 2 & 2 \\ 0 & -2 & d_3 & 2 & 2 & -2 \end{bmatrix} \quad (7.3)$$

which will later be subject to a reinforcement learning policy.

During the relaxation, the lengths of the arm vectors may change, which is, of course, undesirable. Instead, the system should find a suitable joint configuration to reach the target while leaving the arm lengths constant. This can be achieved via a non-linear model.

7.2.3 Non-linear MMC Networks

In order to keep the lengths of the three arm segments fixed, we follow [199] and describe the arm segments in terms of their lengths L_1 , L_2 and L_3 and orientations α , β , γ . At time t , α_t , β_t , γ_t are given as input to the system and the outputs are the predicted orientations α_{t+1} , β_{t+1} , γ_{t+1} that will serve as the input for the next iteration.

The architecture is divided into three components, a forward pass including non-linear transformations, an MMC network component and an inverse component. A sketch of the non-linear MMC architecture is shown in Fig. 7.3.

In the forward pass, the joint angles are first transformed to arm vectors using the transformations \mathbf{V} and \mathbf{T} . \mathbf{V} takes a constant length of an arm segment L and a joint angle α as inputs to compute the

corresponding arm vector

$$\vec{L} = L \begin{bmatrix} \cos \alpha \\ \sin \alpha \end{bmatrix} \quad (7.4)$$

The transformation \mathbf{T} takes an arm vector \vec{L} and an angle α as input and applies a corresponding rotation

$$\vec{L}' = \begin{bmatrix} L'_x \\ L'_y \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} L_x \\ L_y \end{bmatrix} \quad (7.5)$$

The transformations are applied in sequence; for instance, γ_t is transformed by $(\mathbf{V}, \mathbf{T}, \mathbf{T})$, where the latter two transformations are necessary because the joint angles are all relative to their own axis [Fig. 7.2].

The output of the forward pass is the arm vectors $\vec{L}_1(t), \vec{L}_2(t), \vec{L}_3(t)$ which are then fed to the linear MMC Net to compute the predicted arm vectors for the next time step, $\vec{L}_1(t+1), \vec{L}_2(t+1), \vec{L}_3(t+1)$.

Finally, in the inverse component, these vectors are converted back to the corresponding arm orientations $\alpha_{t+1}, \beta_{t+1}, \gamma_{t+1}$ by using the inverse of \mathbf{T} and \mathbf{A} , which is the inverse of \mathbf{V} and computes the arm orientation given an arm vector

$$\alpha = \arctan(L_y, L_x) \quad (7.6)$$

In summary, a non-linear MMC architecture can be decomposed into three components (i) forward components that perform feature selection, (ii) a linear MMC net that plays a role in trajectory prediction, and (iii) inverse controllers that control the actuators.

7.3 Learning a Modular Policy Network

So far, we have discussed planning a trajectory toward a target position using MMC networks. However, in an RL setting, the agent cannot access target positions directly. For instance, in the reacher task, the agent instead receives a feedback vector which is usually the distance between the end-effector and the target denoted as \vec{R}_t . Thus, it can be treated as a proxy for the actual target position. In this case, too, the coefficients of the MMC Net have to be solved, but instead of solving by hand, they can now be trained based on reward signals. To this end, we define the policy π as a MMC policy network π_θ with weights θ which can be then adapted by a policy learning method.

7.3.1 Approach For Solving Reacher Tasks

The main idea is to decompose our MMC policy network π_θ into linear and non-linear components and to adapt only the linear component while non-linear components are used off-the-shelf. At time t , observation o_t is processed as

1. **Forward Component:** This part of the network converts the joint angles contained in the given observation $\vec{o}_t = (\alpha_t, \beta_t, \gamma_t, \vec{R}_t)$ into input vectors $\vec{L}_1(t), \vec{L}_2(t)$ and $\vec{L}_3(t)$ via transformations \mathbf{V} and \mathbf{T} .
2. **MMC Component:** This is a linear MMC net with weights $\vec{\theta}$, which are adapted by a learning method through interaction with the environment. This component then acts as a planning module that predicts the next point in the trajectory $\vec{L}_1(t+1), \vec{L}_2(t+1)$ and $\vec{L}_3(t+1)$.

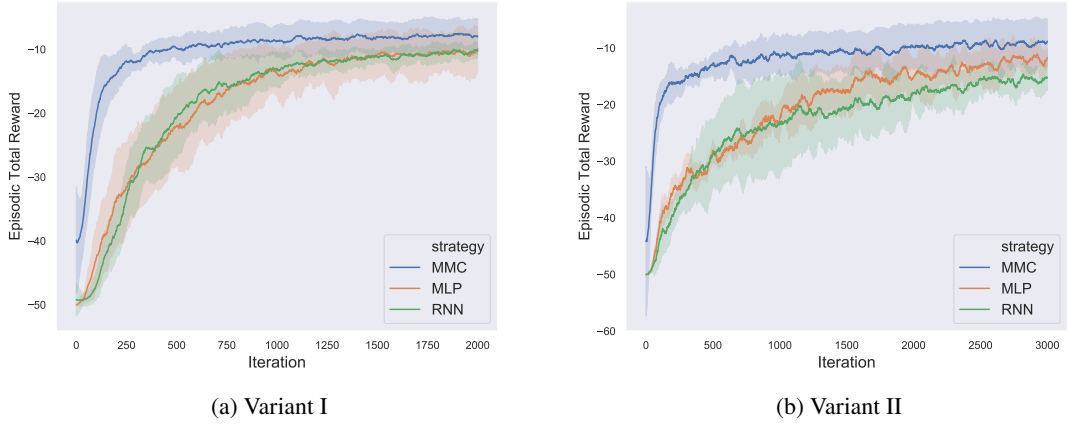


Figure 7.4: **Learning curves:** Evolution of total episodic reward in the learning of Reacher task with two variants; (a) Variant I: in which target position is given in the observation and (b) Variant II: in which target position is not directly available, but as a feedback signal in the observation. As observed, MMC networks outperform end-end approaches in both variants.

3. **Inverse Component:** This complements the forward component. Outputs of the MMC net, such as the predicted arm vectors $\vec{L}_1(t+1)$, $\vec{L}_2(t+1)$ and $\vec{L}_3(t+1)$, will be converted back to joint angles α_{t+1} , β_{t+1} , γ_{t+1} resulting in an action vector \vec{a}_t

Using such a modular system, forward- and inverse components can be shared between similar tasks or can be engineered using domain knowledge to solve different tasks. Only the MMC net needs to be trained for the given task. Most importantly, the task can be solved efficiently by designing the inputs passed to MMC net.

7.3.2 Learning Method

Our goal is to find the optimal value for θ such that the expected cumulative reward is maximized $\theta^* = \operatorname{argmax}_{\theta} J(\theta)$ where $J(\theta) = \mathbb{E}_{\pi_{\theta}} [G_T]$. To train θ of the MMC net, we use a stochastic approximation method, namely Simultaneous Perturbation Stochastic Approximation (SPSA) discussed in Section 5.2.

7.4 Experimental Results

In this section, we demonstrate the performance of the proposed MMC policy network using a simulation environment as shown in Fig. 7.1. In particular, we focus on benchmarking performance metrics such as episodic total reward and sample complexity. Furthermore, we consider **Variant I**, where the agent can observe the desired target position directly, and **Variant II**, where the agent receives a feedback signal based on the difference between end effector position and target position.

We evaluate our MMC policy network against end-end policy networks on learning a deterministic policy that predicts the desired joint angles. Since angular velocities are not included in the observation, we also consider a recurrent neural network as a baseline, treating it as a partially observable task.

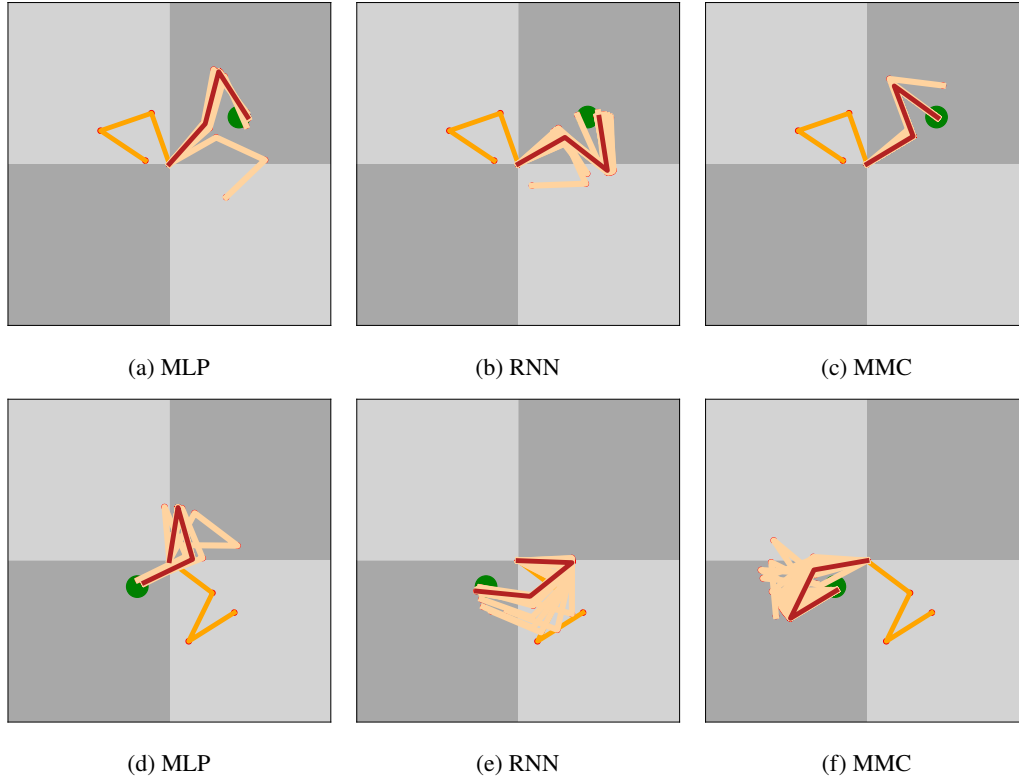


Figure 7.5: **Agent behaviors:** Initial and final arm positions are shown in orange and red respectively; (a), (d) trajectories toward the target with MLP agent; (b), (e) trajectories toward the target with RNN agent; and (c) and (f) with MMC agent. The MMC agent quickly approaches the target and is able to maintain its position around the target throughout the episode

Implementation details: In our experiments, the length of each episode is limited to 30 time steps; Therefore the goal is not just to reach fast but also to maintain the end effector around the target. Our MMC net is implemented as a linear model consisting of 8 input neurons and 6 output neurons. Our baselines are (i) a fully connected multi-layer perceptrons (MLP) with 3 hidden layers with tanh activation functions, (ii) a recurrent neural network (RNN) of gated-recurrent units (GRU) with 3 hidden layers.

All networks are trained using SPSA parameters $a = 0.01$, $A = 10$, $\eta = 0.1$, $c = 0.01$, $\tau = 0.1$. Furthermore, gradients are smoothed via RMSprop with a decay rate of 0.9. Finally, for stable learning, several pairs of perturbations K are evaluated to compute the average gradient for each step and are tuned based on the network size.

7.4.1 Learning Performance

In order to evaluate the performance of the MMC network quantitatively, we consider the learning of an optimal policy using MMC networks and comparing them to baseline techniques. Figure 7.4 shows the evolution of total episodic reward (averaged over different random seeds) in the learning process (the higher the better) for both variants of the task. These results suggest that the MMC policy

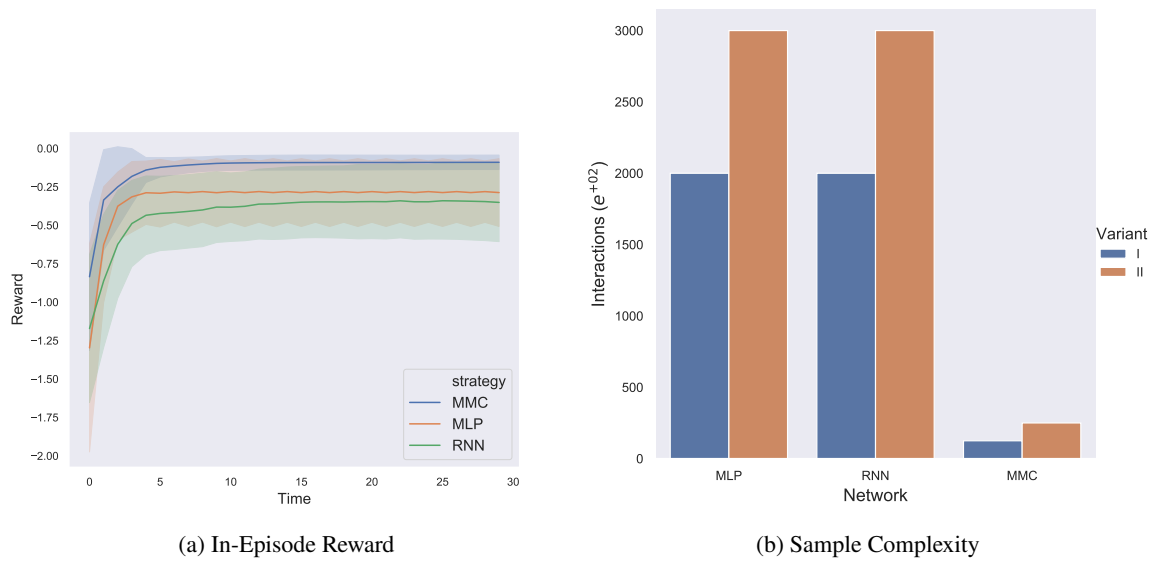


Figure 7.6: **Sample Complexity**: (a) Evolution of reward at each step averaged over 100 runs; showing MMC networks quickly relax to high reward gaining trajectories as soon as an episode starts. (b) a summary of total interactions with the environment to learn an optimal policy with different networks and variants of the Reacher task; MMC networks are 10 times sample-efficient than end-end approaches

Variants	MLP	RNN	MMC
Variant I	-7.37	-9.06	-3.18
Variant II	-10.01	-12.64	-4.12

Table 7.1: **Performance Comparison**: Final Performance in terms of total episodic reward (higher is better) of best policies with different networks averaged over 100 episodes; MMC nets outperform other end-end architectures

network can learn optimal behaviors in only a few iterations. MMC networks also perform similarly well in both variants of the task, which shows the robustness of the proposed modular architecture. On the other hand, the end-to-end approaches seem to have more difficulties when learning a suitable policy for variant II than in the case of variant I. Finally, we compare the performance of the best policies of each network in 100 episodes and summarize their average performance in Table 7.1.

7.4.2 Behaviors

In order to compare the behavior of agents, we ran the task with two random seeds with final trained best policies from each network and plotted their behaviors observed throughout the episode in Fig 7.5. Fig 7.5 (a), (d) show the behavior of agents whose policies were learned using multi-layer perceptrons; (b), (e) reflect policies learned by gated recurrent units, and, finally, (c) and (f) were obtained from MMC policy networks.

These examples suggest that agents trained with the MMC policy network quickly approach the

target, thus gaining high rewards, whereas the other methods slowly adapt to the feedback signal. Fig. 7.6 (a) captures similar behavior in terms of reward obtained at each step in an episode. By averaging over 100 runs, we observe that MMC networks are able to stabilize themselves to high reward positions from very early in the episode (approx $t = 2$) when compared to end-to-end approaches.

7.4.3 Sample Complexity

To evaluate the sample complexity of policy learning with MMC networks, we consider the number of interactions with the environment required for achieving the best performance of the baseline models. To this end, we consider the learning curves in Figure 7.4. For variant I, the baseline models required 2000 iterations to reach their best performance, whereas a MMC achieved similar performance within only about 250 iterations. While training the baseline networks, gradient updates in each iteration are computed after evaluating 100 pairs of perturbations. Similarly, with our MMC policy network, an update is performed after evaluating 50 pairs of perturbations. Therefore, a total of 200.000 interactions is required for the baselines. However, the MMC approach required only $250 \times 50 = 12.500$ interactions with the data. This shows that our MMC network is 16 times more data-efficient than the end-to-end approaches. A similar analysis for variant II shows a 12 fold speed up for the MMC networks. These results are summarized in Fig. 7.6 (b).

7.5 Conclusion and Future Work

We introduced a novel approach to RL by utilizing MMC architectures. Our proposed method enables domain experts to incorporate their knowledge into policy networks by designing and constructing modular components tailored to specific tasks. Through our experiments, we demonstrated that the suggested policy network learns superior policies for variations of the "reacher task" in a more efficient manner, requiring fewer samples compared to baseline techniques. Furthermore, it achieves a remarkable speed increase of over 10 times compared to competing architectures.

The resulting system adopts a modular structure consisting of fixed non-linear transformations and a trainable linear policy network. This decomposition offers an advantage, as the explicit transformations of the non-linear part are known, which is often preferable in practical applications over end-to-end systems composed of neural networks that lack transparency. Consequently, the integration of domain knowledge not only reduces the complexity of obtaining samples but also represents a significant step toward creating more interpretable systems.

MMC networks can be extended in various ways. Firstly, it should be possible to engineer forward and inverse components for other continuous control tasks, such as bi-pedal walking, hopping, swimming, and more, by leveraging expert domain knowledge, although it is not straightforward, it might require extensive engineering.

Secondly, the inherent recurrent nature of the approach, coupled with the ability to predict future states in a trajectory towards a target, allows for trajectory unrolling over time. This unrolling can be utilized for planning extended actions, i.e., actions performed over prolonged durations, facilitating the development of hierarchical policies.

Part III

Practical Structured Prediction using Reinforcement Learning

In this section of the thesis, the focus shifts to addressing practical challenges of SP tasks namely **RQ3** and **RQ4**.

Chapter 8 presents the toolkit *NLPGym* that provides simulated environments for casting typical NLP tasks as sequential decision-making tasks. Experimental results for six tasks using various RL algorithms are presented that serve as baselines for future research.

Chapter 9 focuses on the challenges of RLHF. First, it presents a new framework *RLALMs* which provides easy tools to fine-tune LLMs on arbitrary reward functions. Then, it proposes a novel on-policy RL algorithm that is well-suited for NLP tasks. Finally, a comprehensive benchmark consisting of seven generative NLP tasks is presented, which highlight crucial design choices concerning reward functions, initialization of policies and more.

NLPGym - A toolkit for training RL agents on Natural Language Processing Tasks

In Part I of this thesis, our focus was on sequential decision and prediction (SDP) tasks, specifically exploring the application of echo state networks (ESNs) as random context encoders for three tasks: sequence memorization (Chapter 3), named-entity recognition (Chapter 4) and control tasks (Chapter 5). We demonstrated that ESNs can effectively solve these tasks without adapting the recurrent connections.

Moving on to Part II, our focus shifted towards addressing challenges in reinforcement learning (RL). In Chapter 6, our emphasis was on improving exploration, where we proposed a straightforward method that guides RL using auto-encoders to provide auxiliary novelty rewards. Similarly, Chapter 7 presented a technique to decompose policy networks based on an architecture from computational biology, enhancing both parameter and data efficiency while increasing the interpretability of deep neural networks.

Returning to SP tasks, we will examine the close relationship between SP and RL. In particular, we address the limitations of training SP tasks using supervised learning objectives such as maximum likelihood estimation (MLE). These approaches suffer from *exposure bias* or *data mismatch*, as the model is exposed to inputs that differ from those seen during training when incorrect outputs are predicted. This is mostly because prediction in each step is fed as inputs in subsequent steps and the mistakes accumulate over the inference process. Moreover, they also suffer from *metric mismatch*, where models are evaluated on metrics different from the ones they were trained on. For instance, a text summarization system is evaluated for its consistency and factual correctness.

To tackle these issues, one alternative is to cast SP as RL, enabling models to be trained directly to optimize application-specific metrics. For example, maximizing BLEU or ROUGE scores can solve machine translation tasks. Further, RL facilitates learning from the model's own generated sequences during training, bridging the gap between the training and testing distributions.

Most NLP tasks can now be cast as a sequential-decision making problem and can be solved by RL. While RL is increasingly applied to these tasks [74–79], there is a lack of simulated textual environments available for researchers to consistently apply and benchmark RL on natural language processing (NLP) tasks. To address this gap, we introduce *NLPGym*, an open-source Python toolkit that provides interactive textual environments for standard NLP tasks such as sequence tagging, multi-label classification, and question answering. The proposed toolkit is modular, easy to use

and compatible with standard RL frameworks such as stable-baselines. Further, new tasks, reward functions, datasets can be easily added by users. We also present experimental results for six tasks using various RL algorithms, serving as valuable baselines for future research. The chapter is based on the publication by [89],

- R. Ramamurthy, R. Sifa and C. Bauckhage, “NLP Gym – A Toolkit for Evaluating RL agents on Natural Language Processing Tasks”, *Proceedings of Wordplay Workshop in NeurIPS 2020*, URL: <https://doi.org/10.48550/arXiv.2011.08272>

Rajkumar Ramamurthy proposed the initial framework that casts SP into RL. Rajkumar Ramamurthy took the lead role and carried out the implementation, experimental design and training of the algorithms. He also drafted the initial version of the paper, which was later revised extensively by all authors. All authors actively contributed to the discussion of the work throughout the entire process.

The remainder of the chapter is organized as follows: Section 8.1 provides a brief motivation for the use of RL in the field of NLP, followed by Section 8.2 introducing the *NLP Gym* toolkit, highlighting its usage and supported tasks. In Section 8.3, the toolkit is utilized to obtain a benchmark of the provided tasks using different RL algorithms and reward functions, serving as baseline results for future research.

8.1 Motivation

Comprehension and communication using natural language is a major characteristic of human intelligence. In the domain of artificial intelligence, to assist humans with language-oriented tasks, artificial agents have to acquire similar natural language understanding (NLU) capabilities. In this regard, there has been tremendous progress and machines have become well-versed in natural language processing (NLP) tasks such as language modeling [42, 43], information extraction [215, 216], text summarization [217, 218], or question answering [219, 220].

Deep reinforcement learning (DRL) has become popular in the NLP domain as most tasks can now be formulated as a sequential-decision making problem [28, 72]. This is auspicious for several reasons. First, RL is inherently interactive, allowing training agents through rewards to optimize human preferences directly. Second, learning based on rewards allows the agents to be trained to maximize application-specific metrics such as F1 or ROUGE scores which are generally non-differentiable and cannot be optimized by traditional supervised learning approaches.

Typical applications of DRL in NLP include: (a) sequence tagging [28, 72] where RL is used to solve structured prediction tasks such as named entity recognition and part of speech tagging; (b) text summarization [74–76] where agents select sentences to be included in summaries; (c) question answering [77] where agents rank and select relevant paragraphs and it is interesting to note that question answering can be cast as an interactive game [78]; (d) information extraction [79] where agents query and extract information from external resources to extract information; (e) solving text-based games [83, 221, 222].

Despite their increased use in NLP, there are no frameworks for testing RL agents on standard NLP tasks. In other words, it seems that the application of RL to NLP suffers from lack of availability of open-source frameworks which are similar to Gym [80], Malmo [81], Arcade Learning Environment (ALE) [82], TextWorld [83], or Baby AI [84] which accelerated research on solving robotic tasks, mine-craft playing, ATARI game-playing, text-based games and ground language agents.

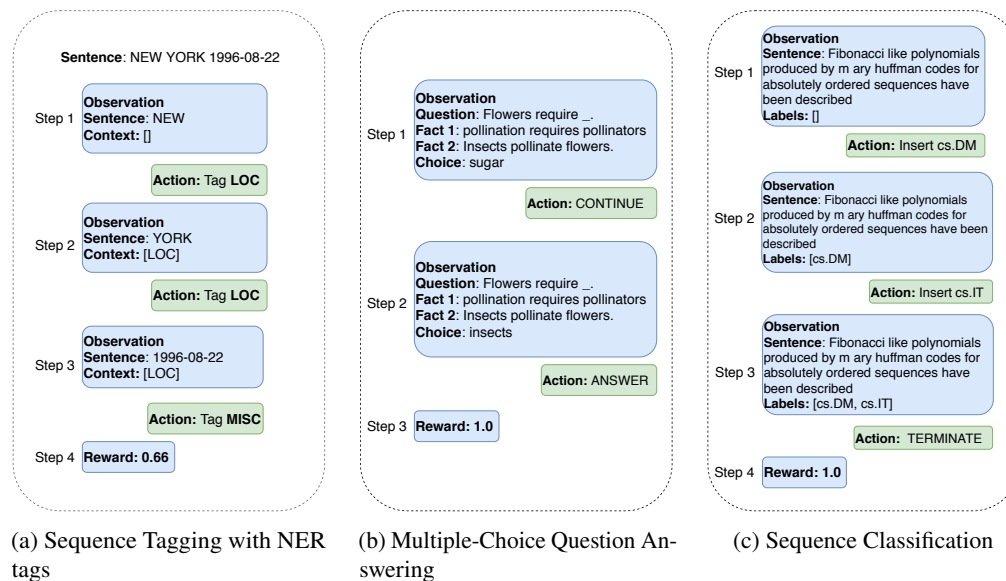


Figure 8.1: **NLP Gym Environments**: An overview of sample episodic interactions of agent-environments for three different tasks. (a) tagging a sequence with NER tags; (b) answering multiple-choice questions; (c) generating label sequence for a given sentence

In this work, we therefore present NLP Gym, a toolkit to bridge the gap between applications of RL and NLP. This aims at facilitating research and benchmarking of DRL applications on natural language processing tasks. Our toolkit provides interactive environments for standard NLP tasks such as sequence tagging, question answering, and sequence classification. The environments offer standard RL interfaces and therefore can be used together with most RL frameworks such as baselines [223], stable-baselines [224], and RLLib [225]. Furthermore, the toolkit is designed modularly, providing flexibility for users to extend tasks with their custom data sets, observations, and reward functions.

To illustrate the capabilities of NLP Gym, this paper also provides examples and detailed experimental results on the above tasks using different RL algorithms, featurizers and reward functions. These results will serve as baselines for further research in this area.

Indeed, in future work, we plan to include environments for text summarization, text generation and machine translation tasks as they can also be formulated as MDP problems. We hope that NLP Gym becomes an ever-growing test-bed for testing agents for learning language and understanding.

Related Work We present a short survey of related work concerning frameworks for solving language-oriented tasks. They fall under two main categories: *grounded language learning* and *text-based games*. Notable works in *grounded-language learning* include 3D simulated environments [226, 227] for developing language-informed agents. Building upon this idea of specifying goals using language, BabyAI [84] enables agents to interact and manipulate objects in environments. In the category of *text-based games*, TextWorld [83] offers gym-style environments for solving text-based adventure games. Likewise, Jericho [228] supports various interactive fiction games. Focussing on dialogue research, LIGHT [229] provides a platform for studying conversational agents that can interact with characters and objects in a large-scale text-based adventure game. Since most toolkits only focus on language-informed tasks and text-based games, there is a need for equivalent frameworks

for developing agents to solve NLP tasks, which we address with our work of NLP Gym.

8.2 NLP Gym Toolkit

The NLP Gym toolkit consists of interactive environments for learning natural language processing tasks. It includes environments for three tasks: sequence tagging, question answering, and multi-label sequence classification. Each task is formulated as a Markov Decision Process (MDP) defined by a tuple: $\langle S, A, T, R \rangle$ where S is a set of states, A is a set of actions, R is the reward function, and T is the transition probability. Figure 8.1 shows an overview of sample episodic interactions of agent-environment for the provided tasks.

The toolkit is implemented in Python, extending on OpenAI Gym’s API with *step()*, *reset()* and *render()* functions for observing and interacting with the environment (see Appendix B.1). Due to its compliance with Gym’s API, NLP Gym can be easily integrated with most RL frameworks [223–225]. Additionally, each environment has default implementations of reward functions, and observation featurizers (see Appendix B.2) and can be used out of the box. Therefore, training a DQN agent can be done with less than 10 lines of code (see Listing 8.1). Moreover, the toolkit is designed in a modular fashion, allowing users to plug-in custom components such as reward functions, featurizers for observation and datasets. For more details, we refer to Appendix B.3, which elaborates on their extensibility using code snippets.

Listing 8.1: Training DQN agent using NLP Gym

```

from nlp_gym.data_pools.custom_seq_tagging_pools import UDPosTaggingPool
from nlp_gym.envs.seq_tagging.env import SeqTagEnv
from nlp_gym.envs.seq_tagging.reward import EntityF1Score
from stable_baselines.deepq.policies import MlpPolicy as DQNPoly
from stable_baselines import DQN
from stable_baselines.common.env_checker import check_env
from rich import print

# data pool
data_pool = UDPosTaggingPool.prepare(split="train")

# reward function
reward_fn = EntityF1Score(dense=True, average="micro")

# seq tag env
env = SeqTagEnv(data_pool.labels(), reward_function=reward_fn)
for sample, weight in data_pool:
    env.add_sample(sample, weight)

# check the environment
check_env(env, warn=True)

# train a MLP Policy using DQN
model = DQN(env=env, policy=DQNPoly, gamma=0.99, batch_size=32, learning_rate=5e-4,
            double_q=True, exploration_fraction=0.1,
            prioritized_replay=False, policy_kwargs={"layers": [100, 100]},
            verbose=1)
model.learn(total_timesteps=int(1e+4))

```

8.2.1 Tasks

Sequence Tagging (ST): Sequence tagging is one of the core tasks in NLP for natural/spoken language understanding (NLU/SLU) and typical tasks include named-entity recognition (NER) and part-of-speech (POS) tagging. Formally, given a sentence with words (w_1, w_2, \dots, w_t) , the task is to generate its associated label sequence (l_1, l_2, \dots, l_t) . This task can be cast as an MDP [72] in which the sentence is parsed in left-to-right order, one word at a time. The environment state s_t at step t consists of w_t and predicted labels for previous words. However, the agent’s observation o_t at each step is just the current word w_t and context (predicted label of the previous word l_{t-1}), which renders the environment partially observable. The available actions are to *TAG* the current word with one of the possible labels and the action space is thus discrete. On performing any action, the transition function is deterministic and trivial, that the tagged label is added to the label sequence. The reward is computed as the entity-level F1-score between actual and predicted labels. It comes in two flavors; sparse (given at the end of the episode) and dense (difference in scores between consecutive steps).

Multiple-choice Question Answering (QA): Multiple-choice question answering (QA) is at the core of machine reading comprehension [230, 231]. The task of QA is to answer a given question q by selecting one of the multiple choices c_1, c_2, \dots, c_t . Besides, questions are often accompanied by supporting facts f , which contain further context. Selecting the correct option out of all choices can be considered as a sequential decision-making task. Each episode spawns with a question to be answered. At each step t , the question q , supporting facts f , and one of the multiple choices c_t are given as the observation to agents. Given this observation, agents’ actions are binary whether to answer *ANS* or to continue with the next choice *CONT*. Consequently, on choosing the action *ANS*, the episode terminates and the last observed choice is considered the final answer. On the other hand, on selecting *CONT*, the environment moves to the next choice and presents its corresponding observation. The reward is given only at the end of the episode, either 0 or 1, based on the selected choice’s correctness.

Multi-label Classification (MLC) Multi-label classification is a generalization of several NLP tasks such as multi-class sentence classification and label ranking [232]. The task of multi-label classification is to assign a label sequence l_1, l_2, \dots, l_n to the given sentence. In information retrieval, this task corresponds to label ranking when preferential relation exists over labels. Likewise, the task reduces to a simple multi-class classification when any label sequence’s maximum length is at most one. In any case, generating this label sequence can be cast as a sequential decision-making task. Each episode begins with a sentence and an empty label sequence. The observation at each step t is the given sentence and generated label sequence until t . Similar to sequence tagging, available actions are to *INSERT* one of the possible labels. Moreover, agents can terminate the episode using the *TERM* action. The reward is chosen as the F1-score between actual and predicted label sequences, either sparse or dense. For ranking problems, users can provide their desired ranking metric [233] as the reward function.

8.2.2 Towards Interactive Learning

Each environment relies on annotated data points that are used in generating episodes. NLP Gym supports injecting samples into the environment dynamically at any point in time. This feature offers the flexibility to develop algorithms in two settings; *batch* setting in which all of the data points

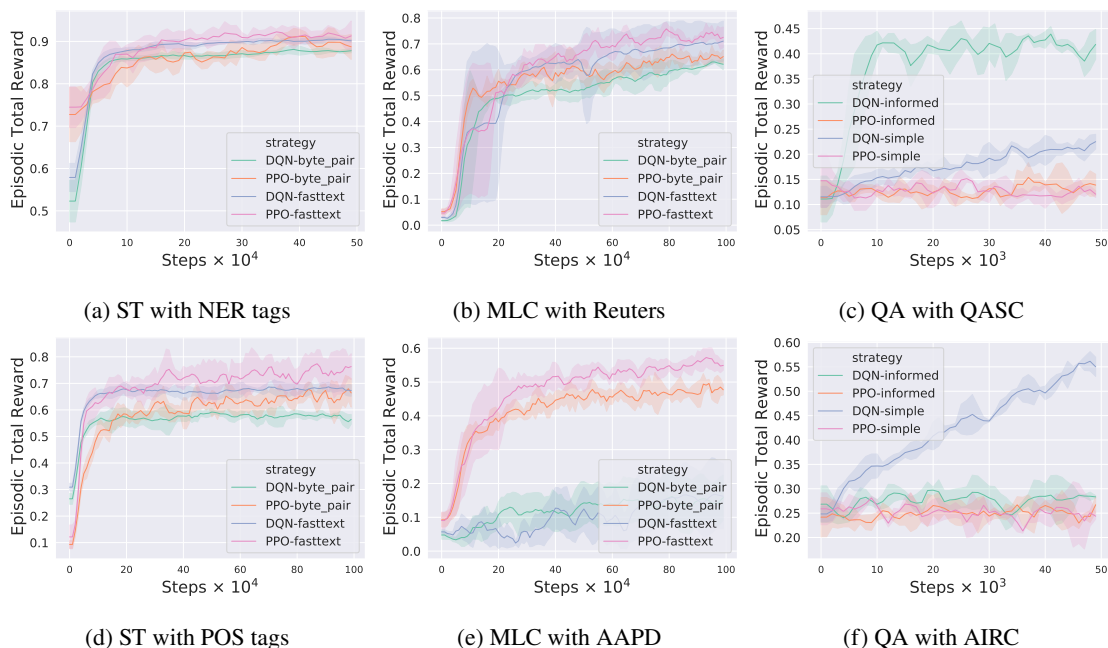


Figure 8.2: **Learning Curves:** Averaged learning curves of PPO and DQN agents with variations. (a),(d) learning curves for sequence tagging with NER and POS tags. (b),(e) MLC with Reuters and AAPD datasets; (c), (f) QA tasks with QASC and AIRC datasets

are added at once and an *interactive/online* setting in which they are added one at a time. A demo script illustrating this has been provided in Appendix B.1. Human-in-the-loop applications favor the later setting in which samples are first annotated by humans and then added to the environment. Furthermore, RL agents can pre-annotate them, thereby reducing human annotation efforts.

8.3 Demo Experiments

To demonstrate the usage of our toolkit, we run the provided environments by plugging-in with benchmark datasets and train Deep Q Networks (DQN) and Proximal Policy Optimization (PPO) agents using stable-baselines [224]. Each experiment is repeated 5 times to obtain average learning curves. All our experiments are run in a batch setting (train data points are added to the environments at once) and evaluated on test split.

Sequence Tagging We chose CONLL [234] and UDPOS [235] as datasets for ST environments. We vary the observation featurizer with fasttext [139] and bytepair [236] embeddings. For each of these settings, we train DQN and PPO agents. For DQN, we chose a feedforward multi-layer perceptron (MLP) with two hidden layers, each consisting of 100 neurons. A discount factor of 0.99 is used. The experiences are stored in a replay buffer and a batch size of 32 is used for sampling. Additionally, Double-Q learning is used. For PPO, we chose the MLP policy with the same number of hidden layers and neurons. The other settings for PPO such as discount factor, batch size, clip parameter, entropy co-efficient are set to 0.99, 64, 0.2 and 0.0 respectively. Both PPO and DQN are trained with

Model/Task	Sequence Tagging		Multi-Label		Question Answering		
	Embedding	CONLL	UDPOS	Reuters	AAPD	QASC	AIRC
DQN	Fasttext	0.92	0.69	0.72	0.35	0.49	0.35
	Byte-Pair	0.91	0.58	0.65	0.26	-	-
PPO	Fasttext	0.93	0.77	0.76	0.61	0.14	0.24
	Byte-Pair	0.91	0.65	0.70	0.49	-	-

Table 8.1: **Test Performance:** Summary of performance of selected DQN and PPO agents on the test set. For ST and MLC, micro F1-scores are reported, and for QA, accuracy is reported

a learning rate of 0.0005.

Question Answering For the QA environment, we test it on QASC [230] and AIRC [237] datasets. The QA experiments are run with two featurizers: *simple* and *informed* featurizers (see observation featurizers in Appendix B.2 for details). The Q network in DQN and policy network in PPO consists of two hidden layers, each consisting of 64 neurons. Both are trained with a learning rate of 0.0001. Other parameters are set to the same values specified in ST experiments.

Multi-label Classification For the MLC environment, we picked Reuters [238] and AAPD [239] datasets. Like in ST, we vary the observation featurizer with fasttext [139] and bytetrain [236] embeddings. For this task, we chose Q-network and policy network to consist of two hidden layers, each with 200 neurons and they are trained with a learning rate of 0.001. Other settings are the same as specified in ST experiments.

Results and Discussion Figure 8.2 shows learning curves for the described agents and Table 8.1 summarizes their generalization performance. Further, in Appendix B.4, we present the predictions (actions) of trained agents. For each agent and its variation, we first evaluate them on the validation set *dev* and select one model based on the validation performance. The selected models are then evaluated on the hold-out *test* set, and their scores are summarized in the Table 8.1. We report micro F1-scores for ST and MLC, and for QA, accuracy is reported. It is observed that both DQN and PPO agents are able to solve ST and MLC tasks with a very good performance, and PPO performs better than DQN agents in most cases.

However, QA tasks are generally difficult to solve and require more complex representations (e.g., BERT [43]), which would improve the results of QA experiments. From Figure 8.2 and Table 8.1, it is seen that the agents seem to memorize the training questions resulting in poor generalization. Nevertheless, all of these results indicate that RL agents can indeed be trained to perform NLP tasks using our toolkit.

8.4 Conclusion

In this work, we presented and demonstrated the usage of NLP Gym for applying DRL to solving NLP tasks. The initial release of this toolkit contains environments for three standard tasks that are ready to use with default components and datasets. The results presented here act as simple baselines to promote the research and benchmarking of RL in NLP settings. We believe that NLP Gym becomes a standard toolkit for testing agents for learning language and understanding. Moreover, in the next Chapter 9, we extend this framework to other NLP tasks such as text summarization, generation and machine translation leveraging the fine-tuning capabilities of large language models.

RL4LMs - Building Blocks, Baselines and Benchmark for NLG using RL

In this part of the thesis, our main focus is to address the limitations associated with structured prediction (SP) tasks. We have discussed that these tasks often face challenges such as data mismatch and metric mismatch. To overcome these challenges, one potential solution is to treat SP as a sequential decision-making problem. This approach is intuitive because both SP and reinforcement learning (RL) share similar characteristics. They both involve generating structured outputs that are essential for the task at hand. Additionally, both SP and RL require step-by-step output generation, where predictions made at each step serve as inputs for subsequent steps.

In the previous chapter, we introduced the toolkit called *NLPGym*, which utilizes RL environments to frame typical NLP tasks and allows direct training to optimize application-specific metrics. However, it's important to acknowledge that this library has its own limitations. For example, the featurizers provided by the language models are simple word embeddings that remain unchanged throughout the training process, with only the final task-specific layers being adapted. Furthermore, modern NLP systems now primarily rely on large language models (LLMs) that can solve most NLP tasks with little or no fine-tuning.

These powerful LLMs are typically trained using a supervised learning objective that predicts the next word given a specific context. With the use of massive amounts of data and billions of parameters, LLMs can effectively handle most NLP tasks with proper prompting. However, despite being fluent text generators, the generated text may sometimes contain biases, toxicity, and harmful outputs. To mitigate this, LLMs undergo *Reinforcement Learning from Human Feedback* (RLHF) [46, 240] to align their outputs with human preferences. However, there is currently a lack of well-benchmarked open-source toolkits available, making systems like ChatGPT and GPT-4 non-reproducible and inaccessible.

In addition to the accessibility problem, there are several challenges associated with RLHF. Firstly, training LLMs involves dealing with large action spaces. Secondly, LLMs may exploit flaws in reward functions and prioritize generating high-reward sentences at the cost of fluency. These factors contribute to the perception that RL is a challenging paradigm for fine-tuning LLMs.

To address the accessibility and stability issues related to RLHF, we first introduce a framework *RL4LMs*. This framework enables the training of decoder and encoder-decoder LLMs using on-policy algorithms such as PPO and A2C. Additionally, we present a benchmark (GRUE) consisting of seven

generative NLP tasks, including abstractive summarization, text continuation, generative common-sense reasoning, data-to-text, machine translation, question answering, and dialog generation. Within this benchmark, we explore a wide range of reward functions and highlight crucial implementation details for training LLMs. Furthermore, we propose a variant of PPO namely Natural Language Policy Optimization (NLPO) that remains stable even when dealing with large action spaces.

The chapter is based on the publication by [90],

- R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi and Y. Choi, “Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”, *Proceedings of International Conference on Learning Representations, 2023*, URL: <https://openreview.net/forum?id=8aHzds2uUyB>

The initial concept of a framework for fine-tuning LLMs using RL was proposed by Prithviraj Ammanabrolu, while Rajkumar Ramamurthy took on the role of lead developer and implemented the core components of the library, including the PPO and A2C algorithms. Prithviraj built the NLPO algorithm on top of this framework. The experimental design and task setups were carried out by Rajkumar, Prithviraj and Kiant Brantley. Jack Hessel provided support with human evaluations and the training of reward models based on human preferences. The resulting paper represents a significant collaborative effort by the main authors and has undergone extensive revisions by all authors.

The remainder of this chapter is organized as follows: In the next section, we provide a brief motivation for utilizing RL in training LLMs, followed by a concise review of the existing literature. Then, in the subsequent section, we introduce *RL4LMs*, discussing text generation environments, reward functions, evaluation metrics, and on-policy algorithms. The NLPO approach is presented in the following section. Finally, we present the GRUE benchmark in the last section 9.4, which focuses on seven generative NLP tasks. We provide details on the experimental setup, evaluation results, and delve into various experiments related to reward hacking, preference reward learning, practical considerations, and data budget.

9.1 Motivation

The primary objective of natural language generation (NLG) systems is to engage with and assist humans in various applications, including summarization, information extraction, question answering, and interactive dialogue assistance. Although these applications inherently involve interaction, most systems rely on training large language models without direct human preference signals. Instead, they use supervised targets that only serve as a rough approximation of human preferences. Moreover, supervised methods often require numerous expert demonstrations and are susceptible to exposure bias.

One approach to incorporating user feedback is through human-in-the-loop training, where users provide feedback for each sample while the model is being trained. However, this level of dense supervision can be inefficient and challenging. Another alternative lies in automated metrics, which offer a promising middle ground between fully supervised and fully interactive training paradigms. Metrics such as BERTScore [241], BLEURT [242], and others have shown improved correlation with human judgments compared to earlier lexical overlap metrics like BLEU [91] and METEOR [243].

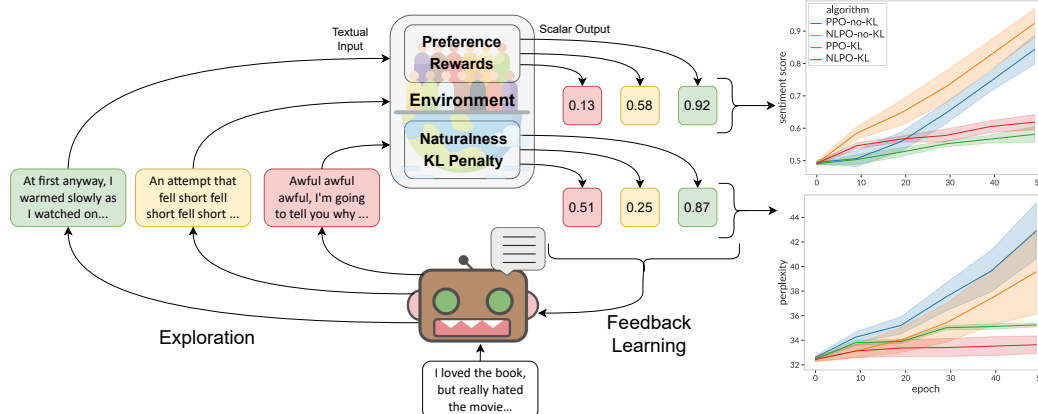


Figure 9.1: **Natural Language Policy Optimization (NLPO)** in the case of sentiment-guided continuation. Here, the LM (i.e., the policy) needs to produce a positive sentiment continuation given a review prompt (we cover other models of human preference in Sec. 9.2.2). Two objectives are balanced: 1) an automated proxy of human preference that serves as a reward (here: a sentiment classifier); and 2) “naturalness” as measured by a KL divergence from an LM not trained with explicit human feedback. The plots show validation learning curves comparing our NLPO to the popular policy gradient method PPO. (Top plot:) RL methods can easily achieve high reward if the KL penalty is removed, (Bottom:) but at the cost of higher perplexity. NLPO+KL, our proposed approach, succeeds in balancing reward and naturalness more effectively than prior work.

Furthermore, these metrics are more cost-effective to obtain than human preferences. Nevertheless, these metrics provide scores for complete generations and do not offer per-token level scores.

Reinforcement Learning (RL) provides a solution by framing natural language generation (NLG) as a sequential decision-making problem and optimizing non-differentiable scalar objectives. However, Goodhart’s Law [244] warns us that when a measure becomes the target, it ceases to be a good measure. This suggests that imperfect metrics utilizing neural networks can inadvertently generate nonsensical samples that achieve high-quality scores, while the actual task remains unsolved. This phenomenon is known as *reward hacking*.

Recent research has demonstrated promising results in aligning language models with human preferences through RL, incorporating fluency considerations in preference-based rewards [46, 240]. However, the progress and reproducibility of this line of research are hindered by the lack of open-source libraries and algorithmic implementations, which leads to the misconception that RL is a challenging paradigm for natural language processing (NLP) [87, 88].

Several under documented pitfalls contribute to this perception. Firstly, training large language models involves dealing with large action spaces (vocabulary), coupled with the high variance in automated metrics, leading to instabilities. Secondly, reward hacking occurs when models exploit flaws in the reward function and learn to generate high-reward sentences for incorrect reasons, such as producing task relevant outputs but not fluent text or vice-versa. Finally, critical implementation details that greatly impact the performance of models, are often undocumented.

To facilitate easeness in training large language models using RL and research in building RL algorithms to better align LMs, we release a library, a comprehensive benchmark and a novel RL algorithm that is well-suited for natural language tasks.

- First, we release the *RL4LMs* library, which enables training of generative transformers models

(eg., GPT-2 or T5) using a variety of existing RL algorithms like PPO/A2C/etc.

- As the second contribution, we train these transformer based LMs using RL4LMs to the new **GRUE (General Reinforced-language Understanding Evaluation)** benchmark consisting of a collection of non-exhaustive NLP tasks. These are: (1) abstractive summarization, (2) text continuation, (3) data-to-text, (4) generative commonsense reasoning, (5) machine translation, and (6) question-answering (7) dialog generation. See Table 9.1 for details. In contrast to existing NLP benchmarks, we pair each task with multiple applicable reward function(s). **GRUE** is designed to challenge RL algorithms to optimize these reward functions while being fluent language generators.
- Next, we introduce a novel on-policy RL algorithm called **NLPO (Natural Language Policy Optimization)**, that dynamically learns task-specific constraints over the distribution of language at a token level.
- Finally, we highlight some of the implementation details that are crucial for training of LMs using RL and identify suitable NLP tasks for the RL paradigm

Experiments on **GRUE** and human evaluations show that NLPO better balances learning preference rewards while maintaining language fluency compared to alternatives, including PPO (See Figure 9.1). We find that using RL to learn from scalar reward feedback can be more: (1) data efficient than using additional expert demonstrations via supervised learning (though a combination of both is best)—a learned reward function enables greater performance when used as a signal for an RL method than a supervised method trained with 5 times more data, and (2) parameter efficient—enabling a 220 million parameter model trained with a combination of supervision and NLPO to outperform a 3 billion supervised model. We hope that the benchmarks, baselines, and building blocks we release serve to drive forward research in aligning LMs to human preferences.

9.1.1 Related Work

Natural Language Generation (NLG) research on sequential decision-making can be broadly categorized into two main paradigms: imitation learning (IL) and reinforcement learning (RL).

Imitation learning for NLP: Numerous IL algorithms have been developed to mimic expert demonstrations. These include Schedule Sampling (SS) [245], Parallel SS [246], SS for Transformers [247], Differential SS [248], LOLS [249, 250], and SEARNN [251], which draw inspiration from DAGGER [252] and SEARN [28]. However, these algorithms are known to suffer from the cliff MDP problem [253–255].

RL for NLP: RL techniques have been successfully applied to improve various NLP models, including machine translation [256, 257], summarization [86, 258], dialogue [259–261], image captioning [262], question generation [263], and text-games [221, 264], among others [265, 266]. For example, [267] adapt reward-conditioned transformers [268] for various language generation tasks. RL has also been explored to align language models with human preferences [85, 86, 240, 269]. Notably, [46] fine-tuned a 175B parameter language model with Proximal Policy Optimization (PPO) [270] to align with models of human preference, although their non-public dataset restricts comparisons.

Despite RL’s success in some of the aforementioned use cases, it has also been criticized for its instability compared to supervised language model training [87]. As a result, there is little consensus

on whether RL, particularly through models of human preference, is a viable approach for training language models compared to alternatives like collecting additional supervised data.

RL for Large Action Spaces: MIXER [265] combined ideas from scheduled sampling and REINFORCE [271]. Additionally, [272] proposed an actor-critic algorithm to address the variance and large action space problems associated with using REINFORCE for language generation. Subsequent works such as KG-A2C [273], TruLL [274], AE-DQN [275], and GALAD [276] have tackled similar challenges by attempting to reduce or eliminate the action space during exploration.

9.2 RL4LMs - A library to train language models using RL

First, we present *RL4LMs*, an open-source library consisting of building blocks for fine-tuning of language models for natural language text generation. The library is built on HuggingFace transformers library [277] and stable-baselines-3 [278], adapting their interfaces to work together which allows interchanging different language models and on-policy algorithms. *RL4LMs* allows training of any decoder only (eg. GPT2) or encoder-decoder transformer models (eg. T5) from HuggingFace with any on-policy RL algorithm such as PPO [270], TRPO [162], A2C [146], and our own NLPO (See section 9.3) from stable-baselines3.

In addition, we release implementations of popular on-policy RL algorithms that are tailored for transformer-based LM fine-tuning together with our own novel algorithm. In the initial release, we provide 7 NLG tasks, 16+ evaluation metrics/rewards and 4 RL algorithms. The library is modular and customizable which enables users to easily plug-in their customized versions of tasks, reward functions, metrics and algorithms.

9.2.1 Environments: Generation as a Token-level MDP

Each environment corresponds to a natural language processing (NLP) task. We are provided with a supervised dataset, denoted as $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$, containing N examples. Here, $\mathbf{x} \in \mathcal{X}$ represents a language input, and $\mathbf{y} \in \mathcal{Y}$ represents the target string that we aim to predict. We can consider the task of generating output as a Markov Decision Process (MDP) denoted as $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, P, \gamma, T \rangle$, utilizing a finite vocabulary \mathcal{V} .

Each episode in the MDP starts by randomly selecting a datapoint (\mathbf{x}, \mathbf{y}) from our dataset and terminates either when the current time step t surpasses the predefined horizon T or when an end-of-sentence (EOS) token is generated. The input $\mathbf{x} = (x_0, \dots, x_m)$ represents a task-specific prompt and serves as our initial state $s_0 = (x_0, \dots, x_m)$, where $s_0 \in \mathcal{S}$ and \mathcal{S} denotes the state space, with $x_m \in \mathcal{V}$.

An action within the environment, denoted as $a_t \in \mathcal{A}$, corresponds to selecting a token from our vocabulary \mathcal{V} . The transition function $P : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ deterministically appends an action a_t to the end of the state $s_{t-1} = (x_0, \dots, x_m, a_0, \dots, a_{t-1})$. This process continues until reaching the end of the horizon $t \leq T$, resulting in a state $s_T = (x_0, \dots, x_m, a_0, \dots, a_T)$. At the end of an episode, a reward is emitted based on the (s_T, \mathbf{y}) pair, denoted as $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{Y} \rightarrow \mathbb{R}^1$. This reward could be determined using automated metrics such as PARENT [279].

The RL4LMs framework provides an OpenAI Gym [80]-style API for an RL environment that simulates this MDP formulation based on language models. By abstracting the specific details of the MDP environment structure, it enables the swift addition of new tasks while maintaining compatibility

with all implemented algorithms, without necessitating any dataset-specific modifications.

9.2.2 Reward Functions and Evaluation Metrics

Due to its generic interface for per-token or per-sequence generation rewards, it becomes possible to easily apply a diverse range of reinforcement learning (RL) algorithms to a similarly wide array of textual metrics-as-rewards. We provide interfaces for several types of metrics, including:

1. **n-gram overlap metrics** such as ROUGE [73], BLEU [91], SacreBLEU [280], and METEOR [243];
2. **model-based semantic metrics** such as BertScore [241] and BLEURT [242], which typically exhibit stronger correlations with human judgment;
3. **task-specific metrics** such as CIDER [281], SPICE [282] (for captioning/commonsense generation), PARENT [279] (for data-to-text), and SummaCZS [283] (for factuality of summarization);
4. **diversity/fluency/naturalness metrics** including perplexity, Mean Segmented Type Token Ratio (MSSTR) [284], Shannon entropy over unigrams and bigrams [285], the ratio of distinct n-grams over the total number of n-grams (Distinct-1, Distinct-2), and the count of n-grams that appear only once in the entire generated text [286];
5. **task-specific, model-based human preference metrics** such as classifiers trained on human preference data collected following the methodology of [46].

9.2.3 On-policy Actor-critic Algorithms

RL4LMs supports fine-tuning and training LMs from scratch via on-policy actor-critic algorithms on language environments. Formally, this class of algorithms allows us to train a parameterized control policy defined as $\pi_\theta : \mathcal{S} \rightarrow \mathcal{A}$, a function that attempts to select an action in a given state so as to maximize long term discounted rewards over a trajectory $\mathbb{E}_\pi [\sum_{t=0}^T \gamma^t \mathcal{R}(s_t, a_t)]$.

Our benchmark experiments focus on fine-tuning a pre-trained LM denoted as π_0 from which we initiate our agent’s policy $\pi_\theta = \pi_0$. Similarly, the value network V_ϕ used to estimate the value function is also initialized from π_0 except for the final layer which is randomly initialized to output a single scalar value. As with other deep RL actor-critic algorithms, we define our value and Q-value functions as $V_t^\pi = \mathbb{E}_{a_t \sim \pi} [\sum_{\tau=t}^T \gamma R(s_\tau, a_\tau, \mathbf{y})]$, $Q_t^\pi(s_t, a_t) = R(s_t, a_t, \mathbf{y}) + \gamma \mathbb{E}_{s_{t+1} \sim P} [V_{t+1}^\pi(s_{t+1})]$ leading to a definition of our advantage function as $A_t^\pi(s, a) = Q_t^\pi(s, a) - V_t^\pi$. To increase training stability, advantage is approximated using Generalized Advantage Estimation [287].

Given an input-output pair (\mathbf{x}, \mathbf{y}) and generation predictions from our agent; because the environment rewards are sequence-level and sparse, following [240] we regularize the reward function using a token-level KL penalty for all on-policy algorithms, to prevent the model from deviating too far from the initialized LM π_0 . Formally, the regularized reward function is:

$$\hat{R}(s_t, a_t, \mathbf{y}) = R(s_t, a_t, \mathbf{y}) - \beta \text{KL}(\pi_\theta(a_t|s_t) || \pi_0(a_t|s_t)) \quad (9.1)$$

where \hat{R} is the regularized KL reward, \mathbf{y} is gold-truth predictions, $\text{KL}(\pi_\theta(a_t|s_t) || \pi_0(a_t|s_t)) = (\log \pi_0(a_t|s_t) - \log \pi_\theta(a_t|s_t))$ and the KL coefficient β is dynamically adapted [85] during training

where,

$$e_t = \text{clip} \left(\frac{\text{KL}(\pi(a_t|s_t) || \pi_0(a_t|s_t)) - \text{KL}_{\text{target}}}{\text{KL}_{\text{target}}}, -0.2, 0.2 \right) \quad (9.2)$$

$$\beta_{t+1} = \beta_t (1 + K_\beta e_t) \quad (9.3)$$

where $\text{KL}_{\text{target}}$ is user-specified KL divergence between initial model π_0 and current policy π and K_β is rate of update which we generally set to 0.2 in our experiments.

9.3 NLPO: Natural Language Policy Optimization

The magnitude of language generation action spaces far surpasses the design scope of most reinforcement learning algorithms that handle discrete action spaces [265, 288]. For example, GPT-2/3 and T5 possess a vocabulary size of 50K and 32K respectively. We hypothesize that the extensive size of the action space constitutes a fundamental factor contributing to the instability encountered when training language models using current RL techniques.

To tackle this issue, we propose NLPO (Natural Language Policy Optimization), which draws inspiration from previous work on action elimination and invalid-action masking [273, 275, 289]. NLPO extends the PPO (Proximal Policy Optimization) algorithm with parameterized masking, allowing it to learn how to mask out less relevant tokens within the context during training. NLPO achieves this through the utilization of top- p sampling, a technique that limits the token selection to the smallest possible set whose cumulative probability exceeds the probability threshold p [290].

NLPO incorporates a masking policy, denoted as π_ψ , which closely mirrors the current policy (π_θ), but undergoes updates only after every μ steps. This policy maintains a parameterized-invalid-mask obtained by selecting the top- p tokens from the vocabulary, followed by applying an invalid-mask to the remaining tokens. The invalid-mask sets the probabilities of these tokens to zero when sampling actions from π_θ during training. It is worth noting that alternative sampling techniques such as top- k or beam search (or even domain expert-defined rules) could be used to train π_ψ , but our empirical findings indicate that top- p sampling is the most effective approach in practice. This periodic update of the policy π_ψ is inspired by off-policy Q-learning algorithms [291], enabling π_θ to strike a balance between maximizing task-relevant information and mitigating potential issues related to reward manipulation derived from π_0 through the application of a KL penalty.

NLPO employs a masking policy, denoted as π_ψ , to learn the process of masking irrelevant language. This masking policy is a replica of the current policy (π_θ), but it undergoes updates only at regular intervals of every μ steps. Given $Z(\pi_\theta)$, which represents the normalization value obtained by summing the probabilities of all actions a in the action space \mathcal{A} for a given state $s \in \mathcal{S}$, let $\mathcal{V}_{\pi_\theta}^P$ be the parameterized top- p vocabulary subset. This subset comprises the p highest probability vocabulary tokens with respect to the policy π_θ . More formally, let Z^P represent the normalization value for the parameterized top- p vocabulary, which can be defined as the subset of tokens that maximizes $Z^P(\pi_\theta) = \sum_{a \in \mathcal{V}_{\pi_\theta}^k} \pi_\theta(a|s)$. Consequently, optimizing a policy based on the parameterized top- p vocabulary can be defined as follows:

$$\pi_\psi(\cdot|s, \pi_\theta) = \begin{cases} \pi_\theta(\cdot|s)/Z^P(\pi_\theta) & \text{if } a \in \mathcal{V}_{\pi_\theta}^P \text{ and } Z(\pi_\theta) \\ 0 & \text{otherwise.} \end{cases} \quad (9.4)$$

Algorithm 4 NLPO - Natural Language Policy Optimization

Input: Dataset $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^N$ of size N
Input: initial policy parameters π_{θ_0}
Input: initial LM π_0
Input: initial value function parameters V_{ϕ_0}
Input: initialize parameterized masked policy $\pi_{\psi_0}(\cdot|\cdot, \pi_{\theta_0})$ with parameterized top- p policy π_{θ_0}
Input: policy update frequency μ
repeat
 Sample mini-batch $\mathcal{D}_m = \{(\mathbf{x}^m, \mathbf{y}^m)\}_{m=1}^M$ from \mathcal{D}
 Collect trajectories $\mathcal{T}_m = \{\tau_i\}$ by running policy π_{ψ_n} in for batch \mathcal{D}_m in env. ▷ Eq.9.4
 Compute Preference and KL penalty rewards \hat{R}_t ▷ Eq. 9.1
 Compute the advantage estimate \hat{A}_t ▷ Sec. 9.2.3
 Update the policy by maximizing the PPO-Clip objective:

$$\pi_{\theta_{m+1}} = \operatorname{argmax}_{\theta} \frac{1}{|\mathcal{D}_m|T} \sum_{\tau \in \mathcal{D}_m} \sum_{\tau=0}^T \min \left(r_t(\theta) A^{\pi_{\theta_m}}, \operatorname{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) A^{\pi_{\theta_m}} \right)$$

where $r_t(\theta) = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_m}(a_t|s_t)}$.

Update the value function:

$$V_{\phi_{m+1}} = \operatorname{argmin}_{\phi} \frac{1}{|\mathcal{D}_m|T} \sum_{\tau \in \mathcal{D}_m} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2$$

Update the parameterized masked policy every μ iterations:

$$\pi_{\psi_{n+1}}(\cdot|\cdot, \pi_{\theta_{m+1}})$$

until convergence and **return** π_{θ}

Finally, the pseudocode of the proposed is presented in Algorithm 4 (green portions highlight the differences with PPO).

9.4 General Reinforced-language Understanding Eval - GRUE Benchmark

GRUE consists of a collection of 7 generative NLP tasks. To address the issue of reward hacking based on a single metric, each task is evaluated at test time using a task-specific combination of metrics, which are outlined in detail in Table 9.1. These metrics are categorized into two groups: **Task preference metrics** assess how well the models generate outputs that meet the specific requirements of each task. For example, in the case of CommonGen, the metrics assess whether the generated outputs covers all the required concepts, while for IMDB, they measure the positive sentiment of the

9.4 General Reinforced-language Understanding Eval - GRUE Benchmark

Dataset	Task	Input	Output	Task Preference Metrics(s)	Naturalness Metrics(s)
IMDB [292]	Text Continuation	Partial Movie Review	A positive completion of the movie review.	Learned Sentiment Classifier	Perplexity (GPT-2)
CommonGEN [293]	Generative Commonsense	Concept Set	A sentence coherently using all input concepts.	CIDER; ROUGE-2,L; BLEU-3,4; METEOR; Coverage	SPICE
CNN Daily Mail [219]	Summarization	News Article	Summarized article.	SummaCZS; ROUGE-1, 2, L, LSum; METEOR; BLEU	BertScore
ToTTo [294]	Data to Text	Highlighted Wiki Table	Factually accurate text describing the information.	SacreBLEU; PARENT	BLEURT
WMT-16 (en-de) [295]	Machine Translation	Text (English)	Translated text (German).	TER; cHRF; ROUGE-1, 2, L, LSum, METEOR; SacreBLEU, BLEU	BertScore
NarrativeQA [296]	Question Answering	Question Context (a Story)	Abstractive answer to the question.	ROUGE-1, 2, L, LSum, LMax; METEOR; BLEU; SacreBLEU	BertScore
DailyDialog [297]	Chitchat Dialogue	Dialogue History	A conversational response	METEOR; Learned Intent Classifier	BertScore

Table 9.1: **GRUE Benchmark using RL4LMs** showing the various tasks, input and output types, and the metrics used. We note that we test RL algorithms on these tasks for a wider range of possible rewards than just the task specific ones shown here. Unless specified, datasets are in English.

generated completions. On the other hand, **Naturalness metrics** focus on aspects such as fluency and readability, providing insights beyond semantics. During training, there are no specific constraints imposed: models are allowed to utilize supervised data, compute metrics on intermediate generations, and so on. The train/validation/test splits adhere to the original works, with the models being trained on the designated train splits. All reported results are averaged across multiple seeds, and specific counts can be found in the Appendix C.

Experimental Setup: We utilize the *RL4LMs* to conduct extensive evaluations of various algorithms on the GRUE benchmark. Specifically, we compare three algorithms for direct fine-tuning: Supervised, PPO (Proximal Policy Optimization), and NLPO (Natural Language Policy Optimization). In addition, we also consider a hybrid approach that combines supervised learning with RL methods. This involves applying PPO and NLPO to checkpoints that have undergone supervised fine-tuning, which we refer to as Supervised+PPO and Supervised+NLPO, respectively. As an additional baseline, we conduct zero-shot evaluations where we design prompts that aim to elicit task-specific generations without any training data or parameter updates.

To isolate the impact of the training method, we select a single pre-trained LM backbone for each task. For IMDB text continuation, we utilize GPT-2 with 117 million parameters, while for the remaining tasks, we employ T5-base with 220 million parameters.

For our RL models (PPO, NLPO, Supervised+PPO, Supervised+NLPO), we conduct separate experiments to thoroughly investigate how reward-hacking may interact with the GRUE evaluation. We optimize multiple task rewards for each task independently, running individual experiments for each metric. For example, in the case of Commongen, which has six task rewards (CIDER, ROUGE-2, ROUGE-L, BLEU-3, BLEU-4, METEOR), we perform six different experiments, each optimizing a specific metric independently. We report all the metrics presented in Table 9.1, regardless of which

individual metric was being optimized for in each experiment.

Human Participant Study: We have collected human judgments for five of the tasks included in the GRUE benchmark. Our objective in doing so is twofold: 1) to validate the correlation between the automated metrics we have selected for GRUE and human judgments, specifically in terms of the relative ranking of models. 2) to provide additional empirical comparisons, particularly between NLPO and PPO, as well as conducting ablations to investigate the effects of the KL naturalness penalty and other factors. The tasks we specifically focus on for human judgment collection are IMDB, Commongen, ToTTo, DailyDialog, and CNN Daily Mail. For each individual sample in a task, we requested the input of three unique human raters to provide Likert judgments on two aspects: 1) quality, i.e., for the specific task, how correct/appropriate is the generation, given the context, and 2) fluency, i.e., how well-written is the generation. We utilized Amazon Mechanical Turk as the platform for collecting these judgments and ensured that crowdworkers were paid a minimum of 15 per hour for their participation. Additional details, including qualification information, interface screenshots, instructions, and more, can be found in the Appendix C.

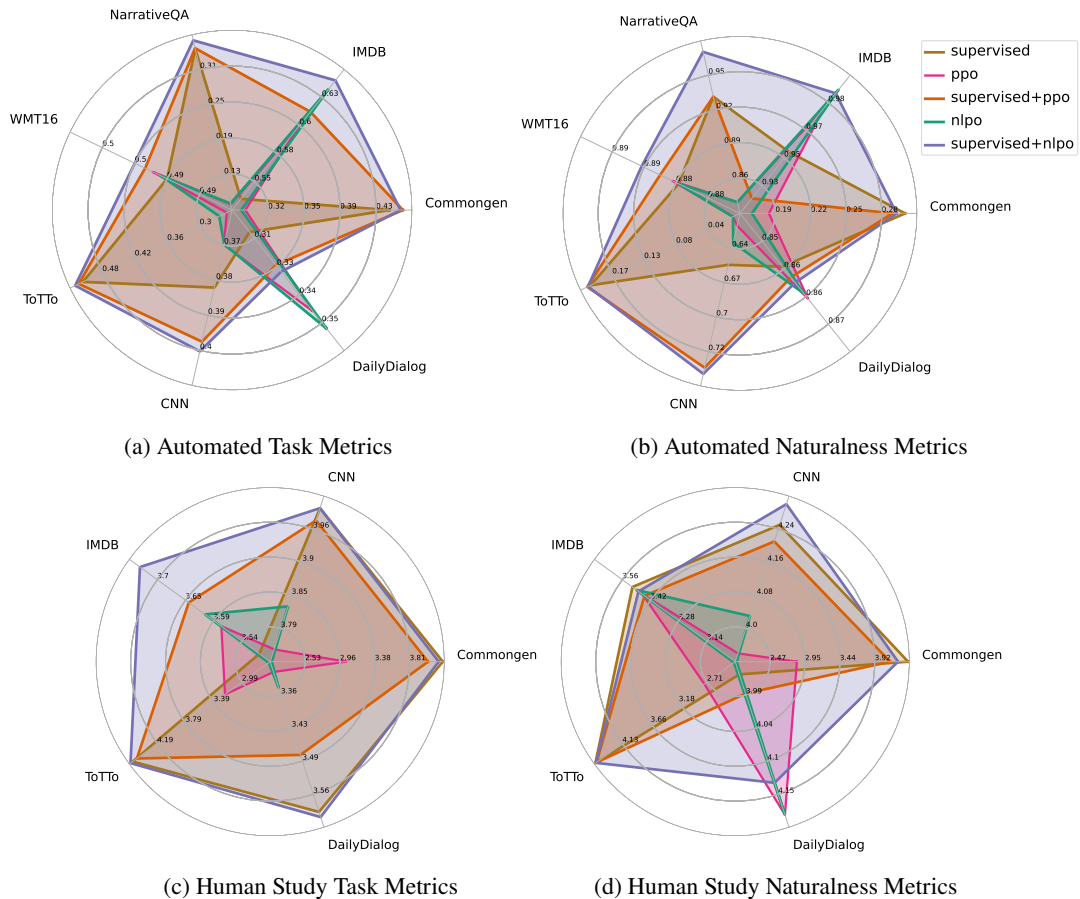


Figure 9.2: **GRUE results:** Summarized results via automated metrics across all 7 GRUE tasks for each of the 5 algorithms we consider, and human participant studies for the 5 tasks suitable for human studies. Test results are averaged over all the respective metrics seen in Table 9.1.

Questions	Tasks					
	IMDB	CommonGen	CNN/DM	ToTTo	IWSLT-17	NarQA
Needs Warm Start	X	✓	X	✓	X	✓
Easily reward hackable?	✓	✓	X	X	X	X
RL > Sup (auto)?	✓	X	X	X	X	X
RL > Sup (human)?	✓	X	X	X	-	-
Sup+RL > Sup (auto)?	✓	✓	✓	✓	✓	✓
Sup+RL > Sup (human)?	✓	X	✓	✓	-	-
Sup+NLPO > Sup+PPO (auto)?	✓	✓	✓	✓	✓	✓
Sup+NLPO > Sup+PPO (human)?	✓	✓	✓	✓	-	-

Table 9.2: Key questions answered using GRUE + RL4LMs:

This table summarizes the results found in the ablations and Fig. and provides an overview of the questions we ask in Section 9.4: which tasks require warm starts or are easily reward hackable; when to use RL over Supervised, when to use both; and when to use NLPO over PPO. All conclusions drawn are the result of statistical analysis as discussed in the experimental setup.

	Ablation	Sentiment	Perplexity
	Zero Shot	0.489	32.171
	Supervised	0.539	35.472
	PPO	0.605	33.497
	NLPO	0.637	32.667
Warm Starting (Sec. 9.4.1)			
	PPO+Supervised	0.617	34.078
	NLPO+Supervised	0.645	33.191
Data Budget (Reward trained on 10% of data, Sec. 9.4.3)			
	PPO	0.598	35.929
	NLPO	0.599	33.536
Removing NLPO Top- p Constraints (Sec. 9.4.2)			
$(p = 1$ is equivalent to PPO, $p = 0.9$ is NLPO)			
	NLPO $p = 0.1$	0.579	32.451
	NLPO $p = 0.5$	0.588	32.447
Removing KL Constraints (Sec. 9.4.2)			
	PPO-no-KL	0.859	37.553
	NLPO-no-KL	0.853	36.812
Discount Ablations ($\gamma = 1$) (Sec. 9.4.4)			
	PPO	0.651	41.035
	NLPO	0.624	43.72

Table 9.3: IMDB Ablation Results

9.4.1 Results on GRUE

Figures 9.2(a), 9.2(b) present the results on GRUE, split into automated task metrics and naturalness metrics, and Tables 9.2, 9.3 highlight key results via ablation studies. Full results are available in Appendix C.1

RL vs Supervised: For text continuation and summarization, with non-trivial zero-shot performance, RL tends to perform better than supervised training, but for tasks like CommonGen and ToTTo, which have very low zero-shot performance, supervised training performs best—with both approaches outperforming zero-shot.

However, **using RL+Supervised learning in conjunction works best;** NLPO+supervised and PPO+supervised usually always outperforms NLPO/PPO (or supervised in isolation) across both task metrics and naturalness metrics. Supervised warm-starting is particularly effective for CommonGen and ToTTo, which our results suggest are more prone to reward hacking. The one exception to this trend is DailyDialog where the RL models outperform warm-started Supervised+RL models likely due to the low performance of the Supervised models.

We note that Supervised+NLPO using a T5-base (220m parameter) LM currently outperforms all the models on the ToTTo leaderboard, many of which have ≥ 3 b parameter supervised models—suggesting that RL is parameter efficient as well. In these cases, it is critical that the initial policy already contain (some) signal for the task due to it being used as a KL constraint and masking constraint in NLPO. If the mask contains no initial priors about task specific language, it will be eliminating the wrong actions—a better initial policy leads to better RL performance downstream.

Human agreement with automated metrics: We incorporate additional statistical analyses to mitigate the potential noise in human judgments. These analyses include assessing inter-annotator agreement using Krippendorff’s alpha score and conducting a one-way ANOVA followed by a post-hoc Tukey HSD test. These statistical measures allow us to determine the significance of differences in average scores among different models.

Our findings reveal that the trends observed in human evaluations generally align with those observed in automated metrics for both task-related and naturalness-related assessments (refer to Figures 9.2(c)

and 9.2(d) which summarize Appendix Tables C.7, C.13, C.20, C.26, and C.37). These trends indicate that the performance ranking is as follows: Supervised+NLPO is superior to Supervised, and Supervised+PPO is better than or equal to Supervised and NLPO, while PPO outperforms Zero-shot. However, there is an exception where Supervised outperforms Supervised+PPO in two out of five tasks, despite automated metrics suggesting the opposite.

Based on these observations, we draw two conclusions:

1. When the generated text surpasses a certain naturalness threshold, the automated metrics generally align with human judgments.
2. This alignment is mostly consistent but not always perfect, as evident in the performance comparison between Supervised and Supervised+PPO. It suggests the possibility of reward hacking behaviors that go unnoticed by automated metrics but are captured by human preference feedback.

9.4.2 Preference Reward Learning, Selection, and Hacking

While the GRUE benchmark’s metric for each task is an average over several measures, the RL models we trained optimized only a single metric independently. Thus, we can empirically investigate which metric for which GRUE produces the best results. We observe that many possible single metric rewards provide task performance gains over supervised methods (results shown in Fig. C.1(a), 9.2(c) are averaged across these reward functions) with the condition that the text is also coherent and natural.

Which constraints are most effective in preventing reward hacking? To prevent reward hacking, the reward function in Equation 9.1 incorporates a KL constraint that balances a task-specific reward. This constraint penalizes models for deviating too far from a base LM in their quest for high reward. As evidenced by Table 9.3 and Table C.2, removing KL constraints altogether allows models to exploit the system and achieve significantly higher sentiment scores at the expense of naturalness, leading to an increase in perplexity.

However, the choice of the base regularizing LM greatly influences the effectiveness of this approach. When the initial policy, i.e., the raw pretrained model, performs poorly on the task, the KL penalty tends to push the policy towards generating nonsensical outputs. For instance, in the case of Comngen and ToTTo, the trained policy learns to repetitively reproduce parts of the input (as observed in Tables C.1.4 and C.1.6). This undesired behavior can be mitigated by using the supervised model as the base regularizing LM, as the reward encourages the policy to strike a balance between the task-specific reward and a more sensible regularization term. Thus, deriving KL penalties from warm-started initial policies is crucial for achieving optimal performance on such tasks.

PPO vs. NLPO. Figure 9.2 illustrates that NLPO generally surpasses PPO and supervised methods, particularly when applied after supervised training. We hypothesize that the primary reason behind NLPO’s enhanced performance and stability lies in the masking policy, which imposes an additional constraint on the current policy. Unlike the KL penalty, which is based on the initial untuned policy, this constraint derives from the policy used μ iterations ago and likely incorporates more task-relevant knowledge acquired during RL training.

The results presented in Table 9.3 (and Appendix Table C.5) demonstrate how performance initially increases and then decreases as the p value in top- p sampling is raised for the masking policy. This relaxation of the constraint leads to fewer tokens being eliminated at each step, implying that there exists a balance in the level of constraint imposed during RL training.

Human Preference Reward Learning. Up until now, our experiments have primarily focused on optimizing evaluation metrics that correlate with human judgments, such as METEOR. In addition, we conducted tests to assess the effectiveness of learning preferences from direct human feedback. To conduct these tests, we chose CommonGen, a benchmark dataset that is well-suited for highlighting differences resulting from human preferences. Initially, we randomly selected prompts from the training set of CommonGen and generated a single completion from both the Supervised and Supervised+NLPO models. These prompt-completion pairs were then presented to three different crowdworkers, who were asked to indicate their preferred option based on commonsense and fluency considerations for a total of 417 unique pairs (Krippendorf $\alpha = .28$). We used this data to train a reward model, T5-11B [298], using a balanced binary classification task.

The goal was to predict which completion was preferred by the majority of the three annotators, given the prompt and completion. The resulting model achieved a test ROC AUC of 69.5, indicating that it successfully captures average human preferences. Further details about this process can be found in Appendix C.1.4. As a baseline, we trained the Supervised+RL model using only the METEOR reward and compared it to a reward function that employed the fine-tuned T5-11B model. Subsequently, we repeated the same pairwise preference collection procedure, this time sampling from the CommonGen test set, and sought human participants' opinions to compare the generations produced by a preference-optimized RL policy with the previously superior Supervised+NLPO policy. When comparing the METEOR-only model to the preference model, the human feedback model was preferred in 682 cases, while the METEOR-only model was preferred in 587 cases ($p < 0.01$). This suggests that the pipeline of collecting preferences, training a reward model, and further fine-tuning the policy leads to improved alignment with human preferences.

9.4.3 Data Budget: Improve your Reward or Gather More Demonstration?

When given a fixed data collection budget, the question arises: is it more efficient to gather feedback to enhance a learned reward function or to gather additional expert demonstrations? To explore this, we focus on the IMDB text continuation task as a case study. In this task, a model is provided with a partial movie review as a prompt and is tasked with continuing it as positively as possible, even if the initial prompt was negative. The original dataset consists of movie reviews with sentiment labels indicating positive, negative, or neutral sentiments. We train a DistilBERT classifier [299] on these labels, which generates sentiment scores representing the level of positivity in a given text. These sentiment scores serve as the task reward. The dilemma lies in deciding whether to acquire more sentiment labels (to improve the reward) or gather positive sentiment reviews (to enhance supervised training).

To investigate this, we train a classifier on various amounts of training data and evaluate its performance on a held-out test dataset. As expected, we observe that increasing the amount of training data improves test accuracy, resulting in a higher-quality reward. Subsequently, we utilize these rewards of varying quality during RL training and evaluate the performance using the same metric as the benchmark. As presented in Table 9.3, we find that enhancing the quality of the reward leads to improved performance of the language model. Additionally, we train a supervised model using at least the same number of samples as those used to train each reward classifier. Intriguingly, we discover that a learned reward function enables superior performance when used as a signal for an RL method compared to a supervised method trained with five times more data. This implies that enhancing reward models can be more data-efficient for a given task compared to collecting expert

demonstrations. Moreover, it is worth noting that assigning sentiment labels is likely a simpler task than writing complete demonstrations. The ablation details for this experiment can be found in Appendix Table C.3.

9.4.4 Practical Considerations: Which Implementation Details Matter Most?

Generation as a token-level MDP, not a bandit environment. Recent approaches to fine-tuning Language Models (LMs) using Reinforcement Learning (RL) typically assign rewards for all tokens in the sentence, and then adjust the probabilities of generating the entire text accordingly [46, 240, 267]. This approach can be viewed as a bandit feedback environment, where the action space represents all possible generated sequences for the given task [26]. In our RL formulation, we can simulate this type of environment by setting the discount factor (γ) to 1. However, our experiments on IMDB dataset, presented in Table 9.3 (and Appendix Table C.4), demonstrate that this approach leads to training instability in terms of naturalness when using both PPO and NLPO. Our standard setting is $\gamma = 0.95$ when calculating discounted rewards-to-go in the token-level MDP formulation, which reduces the magnitude of the reward that is applied to tokens selected at the beginning. The sentiment scores are approximately the same between both settings but the naturalness of language in the bandit setting is significantly less —indicating that discounting rewards with $\gamma < 1$ via a token-level MDP formulation is at least sometimes more effective for language generation.

Dropout and Sampling. We have identified two additional implementation details that significantly impact the stability of RL training. The first crucial aspect is dropout, which has been observed to introduce instability in policy gradient methods within continuous control scenarios, as highlighted by [300]. We have observed a similar effect when utilizing dropout during RL training of LLMs, where training loss often diverges when dropout > 0 in training.

The second important factor, particularly influential in the machine translation task, pertains to the choice of sampling methods. The sampling approach employed to select actions (tokens) from the LM during exploration and at inference plays a crucial role. We have found that using consistent sampling methods throughout both exploration and inference is essential for achieving strong performance on test metrics, as it ensures that the model can effectively generalize its training rewards to test scenarios. Otherwise, the model may exhibit high rewards during training but fail to demonstrate desirable performance on test data.

9.5 Conclusion

In this work, we proposed the GRUE benchmark and the *RL4LMs* library that will help progress in aligning language models to human preferences via RL fine-tuning, by providing the community a standard means of comparing methods. Furthermore, we are highly hopeful that as training stability and consistency improve, our methods will unlock a pathway for iterative advancements in language technologies. This pathway involves crucial steps such as deploying models, actively collecting user feedback, and iteratively re-optimizing them.

Conclusion

In this thesis, the primary focus was on *sequential decision and prediction tasks* (SDP), which include structured prediction (SP) and reinforcement learning (RL) tasks. These tasks share two key characteristics. First, in SP, the output is generated incrementally, one element at a time, with each predicted output being fed as input in the subsequent steps. RL follows a similar setup, where actions are decided one at a time, and these actions have consequences on future states. Second, the outputs exhibit interdependencies, and there exists a temporal structure among them. A common requirement for these tasks is a sequential model that can effectively process both sequential inputs and outputs. While RNNs (such as GRUs and LSTMs), CNNs, and transformers are commonly used to model sequences in various applications and have achieved state-of-the-art results, their computational demands limit their applicability in low-resource settings. Consequently, this gives rise to our research question (**RQ1**): Can random context encoders, such as ESNs, suffice to solve these tasks, and what is the performance gap between them and fully trained RNNs?

To address this research question, Chapter 3 focused on a challenging SP task called sequence memorization. In this chapter, we demonstrated that ESNs have the capability to memorize and reproduce arbitrary sequences of data, including text, images, and videos. Remarkably, ESNs, which are simply randomized networks without full training, achieve this impressive feat. We then leveraged this capability of ESNs to propose a novel cryptography system.

Furthermore, in Chapter 4, we explored the application of ESNs in another SP task, namely NER in NLP. While state-of-the-art approaches for NER rely on LSTMs or transformer architectures to obtain contextualized word representations, we demonstrated that the reservoir states in ESNs act as randomized contextualized representations, which are sufficient to solve the task. ESNs for NER offer practical advantages, requiring significantly lower training times compared to RNNs.

Shifting our focus to RL, Chapter 5 investigated a sequential decision-making task. Specifically, we applied ESNs to a partially observable setting, where the inputs from the environment do not fully capture the state of the environment. In this setting as well, ESNs provide a form of randomized context that proves effective in solving control tasks in RL. In summary, our work on ESNs highlights their effectiveness as random context encoders, capable of successfully tackling a wide range of SDP tasks.

Next part of the thesis focused on the primary challenges that exist in RL. In many sequential-decision making tasks, agents are only rewarded in the goal state, necessitating a significant number of interactions to acquire a suitable policy. This gives rise to a fundamental research question identified as

(RQ2): how can exploration and sample complexity be enhanced? Addressing this question, Chapter 6 examined Novelty Search (NS), which belongs to a class of methods designed to promote exploration. NS is capable of learning competitive behaviors in various tasks without relying on environment rewards. Its underlying concept involves defining a behavior characteristic (BC) specific to the task and providing agents with novelty rewards. However, NS methods often rely on neighborhood models that store behaviors in an archive set, limiting their scalability and generalization to complex tasks requiring numerous policy evaluations. To tackle these challenges, we propose a function approximation method that utilizes auto-encoders to learn sparse representations of BC. Our experimental results demonstrate that the proposed approach outperforms traditional NS methods.

In an effort to enhance sample complexity in RL, Chapter 7 introduced the concept of constructing policy networks by leveraging domain knowledge, aiming to improve transparency, modularity, and data efficiency. We accomplished this by decomposing policy networks into adaptable and hand-designed components, based on an architecture inspired by computational biology, and apply it to solve a robotic task. The experimental results illustrated that learning such a modular network requires only one-tenth of the interactions compared to fully trained end-to-end recurrent networks.

In the final part of the thesis, the focus shifted to addressing practical challenges pertaining to SP tasks. SP tasks that are trained with MLE objectives suffer from *data* and *metric mismatch* problems. One alternative is to cast SP as RL task which enables models to be trained directly to optimize application-specific metrics. Now that most NLP tasks are reformulated this way, there is a lack of open-source libraries that provide simulated environments to test them consistently **(RQ3)** To address this gap, we presented *NLPGym*, a modular, easy-to-use toolkit that casts typical NLP tasks such as sequence labeling, multi-label classification and multiple choice question answering as RL tasks.

Extending this approach to modern large language models (LLMs) and also addressing the accessibility issues pertaining to Reinforcement Learning from Human Feedback (RLHF), we presented *RLALMs*, an open-source library to fine-tune LLMs of decoder type (eg. GPT2), encoder-decoder (eg. T5) type on any arbitrary reward function. These reward functions can be based on automated metrics like BLEU or learned reward models trained using human preferences. Alongside this library, we introduced a comprehensive benchmark called *Generative Reinforced-language Understanding Evaluation* (GRUE), which consists of seven text generation tasks, including summarization, translation, and dialogue generation. Additionally, to address **RQ4**, an on-policy algorithm specifically designed to handle the large action space associated with LLMs is proposed, making it easier to use in practice.

10.1 Outlook

While the presented work in this thesis has made several contributions toward practical and efficient methods for SDP tasks, there are still opportunities for further research to enhance its findings and at the same time, it is essential to acknowledge the limitations.

Let us examine the work presented in the first part of the thesis concerning the application of Echo State Networks (ESNs) as random context encoders for SDP tasks. Our notable contribution in Chapter 3 is the proposal of a neural cryptographic system that meets basic security requirements. However, it should be noted that the size of the output weights used as keys (ciphertext) for the system may exceed that of the original sequence (plaintext). To address this issue, it is necessary to explore techniques that encourage the model to learn sparse weights, thereby ensuring that the final ciphertext size is comparable. Recently, quantization techniques [301] have been proposed to

reduce the computational and memory costs of Deep Neural Networks (DNNs). These techniques involve representing weights and activations using low-precision data types, such as 8-bit integers (int8), instead of the traditional 32-bit floating-point format. Leveraging these techniques could be beneficial for the proposed security system. In the subsequent work (Chapter 4), we demonstrated the effectiveness of ESNs as context encoders for Named Entity Recognition (NER) tasks. Although ESNs are practical and efficient to train, there remains a slight performance gap when compared to fully trained RNNs. Previous research on deep ESNs suggests that incorporating multiple layers of reservoirs could potentially enhance performance. Furthermore, it would be intriguing to compare the performance with Transformer-based context encoders.

In Chapter 6, our work addressed the issues of scalability and generalizability in NS methods by replacing neighborhood models with auto-encoders. This work opens up several potential extensions. First, given that modern NLP systems have replaced RNNs with transformers, utilizing transformers as auto-encoders for modeling BCs could be an interesting avenue to explore. Second, the aspect of BCs is intriguing because, in most applications, it is easier to define BCs than reward functions. Therefore, exploring the design of BCs could be a future direction for leveraging domain knowledge to improve the learning process. Furthermore, in Chapter 7, our work proposes incorporating domain knowledge by decomposing policy networks into different components, including learnable and non-learnable forward and inverse kinematics components. This approach resulted in easily interpretable, modular, and efficient policy networks specifically designed for the reacher task. Extending this approach to other tasks, such as bipedal walking, hopping, and more, is possible but not straightforward and may require substantial engineering efforts.

Our contribution discussed in the Chapter 9, is particularly promising for future research. Since the initial release of *RL4LMs*, a multitude of works have emerged utilizing this framework. Notably, the NLPO algorithm, which we proposed, has been effectively employed to train guiding models that offer hints [302, 303] or critiques to enhance text generation tasks [304]. Furthermore, the *RL4LMs* library has been utilized to train LLMs capable of guiding agents in Dungeons and Dragons (DnD) [305].

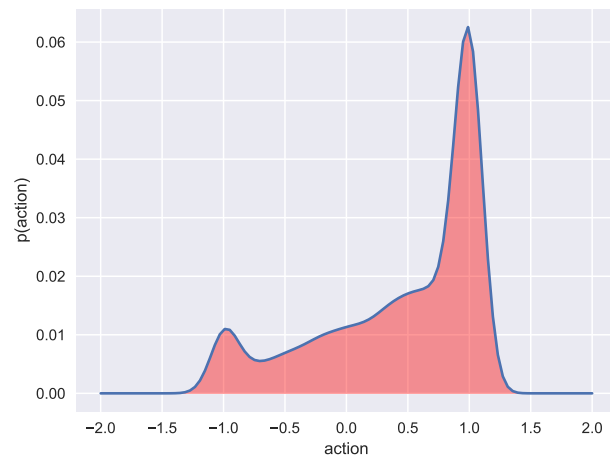
Despite the extensive work in RLHF, there remain several open research questions. Firstly, the current RLHF pipeline is not entirely online; rather, it follows an iterative process consisting of several steps: supervised fine-tuning, preference data collection, training reward models, and fine-tuning using RL. This decoupling of reward model training and RL fine-tuning limits their applicability in interactive applications. For instance, in the case of ChatGPT, user feedback is not immediately incorporated into fine-tuning the reward model or LLMs. We desire an approach that trains the reward model and language model together in an online fashion while users are actively interacting with the system. Such an approach is crucial for developing personalized LLMs. Secondly, the rewards provided by preference-based RLHF are often sparse and unreliable when it comes to evaluating long sentences. Moreover, relying on a single preference signal for judging sentences may be challenging for users, and more fine-grained feedback is desired [306]. Therefore, exploring alternative ways to train reward models using different forms of interaction is worth exploring.

Another interesting direction for future research involves the recent success of LLMs, which can handle various NLP tasks through a unified *text-in* and *text-out* approach. LLMs are trained on vast amounts of data and can perform tasks with zero-shot/few-shot learning. In contrast, RL is criticized for its sample inefficiency, as it aims to learn every task from scratch, and the concept of a foundational model capable of performing multiple tasks remains unexplored. One potential idea is to leverage the reasoning capabilities of LLMs and formulate RL tasks also in a text format. This approach could help improve the sample complexity of RL.

ESN for Partial Observability

category	parameter	deterministic		stochastic	
		Acrobot (discrete)	Mountain Car (discrete)	Acrobot (discrete)	Mountain Car (continuous)
SPSA	learning rate (l)	1e-6	1e-3	5e-5	5e-3
	scaling factor (c)	1e-1	1e-1	1e-1	1e-1
	L	10	100	10	100
	α	0.102	0.602	0.102	0.602
	γ	0.101	0.101	0.101	0.101
ESN	reservoir size	40	40	40	40
	input connectivity	0.7	0.3	0.3	0.7
	reservoir connectivity	0.7	0.3	0.7	0.7
	output scaling	0.1	0.1	1e-5	1e-2
	spectral radius	1.0	1.0	1.0	1.0
	leaking rate	0.3	0.3	0.3	0.3
RL	discount factor	0.99	1.0	0.99	0.99
	learning rate	1e-2	1e-2	1e-3	1e-3

Table A.1: **Summary of Hyperparameters:** Hyperparameters and their values for different experiments.



(a)

Figure A.1: **Visualization of MC policy:** Visualization of a Gaussian policy learned for the mountain car task.

NLPGym

B.1 Demo Scripts

Sample Interactions

```
from nlp_gym.data_pools.custom_question_answering_pools import QASC
from nlp_gym.envs.question_answering.env import QAEVn
```

```
# data pool
pool = QASC.prepare("train")

# custom answering env
env = QAEVn()
for sample, weight in pool:
    env.add_sample(sample)

# play an episode
done = False
state = env.reset()
total_reward = 0
while not done:
    action = env.action_space.sample()
    state, reward, done, info = env.step(action)
    total_reward += reward
    env.render()
    print(f"Action: {env.action_space.ix_to_action(action)}")
print(f"Total reward: {total_reward}")
```

```
Step 0
Question: Machines can use gasoline to do what?
Fact: a gasoline lawn mower converts gasoline into motion
Fact: Machines mow down forests much as a lawn mower cuts grass.
Choice A: energy
Action: CONTINUE
Step 1
Question: Machines can use gasoline to do what?
Fact: a gasoline lawn mower converts gasoline into motion
Fact: Machines mow down forests much as a lawn mower cuts grass.
Choice A: energy
Fact: a gasoline lawn mower converts gasoline into motion
Fact: Machines mow down forests much as a lawn mower cuts grass.
Choice B: destroy matter
Action: ANSWER
Total reward: 0.0
```

Online Learning

We illustrate below the usage of the environment in an online setting. In each iteration, one sample (data point) is added to the environment for sampling. This allows us to develop interactive algorithms (for instance, active learning systems) that involve human interaction. A complete run of such an algorithm corresponds to precisely one pass over the entire dataset.

```
from nlp_gym.envs.seq_tagging.env import SeqTagEnv
from nlp_gym.data_pools.custom_seq_tagging_pools import UDPosTaggingPool
from stable_baselines.common.policies import MlpPolicy
from stable_baselines import PPO1
from nlp_gym.envs.seq_tagging.reward import EntityF1Score
from nlp_gym.envs.seq_tagging.featurizer import DefaultFeaturizerForSeqTagging
from nlp_gym.metrics.seq_tag import EntityScores
import tqdm

def predict(model, sample):
    done = False
    obs = env.reset(sample)
    predicted_label = []
    while not done:
        action, _ = model.predict(obs)
        obs, _, done, _ = env.step(action)
        predicted_label.append(env.action_space.ix_to_action(action))
    return predicted_label

# data pool
data_pool = UDPosTaggingPool.prepare(split="train")

# reward function
reward_fn = EntityF1Score(dense=True, average="micro")

# seq tag env
env = SeqTagEnv(data_pool.labels(), reward_function=reward_fn)

# observation featurizer
feat = DefaultFeaturizerForSeqTagging(env.action_space, embedding_type="fasttext")
env.set_featurizer(feat)

# PPO model
model = PPO1(MlpPolicy, env, verbose=0)

# train loop that goes over each sample only once
running_match_score = 0
for ix, (sample, _) in enumerate(tqdm.tqdm(data_pool)):

    # run the sample through the model and get predicted label
    predicted_label = predict(model, sample)

    # after few epochs, predicted_label can be used as pre-annotated input
    # then the user can just correct it
    # to reduce human efforts

    # get annotated label from user (just simulated for now)
    annotated_label = sample.oracle_label

    # match score
    match_ratio = EntityScores()(annotated_label, predicted_label)["f1"]
    running_match_score += match_ratio
```

```

# add the new sample to the environment
sample.oracle_label = annotated_label
env.add_sample(sample)

# train agent for few epochs
model.learn(total_timesteps=1e+2)

if (ix+1) % 50 == 0:
    print(f"Running match score {running_match_score/50}")
    running_match_score = 0.0

```

B.2 Default Featurizers

The toolkit provides its users with simple observation featurizers based on pre-trained word embeddings to get started with the environments without much setup. They are easily extendable and replaceable with custom components (see B.3)

Sequence tagging (ST): In this environment, observation at any time step is a word/token and a tagged label of the previous word. The word is vectorized into one of the pre-trained word embeddings such as fasttext [139], byte-pair [236] and flair embeddings [215]. On the contrary, the tagged label is converted to a one-hot encoded representation since the label vocabulary is known. In the end, the observation vector is simply a concatenation of the word vector and the label vector.

Multi-Label Classification (MLC): The observation at any time step in this environment is a sentence and a generated label sequence up to the time step. The sentence is converted to a fixed-length representation by pooling over its corresponding word embeddings (pre-trained embeddings). The label sequence is converted to a Bag-of-Words representation (BoW). To obtain the observation vector, the representations of these two are concatenated together.

Question-answering (QA): For the QA environment, the observation is a triplet consisting of question, choice and facts. The toolkit offers two types of featurizers: *simple* and *informed*. The *simple* featurizer converts the observation into a concatenated vector of sentence representations of the question, choice and facts. On the contrary, *informed* featurizer leverages the fact that the correct choice should be semantically similar to the question and the given facts than other choices. To this end, the observation is simply a 2-D vector consisting of cosine similarity between choice-question and choice-facts.

B.3 Custom Components

The environments provide ready-to-use default implementations for observation featurizers, reward functions and datasets. However, the toolkit is modular in a way that users can plug-in their own implementations of these components. This gives the flexibility to implement some of the components to have trainable components (e.g. observation featurizer) that can be optimized either end-to-end or by pre-training. Fig shows the class diagram of NLP Gym toolkit capturing relationships between different components.

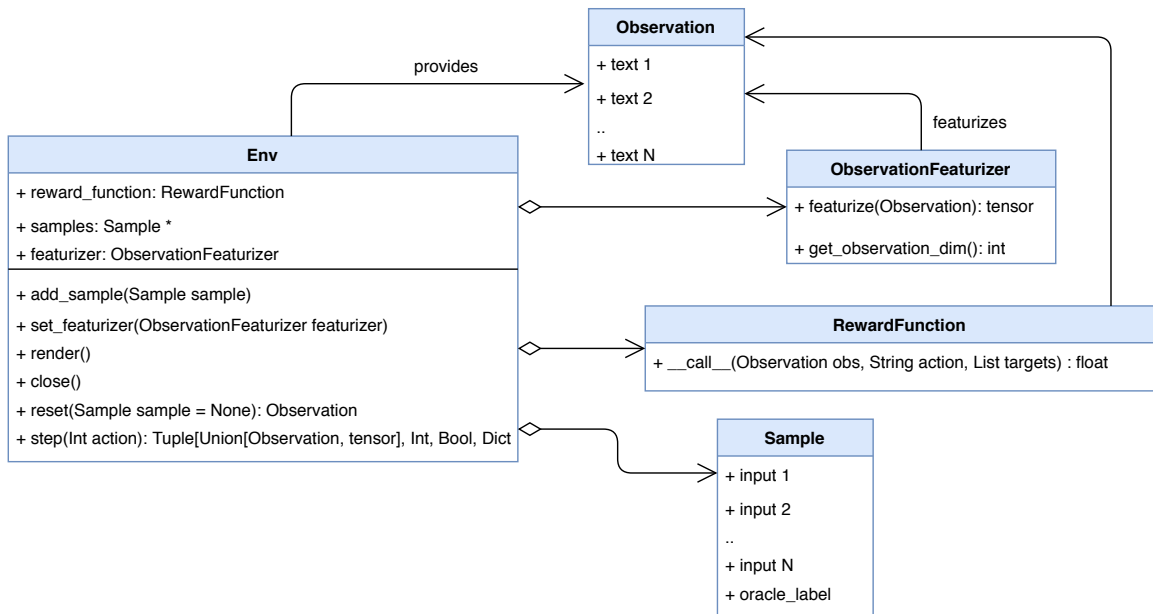


Figure B.1: Class diagram of NLP Gym:

Datasets

For instance, to create a custom sequence tagging dataset, users must provide a list of samples (data points) and possible labels (that correspond to agent actions). These samples should be instances of the *Sample* data class as shown below.

```

class Sample:
    """
    Dataclass for holding datapoints

    Attributes:
        input_text - textual input
        oracle_label - true label for the given data point
    """
    input_text: str
    oracle_label: List[str]
  
```

With this setup, the creation of a custom dataset is straightforward, as shown below:

```

custom_dataset = SeqTaggingPool(samples, possible_labels=[""])
  
```

Reward Function

Users can define their own custom reward function by sub-classing *RewardFunction* which is a *Callable* that takes *BaseObservation*, *current action*, *targets* and returns a scalar reward.

```

from nlp_gym.envs.common.observation import BaseObservation
from nlp_gym.envs.common.reward import RewardFunction

class MyRewardFunction(RewardFunction)
    def __call__(self, observation: BaseObservation, action: str, targets: List[str]) -> float:
        """
  
```



```

My custom reward function
Args:
    observation (BaseObservation): current observation at t
    action (str): current action at t
    targets (List[str]): targets of the current sample

Returns:
    - a scalar reward
"""
pass

```

Observation Featurizer

Similarly, observation featurizer can have its custom implementation by sub-classing it from *BaseObservationFeaturizer* as shown below. Each featurizer must implement *featurize()* and *get_observation_dim()* by default. Additionally, based on the task, some more functions need to be provide by user. For instance, for the sequence tagging environment, the featurizer must additionally implement *init_on_reset()* which is called by the environment on *reset()*.

```

class BaseObservationFeaturizer(ABC):

    @abstractmethod
    def featurize(self, observation: BaseObservation) -> torch.Tensor:
        raise NotImplementedError

    def get_observation_dim(self) -> int:
        """
        Returns the observation dim
        """
        return self.get_input_dim() + self.get_context_dim()

# for sequence tagging environment
class ObservationFeaturizer(BaseObservationFeaturizer):

    @abstractmethod
    def init_on_reset(self, input_text: Union[List[str], str]):
        """
        Takes an input text (sentence) or list of token strings and featurizes it or prepares it
        This function would be called in env.reset()
        """
        raise NotImplementedError

```



```

-----
"text": "Wei Ligang , a rising star of modern art in China , just had an
        exhibition in mid-July , 2005 in Hong Kong .",
"true_label": ["PROPN", "PROPN", "PUNCT", "DET", "VERB", "NOUN", "ADP", "ADJ",
               "NOUN", "ADP", "PROPN", "PUNCT", "ADV", "VERB", "DET", "NOUN",
               "ADP", "PROPN", "PUNCT", "NUM", "ADP", "PROPN", "PROPN", "PUNCT"],
"predicted_label": ["PROPN", "PROPN", "PUNCT", "DET", "VERB", "NOUN", "ADP", "ADJ",
                   "NOUN", "ADP", "PROPN", "PUNCT", "ADV", "VERB", "DET", "NOUN",
                   "ADP", "PROPN", "PUNCT", "NUM", "ADP", "PROPN", "PROPN",
                   "PUNCT"],
"total_reward": 1.0
-----
"text": "What if Google Morphed Into GoogleOS ?",
"true_label": ["PRON", "SCONJ", "PROPN", "VERB", "ADP", "PROPN", "PUNCT"]
"predicted_label": ["PRON", "CCONJ", "PROPN", "VERB", "ADP", "PROPN", "PUNCT"]
"total_reward": 0.8571428571428571
-----
"text": "They know that the American advent implies for them a demotion ,
        and an elevation of the Shiites and Kurds , and
        they refuse to go quietly .",
"true_label": ["PRON", "VERB", "SCONJ", "DET", "ADJ", "NOUN", "VERB", "ADP",
               "PRON", "DET", "NOUN", "PUNCT", "CCONJ", "DET", "NOUN", "ADP",
               "DET", "PROPN", "CCONJ", "PROPN", "PUNCT", "CCONJ",
               "PRON", "VERB", "PART", "VERB", "ADV", "PUNCT"],
"predicted_label": ["PRON", "VERB", "PRON", "DET", "ADJ", "NOUN", "VERB", "ADP",
                   "PRON", "DET", "NOUN", "PUNCT", "CCONJ", "DET", "NOUN",
                   "ADP", "DET", "PROPN", "CCONJ", "PROPN", "PUNCT", "CCONJ",
                   "PRON", "VERB", "ADP", "VERB", "ADV", "PUNCT"],
"total_reward": 0.9285714285714286
-----
"text": "...",
"true_label": ["SYM"]
"predicted_label": ["PUNCT"]
"total_reward": 0.0
-----
"text": "Can police trace a cell phone even if it is switched off ?"
"true_label": ["AUX", "NOUN", "VERB", "DET", "NOUN", "NOUN", "ADV",
               "SCONJ", "PRON", "AUX", "VERB", "ADP", "PUNCT"]
"predicted_label": ["AUX", "ADJ", "NOUN", "DET", "NOUN", "NOUN",
                   "ADV", "CCONJ", "PRON", "AUX", "VERB", "ADP", "PUNCT"]
"total_reward": 0.7692307692307693
-----

```

Multi-Label Classification

```

-----
"text": "FED EXPECTED TO ADD RESERVES VIA CUSTOMER RPS\n
The Federal Reserve is expected to enter\n
the government securities market to add reserves via customer\n
repurchase agreements, economists said.\n
They expected the amount to total around 1.5 billion to two\n
billion dlrs.\n
Economists added that the low rate on federal funds\n
indicates the Fed is unlikely to add funds aggressively through\n
overnight system repurchases, unless it feels the need to calm\n
volatile financial markets.\n
Federal funds were trading at 7-1/8 pct, down from\n
yesterday's average of 7.61 pct.\n \n\n",
"true_label": ["interest", "money-fx"],
"predicted_label": ["money-fx", "interest"],
"total_reward": 1.0
-----

```

```

-----
"text": "NERCI <NER> UNIT CLOSES OIL/GAS ACQUISITION\n
Nerco Inc said its oil and gas\n
unit closed the acquisition of a 47 pct working interest in the\n
Broussard oil and gas field from <Davis Oil Co> for about 22.5\n
mln dlrs in cash.\n
Nerco said it estimates the field's total proved developed\n
and undeveloped reserves at 24 billion cubic feet, or\n
equivalent, of natural gas, which more than doubles the\n
company's previous reserves.\n
The field is located in southern Louisiana.\n \n\n",
"true_label": ["acq", "crude", "nat-gas"],
"predicted_label": ["crude"],
"total_reward": 0.5
-----

```

```

-----
"text": "SOFTWARE SERVICES OF AMERICA INC <SSOA.O> NET\n
3rd qtr Feb 28\n
Shr profit 14 cts vs loss four cts\n
Net profit 311,994 vs loss 66,858\n
Revs 2,229,273 vs 645,753\n
Nine mths\n
Shr profit 51 cts vs profit two cts\n
Net profit 1,126,673 vs profit 42,718\n
Revs 7,277,340 vs 1,378,372\n \n\n",
"true_label": ["earn"],
"predicted_label": ["earn"],
"total_reward": 1.0
-----

```

```

-----
"text": "PORTUGUESE CONSUMER PRICES UP 1.4 PCT IN MARCH\n
Portugal's consumer prices rose 1.4 pct\n
in March after a one pct increase in February and a 1.2 pct\n
rise in March 1986, the National Statistics Institute said.\n
The consumer price index (base 100 for 1976) rose to 772.0\n
from 761.3 in February and compared with 703.4 in March 1986.\n
This gave a year-on-year March inflation rate of 9.8 pct\n
against 9.5 pct in February and 12.2 pct in March 1986.\n
Measured as an annual average rate, inflation in March was\n
10.9 pct compared with 11.1 pct in February. The government\n
forecasts annual inflation of about eight pct this year.\n \n\n",
"true_label": ["cpi"],
"predicted_label": ["money-supply"],
"total_reward": 0.0
-----

```

```

-----
"text": "a relay channel is one in which a source and destination use an
intermediate relay station in order to improve communication rates
we propose the study of relay channels with classical inputs and
quantum outputs and prove that a partial decode and forward strategy
is achievable we divide the channel uses into many blocks and build
codes in a randomized , block markov manner within each block the relay
performs a standard holevo schumacher westmoreland quantum measurement
on each block in order to decode part of the source 's message and then
forwards this partial message in the next block the destination performs
a novel sliding window quantum measurement on two adjacent blocks in
order to decode the source 's message this strategy achieves non trivial
rates for classical communication over a quantum relay channel",
"true_label": ["quant-ph", "cs.IT", "math.IT"],
"predicted_label": ["cs.IT", "math.IT"],
"total_reward": 0.8
-----

```

```

"text": "recently , social phenomena have received a lot of attention not only
from social scientists , but also from physicists , mathematicians and
computer scientists , in the emerging interdisciplinary field of complex
system science opinion dynamics is one of the processes studied ,
since opinions are the drivers of human behaviour , and play a crucial
role in many global challenges that our complex world and societies are
facing global financial crises , global pandemics , growth of cities ,
urbanisation and migration patterns , and last but not least important ,
climate change and environmental sustainability and protection opinion
formation is a complex process affected by the interplay of different
elements , including the individual predisposition , the influence of
positive and negative peer interaction \\( social networks playing
a crucial role in this respect \\) , the information each individual is
exposed to ,and many others several models inspired from those in use in
physics have been developed to encompass many of these elements , and to
allow for the identification of the mechanisms involved in the opinion
formation process and the understanding of their role , with the
practical aim of simulating opinion formation and spreading under various
conditions these modelling schemes range from binary simple models such
as the voter model, to multi dimensional continuous approaches here ,
we provide a review of recent methods , focusing on models employing
both peer interaction and external information , and emphasising the
role that less studied mechanisms , such as disagreement , has in
driving the opinion dynamics",
"true_label": ["physics.soc-ph", "cs.SI"],
"predicted_label": ["physics.soc-ph", "cs.SI"],
-----

```

```

"text": "we propose a new platform for implementing secure wireless ad hoc
networks our proposal is based on a modular architecture ,
with the software stack constructed directly on the ethernet layer
within our platform we use a new security protocol that we designed
to ensure mutual authentication between nodes and a secure key exchange
the correctness of the proposed security protocol is
ensured by guttman 's authentication tests",
"true_label": ["cs.CR", "cs.NI"],
"predicted_label": ["cs.CR", "cs.NI"],
"total_reward": 1.0
-----

```

Question Answering

```
-----  
"question": "What can increase the chances of flooding?",  
"facts": ["if weather is stormy then there is a greater chance of rain",  
          "Rain is good, but lots of rain causes destructive flooding."],  
"choices": {  
  "A": "filter feeders",  
  "B": "low tide",  
  "C": "permeable walls",  
  "D": "fortifying existing levees",  
  "E": "higher corn prices",  
  "F": "stormy weather",  
  "G": "Being over land",  
  "H": "feedback mechanisms"  
},  
"true_label": "F",  
"predicted_label": "F",  
"total_reward": 1.0  
-----
```

```
"question": "How can animals like cyptosporidium be classified?",  
"facts": ["Protozoa can be classified on the basis of how they move.",  
          "Cryptosporidium parvum is the hardest protozoa to kill."],  
"choices": {  
  "A": "prokaryotic cells",  
  "B": "holding nutrients",  
  "C": "Laboratory",  
  "D": "how they move",  
  "E": "eukaryotic cells",  
  "F": "coelenterates",  
  "G": "melanin content",  
  "H": "angiosperm"  
},  
"true_label": "D",  
"predicted_label": "A"  
-----
```

```
"question": "What can magnets be used to do?",  
"facts": ["a compass is used for determining direction",  
          "Magnets are used in compasses."],  
"choices": {  
  "A": "capture prey",  
  "B": "moving over land",  
  "C": "feedback mechanisms",  
  "D": "Destroy magnets",  
  "E": "reproduce",  
  "F": "help other species benefit",  
  "G": "Direct a play",  
  "H": "Determine direction"  
},  
"true_label": "H",  
"predicted_label": "H",  
"total_reward": 1.0  
-----
```

RL4LMs

C.1 Experimental Details

C.1.1 Crowdsourcing Details

Qualification round We ran a qualification round using the IMDB task. We opened the qualification around to users from {AU, CA, NZ, GB, US} with 5K prior approved HITs and a minimum acceptance rate of 97% on their previous HITs. We gathered judgments over 600 generations from 3 annotators per generation. One of the authors of this paper also completed 17 random HITs to serve as a proxy for “ground truth.” After gathering these annotations, we selected workers who: 1) didn’t significantly disagree with other annotators on the same instance more than 20% of the time; 2) who completed at least 5 HITs; 3) who didn’t disagree with the author annotator on the 17 HITs by more than 1 point; and 4) (likely) spent a reasonable amount of time reading the instructions/examples provided. In the end, 56 annotators were qualified. Additional per-task details are provided in the per-task sections of the Appendix.

Compensation details As per Amazon Mechanical Turk policy, annotators were compensated on a per-HIT basis. In addition, we used a timing script to estimate hourly wages to ensure our target of \$15/hr was met. In cases where this minimum hourly rate was not met, we manually assigned bonuses.

C.1.2 GRUE Experiment Setup

We benchmark 5 training algorithms on 6 tasks (see Table 9.1) using either an encoder model (eg. GPT-2) or encoder-decoder model (eg. T5). We train policies using PPO, NLPO with variations of whether supervised pre-training is applied before RL fine-tuning and compare against supervised policy. The choice of LM is based on the type of task. For IMDB text continuation, we use GPT-2 and T5 for rest of the tasks. We use two separate LM models as actor and critics networks (i.e. no shared layers) in which the critic network has an additional linear layer mapping last token’s hidden representation to a scalar value. We use AdamW optimizer [307] with fixed learning rate and no scheduling.

C.1.3 IMDB

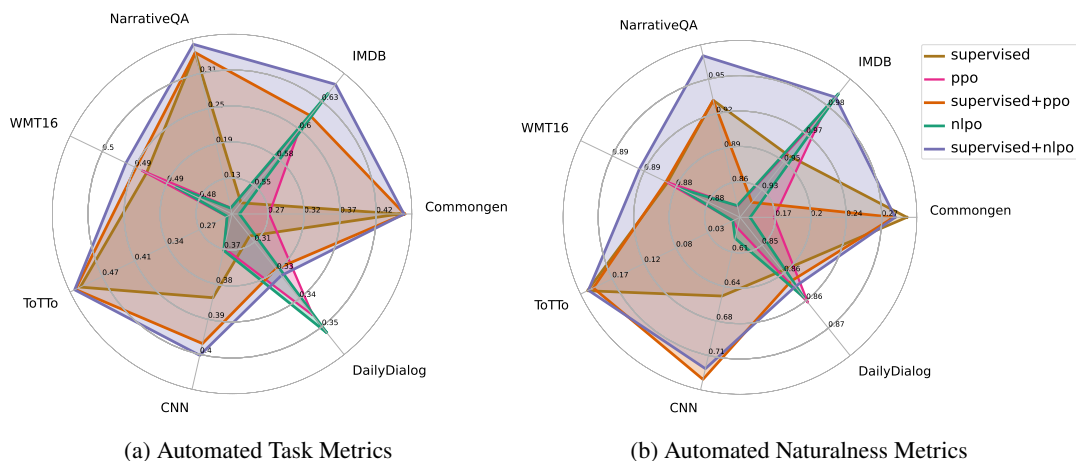


Figure C.1: **Results summary:** Summarized results via automated metrics across all 7 GRUE tasks for each of the 5 algorithms we consider, and human participant studies for the 5 tasks suitable for human studies. We break up the metrics into task-specific, e.g. average positive sentiment for IMDB task, and naturalness metrics, such as perplexity and human perceived coherence for the human rated metrics. This plot differs from Figure 9.2 as this one averages over multiple reward functions per each task.

Setup We consider IMDB dataset for the task of generating text with positive sentiment. The dataset consists of 25k training, 5k validation and 5k test examples of movie review text with sentiment labels of positive and negative. The input to the model is a partial movie review text (upto 64 tokens) that needs to be completed (generating 48 tokens) by the model with a positive sentiment while retaining fluency. For RL methods, we use a sentiment classifier [299] that is trained on pairs of text and labels as a reward model which provides sentiment scores indicating how positive a given piece of text is. For supervised Seq2Seq baselines, we consider only the examples with positive labels. We chose GPT-2 as LM for this task as it is more suited for text continuation than encoder-decoder LMs (eg. T5). We use top-k sampling with $K = 50$ as the decoding method and for fair comparison, we keep this setting for all methods. For PPO and NLPO models, we train for $64k$ steps in total and update policy and value networks every 1280 steps with a mini-batch size of 64 and epochs of 5 per update. We apply adaptive KL controllers with different target KLs of 0.02, 0.05, 0.2, 0.5, 1.0, inf with an initial KL co-efficient of $\beta = 0.01$. Table C.1 provides an in-depth summary of all hyperparameters and other implementation details.

Results and Discussion

Target KL ablation Fig C.2 shows learning curves for PPO and NLPO in terms of episodic training reward, corpus level sentiment scores and perplexity scores on validation set averaged for 5 random seeds. It is seen that higher target KL of 0.2 is desired to achieve higher rewards but results in drifting away from pre-trained LM and loses fluency. Therefore, a lower target KL (0.02 or 0.05) is required to keep the LM closer to original LM. This is also seen in Table C.2 where we presented a comparative analysis of final performance of all models.

Model Params	value
supervised	batch size: 64 epochs: 10 learning rate: 0.00001
ppo	steps per update: 1280 total number of steps: 64000 batch size: 64 epochs per update: 5 learning rate: 0.000001 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5
nlpo	steps per update: 1280 total number of steps: 64000 batch size: 64 epochs per update: 5 learning rate: 0.000001 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 top mask ratio: 0.9 target update iterations: 5
decoding	sampling: true top k: 50 min length: 48 max new tokens: 48
tokenizer	padding side: left truncation side: left max length: 64

Table C.1: **IMDB Hyperparams**: Table shows a list of all hyper-parameters and their settings

Training data size ablation We vary the amount of data used to train the reward classifier and the supervised baseline model to understand whether it is more efficient to gather data to improve reward model or to gather expert demonstrations for supervised learning. As observed in Table C.3, improving the quality of reward function increases the performance on the overall task better than training with more data for supervised training, indicating that improving reward models is efficient than collect expert demonstrations for supervised training from a data efficiency perspective.

Discount factor ablation To understand the effect of discounted vs undiscounted (bandit) environments, we report sentiment and perplexity scores for different values of discount factor (0.5, 0.95 and 1.0) in Table C.4 and observe that using a bandit environment (discount factor of 1.0) results

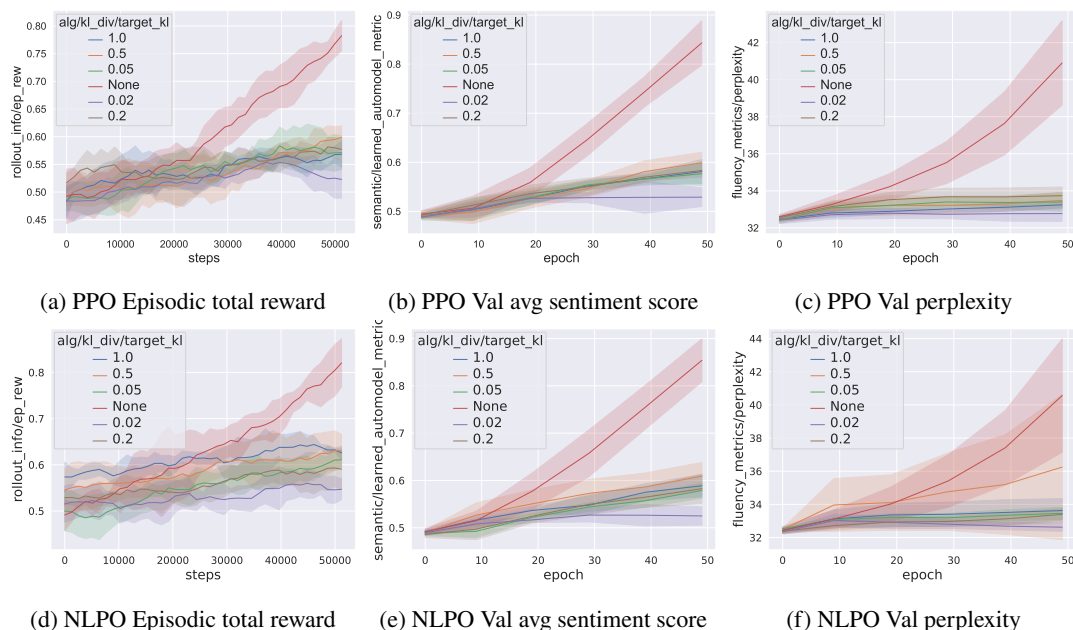


Figure C.2: **Learning Curves:** Averaged learning curves over 5 different runs by varying target KL, shaded regions indicate one standard deviation. (a) shows the rollout episodic total reward during training (b) shows evolution of sentiment scores on the validation split (c) shows evolution of perplexity on the validation split. From (a) and (b), it is seen that higher target KL (0.2) is desired to achieve higher rewards. However, this setting drifts away from the original LM too much and loses fluency. Therefore a lower target KL (0.02 or 0.05) is required to keep the model closer to original LM. Similar trends hold for NLPO but when compared to PPO, it retains lower perplexities and is more stable even with higher KL targets, enabling higher sentiment scores.

in performance loss in the case of NLPO and reward hacking in the case of PPO, indicating that discounted setting (with 0.95) is desired.

NLPO params Table. C.5 shows ablation on different hyperparameters in NLPO algorithm.

Human Participant Study Figure C.3 shows the IMDB instructions, example, and interface used both for the qualification round, and then later, for the human evaluation experiments. Tables C.6, C.7 show averaged results, annotator agreement, and the results of statistical significance tests to determine which models output better generations when rated by humans.

Target-KL	Semantic and Fluency Metrics		Diversity Metrics						
	Sentiment Score \uparrow	Perplexity \downarrow	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂
Zero-Shot	0.489 \pm 0.006	32.171 \pm 0.137	0.682 \pm 0.001	0.042 \pm 0.001	0.294 \pm 0.001	8.656 \pm 0.004	13.716 \pm 0.003	5063 \pm 14.832	47620 \pm 238
Supervised	0.539 \pm 0.004	35.472 \pm 0.074	0.682 \pm 0.001	0.047 \pm 0.001	0.312 \pm 0.002	8.755 \pm 0.012	13.806 \pm 0.016	5601 \pm 57	51151 \pm 345
PPO									
0.02	0.530 \pm 0.021	32.921 \pm 0.322	0.680 \pm 0.002	0.042 \pm 0.001	0.293 \pm 0.002	8.642 \pm 0.015	13.676 \pm 0.025	5042 \pm 135	47554 \pm 418
0.05	0.578 \pm 0.022	33.469 \pm 0.532	0.660 \pm 0.021	0.044 \pm 0.002	0.287 \pm 0.011	8.553 \pm 0.130	13.389 \pm 0.356	5352 \pm 251	46158 \pm 2568
0.2	0.585 \pm 0.006	33.627 \pm 0.236	0.665 \pm 0.005	0.044 \pm 0.001	0.287 \pm 0.008	8.584 \pm 0.055	13.438 \pm 0.124	5315 \pm 171	46834 \pm 1469
0.5	0.605 \pm 0.023	33.497 \pm 0.447	0.666 \pm 0.013	0.043 \pm 0.002	0.287 \pm 0.008	8.575 \pm 0.073	13.484 \pm 0.244	5230 \pm 363	46483 \pm 1318
1.0	0.579 \pm 0.025	33.161 \pm 0.117	0.676 \pm 0.002	0.042 \pm 0.001	0.291 \pm 0.007	8.625 \pm 0.041	13.625 \pm 0.089	5027 \pm 173	47082 \pm 1375
inf	0.847 \pm 0.039	40.650 \pm 2.154	0.566 \pm 0.038	0.035 \pm 0.006	0.200 \pm 0.025	7.715 \pm 0.289	11.763 \pm 0.496	4380 \pm 775	32462 \pm 5020
PPO+supervised									
0.5	0.617 \pm 0.011	34.078 \pm 0.253	0.672 \pm 0.003	0.047 \pm 0.002	0.308 \pm 0.007	8.725 \pm 0.054	13.711 \pm 0.068	5513 \pm 173	50410 \pm 1277
inf	0.829 \pm 0.049	46.906 \pm 2.168	0.615 \pm 0.037	0.037 \pm 0.005	0.225 \pm 0.04	8.072 \pm 0.373	12.537 \pm 0.707	4480 \pm 443	35583 \pm 6631
NLPO									
0.02	0.530 \pm 0.020	32.824 \pm 0.227	0.680 \pm 0.002	0.043 \pm 0.001	0.295 \pm 0.004	8.658 \pm 0.031	13.689 \pm 0.050	5129 \pm 169	47863 \pm 840
0.05	0.581 \pm 0.017	32.298 \pm 0.362	0.674 \pm 0.004	0.043 \pm 0.001	0.295 \pm 0.008	8.647 \pm 0.040	13.638 \pm 0.099	5110 \pm 121	47911 \pm 1478
0.2	0.591 \pm 0.012	32.602 \pm 0.261	0.663 \pm 0.015	0.044 \pm 0.001	0.287 \pm 0.012	8.586 \pm 0.096	13.442 \pm 0.240	5314 \pm 166	46665 \pm 2124
0.5	0.611 \pm 0.014	32.241 \pm 0.932	0.650 \pm 0.035	0.042 \pm 0.002	0.271 \pm 0.013	8.389 \pm 0.270	13.081 \pm 0.741	5159 \pm 536	43840 \pm 1217
1.0	0.637 \pm 0.013	32.667 \pm 0.631	0.677 \pm 0.014	0.044 \pm 0.002	0.288 \pm 0.010	8.588 \pm 0.100	13.484 \pm 0.236	5205 \pm 189	46344 \pm 2688
inf	0.859 \pm 0.041	37.553 \pm 3.22	0.567 \pm 0.037	0.036 \pm 0.007	0.205 \pm 0.034	7.725 \pm 0.326	11.772 \pm 0.571	4568 \pm 1046	33056 \pm 6365
NLPO+supervised									
1.0	0.645 \pm 0.027	33.191 \pm 0.187	0.656 \pm 0.014	0.049 \pm 0.005	0.3 \pm 0.026	8.648 \pm 0.213	13.396 \pm 0.331	6053 \pm 609	49468 \pm 4745
inf	0.853 \pm 0.106	36.812 \pm 0.207	0.427 \pm 0.081	0.054 \pm 0.019	0.205 \pm 0.077	6.82 \pm 1.0	9.684 \pm 1.475	7788 \pm 2718	35213 \pm 13349

Table C.2: **Target KL Ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. It is seen from perplexity scores that a lower target KL constraint is desired to keep the model closer to the original model. On the otherhand, a higher target KL yields higher sentiment scores at the cost of fluency. inf KL penalty (target KL of inf), model simply learns to generate positive phrases (eg: "I highly recommend this movie to all!", "worth watching") regardless of the context

Perc Data (size)	Semantic and Fluency Metrics		Diversity Metrics						
	Sentiment Score \uparrow	Perplexity \downarrow	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂
Zero-Shot	0.489 \pm 0.006	32.371 \pm 0.137	0.682 \pm 0.001	0.042 \pm 0.001	0.294 \pm 0.001	8.656 \pm 0.004	13.716 \pm 0.003	5063 \pm 14.832	47620 \pm 238
Supervised									
0.0 (0k)	0.489 \pm 0.006	32.371 \pm 0.137	0.682 \pm 0.001	0.042 \pm 0.001	0.294 \pm 0.001	8.656 \pm 0.004	13.716 \pm 0.003	5063 \pm 14	47620 \pm 238
0.1 (1k)	0.531 \pm 0.005	34.846 \pm 0.123	0.685 \pm 0.001	0.045 \pm 0.001	0.313 \pm 0.004	8.775 \pm 0.023	13.854 \pm 0.032	5215 \pm 62	51125 \pm 685
0.5 (5k)	0.536 \pm 0.006	35.008 \pm 0.229	0.684 \pm 0.001	0.047 \pm 0.000	0.314 \pm 0.002	8.764 \pm 0.010	13.837 \pm 0.0178	5489 \pm 44	51284 \pm 576
1.0 (10k)	0.539 \pm 0.004	35.472 \pm 0.074	0.682 \pm 0.001	0.047 \pm 0.001	0.312 \pm 0.002	8.755 \pm 0.012	13.806 \pm 0.016	5601 \pm 57	51151 \pm 345
PPO									
0.0 (0k)	0.492 \pm 0.01	33.57 \pm 0.323	0.69 \pm 0.02	0.047 \pm 0.001	0.321 \pm 0.015	8.816 \pm 0.149	13.866 \pm 0.36	5629 \pm 240	52911 \pm 1786
0.1 (2k)	0.598 \pm 0.017	35.929 \pm 1.397	0.698 \pm 0.009	0.051 \pm 0.003	0.339 \pm 0.012	8.968 \pm 0.083	14.013 \pm 0.158	6173 \pm 360	55918 \pm 2641
0.5 (10k)	0.593 \pm 0.026	35.95 \pm 2.177	0.666 \pm 0.073	0.049 \pm 0.003	0.314 \pm 0.046	8.635 \pm 0.634	13.432 \pm 1.173	5882 \pm 356	51403 \pm 9297
1.0 (20k)	0.605 \pm 0.023	33.497 \pm 0.447	0.666 \pm 0.013	0.043 \pm 0.002	0.287 \pm 0.008	8.575 \pm 0.073	13.484 \pm 0.244	5230 \pm 363	46483 \pm 1318
NLPO									
0.0 (0k)	0.487 \pm 0.01	32.572 \pm 0.165	0.685 \pm 0.003	0.043 \pm 0.001	0.299 \pm 0.003	8.691 \pm 0.023	13.787 \pm 0.034	5126 \pm 177	48475 \pm 491
0.1 (2k)	0.599 \pm 0.007	33.536 \pm 0.378	0.67 \pm 0.01	0.043 \pm 0.001	0.289 \pm 0.009	8.608 \pm 0.061	13.576 \pm 0.192	5125 \pm 220	46755 \pm 1449
0.5 (10k)	0.617 \pm 0.021	33.409 \pm 0.354	0.668 \pm 0.005	0.041 \pm 0.001	0.281 \pm 0.006	8.552 \pm 0.044	13.533 \pm 0.091	4926 \pm 183	45256 \pm 1022
1.0 (20k)	0.637 \pm 0.013	32.667 \pm 0.631	0.677 \pm 0.014	0.044 \pm 0.002	0.288 \pm 0.010	8.588 \pm 0.100	13.484 \pm 0.236	5205 \pm 189	46344 \pm 2688

Table C.3: **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, data budget ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table measures performance differences as a function of the fraction of the dataset that has been used. In the case of the RL approaches, this measures how much data is used to train the reward classifier, and for the supervised method it directly measures fraction of positive reviews used for training. We note that using even a small fraction of data to train a reward classifier proves to be effective in terms of downstream task performance while this is not true for supervised approaches. This lends evidence to the hypothesis that adding expending data budget on a reward classifier is more effective than adding more gold label expert demonstrations.

Appendix C RL4LMs

Gamma	Semantic and Fluency Metrics		Diversity Metrics						
	Sentiment Score \uparrow	Perplexity \downarrow	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂
Zero-Shot	0.489 \pm 0.006	32.371 \pm 0.137	0.682 \pm 0.001	0.042 \pm 0.001	0.294 \pm 0.001	8.656 \pm 0.004	13.716 \pm 0.003	5063 \pm 14.832	47620 \pm 238
PPO									
0.5	0.511 \pm 0.023	35.945 \pm 0.92	0.69 \pm 0.001	0.044 \pm 0.002	0.304 \pm 0.007	8.726 \pm 0.041	13.793 \pm 0.055	5304 \pm 285	49668 \pm 1496
0.95	0.605 \pm 0.023	33.497 \pm 0.447	0.666 \pm 0.013	0.043 \pm 0.002	0.287 \pm 0.008	8.575 \pm 0.073	13.484 \pm 0.244	5230 \pm 363	46483 \pm 1318
1.0	0.651 \pm 0.05	41.035 \pm 2.885	0.691 \pm 0.017	0.042 \pm 0.004	0.295 \pm 0.031	8.697 \pm 0.237	13.563 \pm 0.396	5127 \pm 460	48319 \pm 5650
NLPO									
0.5	0.49 \pm 0.01	37.279 \pm 5.137	0.688 \pm 0.01	0.045 \pm 0.002	0.312 \pm 0.016	8.746 \pm 0.113	13.873 \pm 0.25	5395 \pm 192	50828 \pm 2506
0.95	0.637 \pm 0.013	32.667 \pm 0.631	0.677 \pm 0.014	0.044 \pm 0.002	0.288 \pm 0.010	8.588 \pm 0.100	13.484 \pm 0.236	5205 \pm 189	46344 \pm 2688
1.0	0.624 \pm 0.039	43.72 \pm 2.475	0.662 \pm 0.019	0.05 \pm 0.007	0.3 \pm 0.038	8.624 \pm 0.277	13.360 \pm 0.537	6337 \pm 921	49441 \pm 6520

Table C.4: **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, discount factor ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table measures performance differences for the discount factor. We note that most NLP approaches using RL follow the style of [240, 259] and use a discount factor of 1. This is equivalent to reducing the generation MDP to a bandit feedback environment and causes performance loss (in the case of NLPO) and reward hacking and training instability (in the case of PPO).

Hyperparams	Semantic and Fluency Metrics		Diversity Metrics						
	Sentiment Score \uparrow	Perplexity \downarrow	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂
Target Update Iterations μ									
1	0.594 \pm 0.018	32.671 \pm 0.201	0.669 \pm 0.008	0.042 \pm 0.002	0.284 \pm 0.007	8.575 \pm 0.064	13.503 \pm 0.181	4986 \pm 265	45916 \pm 1168
10	0.622 \pm 0.014	32.729 \pm 0.567	0.659 \pm 0.019	0.042 \pm 0.002	0.274 \pm 0.007	8.489 \pm 0.106	13.31 \pm 0.272	5138 \pm 385	43989 \pm 1120
20	0.637 \pm 0.013	32.667 \pm 0.631	0.677 \pm 0.014	0.044 \pm 0.002	0.288 \pm 0.010	8.588 \pm 0.100	13.484 \pm 0.236	5205 \pm 189	46344 \pm 2688
50	0.603 \pm 0.015	33.397 \pm 0.325	0.67 \pm 0.006	0.043 \pm 0.001	0.287 \pm 0.004	8.605 \pm 0.041	13.54 \pm 0.116	5228 \pm 113	46418 \pm 685
Top-p mask									
0.1	0.579 \pm 0.021	32.451 \pm 0.243	0.67 \pm 0.008	0.042 \pm 0.001	0.283 \pm 0.01	8.569 \pm 0.084	13.515 \pm 0.195	5018 \pm 47	45760 \pm 1579
0.3	0.588 \pm 0.019	32.451 \pm 0.303	0.666 \pm 0.007	0.043 \pm 0.001	0.285 \pm 0.004	8.568 \pm 0.032	13.482 \pm 0.172	5201 \pm 247	46357 \pm 539
0.5	0.588 \pm 0.01	32.447 \pm 0.393	0.669 \pm 0.001	0.044 \pm 0.003	0.291 \pm 0.008	8.614 \pm 0.053	13.535 \pm 0.06	5305 \pm 384	47251 \pm 1226
0.7	0.619 \pm 0.013	32.373 \pm 0.329	0.663 \pm 0.008	0.043 \pm 0.001	0.28 \pm 0.006	8.533 \pm 0.043	13.366 \pm 0.129	5186 \pm 216	45149 \pm 1452
0.9	0.637 \pm 0.013	32.667 \pm 0.631	0.677 \pm 0.014	0.044 \pm 0.002	0.288 \pm 0.010	8.588 \pm 0.100	13.484 \pm 0.236	5205 \pm 189	46344 \pm 2688

Table C.5: **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, NLPO hyperparameter ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table shows results of NLPO’s stability to the unique hyperparameters introduced in the algorithm - all other parameters held constant from the best PPO model. The number of iterations after which the masking model syncs with the policy and the top-p nucleus percentage for the mask model itself. We see that in general, the higher the top-p mask percentage, the better the performance. For target update iterations, performance is low if the mask model is not updated often enough or if it updated too often.

Algorithm	Unique N	Coherence			Sentiment		
		Value	Alpha	Skew	Value	Alpha	Skew
NLPO with KL	27	3.49	0.196	3.497	3.61	0.2	3.601
NLPO without KL	29	3.16	0.21	3.158	4.41	0.158	4.403
PPO without KL	27	3.16	0.17	3.163	4.36	0.196	4.363
PPO with KL	29	3.46	0.124	3.462	3.58	0.116	3.575
Zero Shot	28	3.6	0.162	3.591	3.1	0.13	3.097
Supervised	29	3.51	0.192	3.512	3.43	0.2	3.428
Human	27	4.13	0.159	4.128	3.01	0.31	3.017
Supervised+PPO	22	3.45	0.211	3.147	3.64	0.21	3.161
Supervised+NLPO	22	3.48	0.181	3.226	3.73	0.22	3.047

Table C.6: **IMDB Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorff’s alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 300 data points per algorithm.

Instructions (click to expand)

In this HIT you will be presented with a partial movie review that acts as a prompt and a system's automatically-generated continuation of that excerpt. Your job is to rate the the system generation across 2 axes:

- **Coherence/Quality:** *Is the system's generation grammatical, easy-to-read and does it follow from the prompt?*
- **Sentiment:** *Just considering the completion, how positive is it?*

You will be able to rate each of the three axes on a scale from 1 to 5, with 1 being the lowest/worst and 5 the highest/best. The specific scales are:

- **Coherence/Quality:**
 - 5/5 (excellent): The completion follows effortlessly from the prompt, and is grammatical, fluent, and reasonable.
 - 4/5 (good): The completion makes sense given the input, but there are minor grammatical errors or topical shifts that don't make for the best writing.
 - 3/5 (okay): I can see why this continued from the input, and it's readable, but there are problems that can't be ignored.
 - 2/5 (poor): Some parts of the completion might make sense given the input, but it's unnatural, illogical, or quite hard to read.
 - 1/5 (terrible): The completion completely ignores or contradicts the input, and/or there are severe errors in grammaticality or fluency.
- **Sentiment**
 - 5/5 (very positive): The completion is glowingly positive.
 - 4/5 (mostly positive): The completion is mostly positive, but there are some imperfections mentioned.
 - 3/5 (neutral): The completion either doesn't represent a positive/negative opinion, or it contains both very positive and very negative aspects.
 - 2/5 (mostly negative): Most of what's expressed here is very negative.
 - 1/5 (scathingly negative): The completion offers a strongly negative opinion.

Note: for rating sentiment, only consider the completion, and not the prompt itself!

Examples (click to expand)

Examples (click to expand)

Example 1:
Prompt:

I cannot BELIEVE anyone is giving this film a good rating. In addition to the terrible acting, thin (nonexistent?) plot line and sloooooooooooow pace, this would be the movie to watch if you were really TRYING to fall asleep. The writer's and director's brains must have been fried eggs to ever have concocted something as abominable as this. Based on the

System's generation (rate this!):

... director's prior work, however, I have hope for her future films. I think this small misstep is the result of a producer hampering her excellent creativity. There are some shining moments that show her expertise, and I look forward to the director's future films!

- **Coherence/Quality: 4/5** *Why?* The completion recognizes that this movie isn't good (as described in the prompt), and makes a reasonable pivot towards talking about the director's other works. The shift in sentiment is somewhat abrupt, but is well-explained.
- **Sentiment: 4/5** *Why?* The completion recognizes some shortcomings as a pivot, but also, provides a hopeful message for the director's future work.

Example 2:
Prompt:

I cannot BELIEVE anyone is giving this film a good rating. In addition to the terrible acting, thin (nonexistent?) plot line and sloooooooooooow pace, this would be the movie to watch if you were really TRYING to fall asleep. The writer's and director's brains must have been fried eggs to ever have concocted something as abominable as this. Based on the

System's generation (rate this!):


... amazing filmmaking, the film is a must-see for anyone who loves movies. A masterpiece of cinema, great for all ages. I highly recommend this film to anyone. 10/10.

- **Coherence/Quality: 1/5** *Why?* The very positive completion doesn't make any sense given the very negative discussion in the prompt. It contradicts the prompt entirely.
- **Sentiment: 5/5** *Why?* The completion, in isolation, offers only very positive opinions.

Coherence/Quality: 3/5

Is the system's generation grammatical, easy-to-read and does it follow from the prompt?

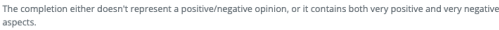
Bad
Excellent



Sentiment: 3/5

Just considering the system's generation, how positive is it?

Very Negative
Very Positive



(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Figure C.3: **IMDB Human Study Setup:** Instructions, example, and interface for the IMDB sentiment completion task.

Group 1	Group 2	Coherence		Sentiment	
		Diff (G2-G1)	<i>p-values</i>	Diff (G2-G1)	<i>p-values</i>
PPO with KL	PPO without KL	-0.3	0.035	0.783	0.001
PPO with KL	NLPO with KL	0.03	0.9	0.027	0.9
PPO with KL	NLPO without KL	-0.3	0.035	0.827	0.001
PPO with KL	Supervised	0.05	0.9	-0.15	0.591
PPO with KL	Human	0.667	0.001	-0.567	0.001
PPO with KL	Zero Shot	0.137	0.776	-0.483	0.001
PPO without KL	NLPO with KL	0.33	0.013	-0.757	0.001
PPO without KL	NLPO without KL	0.001	0.9	0.043	0.9
PPO without KL	Supervised	0.35	0.006	-0.933	0.001
PPO without KL	Human	0.967	0.009	-1.35	0.001
PPO without KL	Zero Shot	0.437	0.001	-1.267	0.001
NLPO with KL	NLPO without KL	-0.33	0.013	0.8	0.001
NLPO with KL	Supervised	0.02	0.9	-0.177	0.404
NLPO with KL	Human	0.637	0.001	-0.593	0.001
NLPO with KL	Zero Shot	0.107	0.9	-0.51	0.001
NLPO without KL	Supervised	0.35	0.006	-0.977	0.001
NLPO without KL	Human	0.967	0.001	-1.393	0.001
NLPO without KL	Zero Shot	0.437	0.001	-1.31	0.001
Supervised	Human	0.617	0.001	-0.417	0.001
Supervised	Zero Shot	0.087	0.9	-0.333	0.0027
Human	Zero Shot	-0.53	0.001	0.083	0.9
Supervised+PPO	Supervised+NLPO	0.03	0.9	0.09	0.035
Supervised+PPO	NLPO with KL	0.04	0.9	-0.03	0.9
Supervised+PPO	NLPO without KL	-0.29	0.001	0.77	0.001
Supervised+PPO	PPO without KL	-0.29	0.006	0.72	0.001
Supervised+PPO	PPO with KL	0.01	0.9	-0.06	0.001
Supervised+PPO	Zero Shot	0.15	0.035	-0.54	0.001
Supervised+PPO	Supervised	0.06	0.001	-0.21	0.001
Supervised+PPO	Human	0.68	0.001	-0.63	0.001
Supervised+NLPO	NLPO with KL	0.01	0.9	-0.12	0.001
Supervised+NLPO	NLPO without KL	-0.32	0.001	0.68	0.001
Supervised+NLPO	PPO without KL	-0.32	0.035	0.63	0.001
Supervised+NLPO	PPO with KL	-0.02	0.9	-0.15	0.006
Supervised+NLPO	Zero Shot	-0.12	0.001	-0.63	0.001
Supervised+NLPO	Supervised	0.03	0.9	-0.3	0.001
Supervised+NLPO	Human	0.65	0.001	-0.72	0.006

Table C.7: **IMDB Human Study Tukey Results**: Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding *p*-values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall *p*-values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.

Qualitative Results

We show sample generations from each of the algorithms for three randomly picked prompts below.

Sample 1

Prompt: A man and his wife are not getting along because of the wife's jealousy and drinking problems. When the wife disappears, her sister begins a vigorous search involving the

Zero-Shot: husband's sister, which leads several men to a remote cabin and then a boat. The wife and her husband then start to see the same things. Finally, a strange woman is driving out to her cabin and discovers the husband's disappearance but

References: local police, state government, and a television crime show. What she finds out stuns the entire community. Good drama with lots of fine performances; based on a true story

PPO: police and the local public. At the end, the woman is killed and the wife is found naked and with a red head tattooed on her back. This is part of a special feature called "On the Road, Hidden" which was

PPO-no-KL: police and other local governments. Written by G. Scott Fitzgerald, who also stars in this production, "A Very Merry Christmas" is a wonderful family movie with a wonderful soundtrack and wonderful characters. It brings home the true beauty of small-

NLPO: mother while the brother goes under. After all, his sister, who was apparently kidnapped but who turns out to be a man, is no longer in her place. If a film that was made in the 1950s

NLPO-no-KL: friends of the wife, and also the friend of the wife of the woman that is missing, and eventually he begins to find that she has gone to Germany and he has made a very good friend. There is a good scene between John and

Supervised: family tree. The widower, now widowed, returns, the daughter, daughter is not pleased when his wife is mistaken for one of her sons, and the patriarch, then a wealthy man, falls victim to a jealous husband. The

Sample 2

Prompt: A very credible and unsettling movie portraying the effects of war on the general public, Shame gives us a chilling understanding on Ingmar Bergman's beliefs. The absence of color and sound (as in a soundtrack) helps to give the film a more realistic feeling. There is no soft or hard lighting or a variety of camera angles to give the story a charming or dramatic effect. Everything is very simply portrayed as it would be in reality. At times the storyline was confusing, but perhaps in order to give a sense of the characters' frame of mind how they too do not know what is happening or why. The simplicity of the

Zero-Shot: camera work is at once apparent and fascinating in the manner in which it affects the way things are portrayed. In other films these two have been referred to from time to time in a form and manner more suited to the screen. The film shows

References: film made it somewhat boring, but I could understand how the director is trying not to glamorize the story and points. He depicted the harsh and unsettling events and effects of war. In the beginning the characters were introduced and shown as content with the lighter settings

PPO: characters relationships is so engrossing and the great and unique story presented by each pair of brothers and sisters creates a truly gripping and enjoyable experience. To put it simply, I strongly recommend to everyone who plays the role to buy this DVD!!!

PPO-no-KL: movie helps it to flow along in a believable way. Rating 4 out of 5 Great Movie! I highly recommend these movies! I highly recommend these movies! Rating 5 out of 5 Great Movie! I highly recommend these movies! Rating 10 of"

NLPO: novel makes it one of the first books I have read by a British actor. It is a fun way for someone who likes their books, like me, to sit through a novel. If you like to read novels, check it out.

NLPO-no-KL: setting makes the final product very enjoyable to watch. Highly recommended!! ~ Michael Bowerman, SPSA Film Festival Host, SPSA, WA (December 17, 2006), 8/17/2006 Great photography, good acting,

Supervised: "Dakota Clan" and his sister, the two become friends with Ted Masters who soon finds the chance to fly back to their home land and go back to save his brother's life and his family. The

Sample 3

Prompt: Shakespearean language is wild and rambling, saturated in multiple meanings, word play and metaphor. To be understood it must be wrangled and tamed by an actor with the strength

and knowledge to do so. When an actor fails, the words pour forth in a torrent of incomprehensible words, but when he succeeds, the English language springs to life with an immediacy and vibrancy that takes your breath away. Al Pacino is one such actor, and here displays an incredible level of clarity and control that, were there any justice,

Zero-Shot: might still be seen in the world today. This movie, like "Vacation", is an effort at telling the tale of two families from different races. One is African but never grew up the other. One of them, a man

References: would sweep every award in the offering. He meets the challenge of presenting Shylock head on, and delivers an extraordinarily subtle and nuanced performance. It would be a crime if we never got the opportunity to see what he does with King Lear.

The supporting cast

PPO: would have made him an instant-classic. This movie captures the spirit of childhood and has everyone in the world watching it. It's a love story and it's really heartbreaking. And at the same time, it's a classic. It

PPO-no-KL: will bring victory to the fight against hatred and extremism. Al Pacino is a very beautiful, well executed actor who brings an absolute intensity and sensibility to his portrayal of love and loyalty. A must see! Recommended! Also, an Excellent

NLPO: I would watch and enjoy. 7/10, Oscar Wilde is always a pleasure to see. A must see. Highly recommend. Highly recommend! One of my top ten films of all time. A must see!!

NLPO-no-KL: the whole film would have fallen to the same fate, just as it did just a couple hundred years ago. Don't miss it. It's a real classic. Highly Recommended. * outta five stars for it!

Supervised: his performance (so far) would seem mere shadow. He is truly in the middle of a movie , and this film is one of those films where he can be convincing in it (and his trademark acting, as you can see in the

C.1.4 CommonGen

Setup CommonGen [293] deals with task of generating coherent sentences describing an input set of concepts (eg. "a man is throwing a frisbee"). For training RL methods, we consider 3 traditional lexical rewards namely Rouge-1, Rouge-avg (which is an average of Rouge-1, 2 and L) and meteor. Additionally, we also train with task-specific rewards such as CIDEr [281], SPICE [282] and SPiDer [308] which is a just a linear combination of both with equal weights. We chose T5-base as the base LM since it is well-suited for structure to text tasks. We additionally note that concept set inputs are prefixed with "generate a sentence with:" to encourage exploration.

During our initial experiments when fine-tuning directly on LM, we observed that policy learns to repeat the prompted concepts in order to maximize rewards resulting in a well-known problem of *reward hacking*. To mitigate this, we add a penalty score of -1 to final task reward if the n-grams of prompt text overlaps with generated text. In contrast, when initialized with a supervised policy, this problem is not seen and hence penalty score is not applied. We use beam search as the decoding method during evaluation whereas for rollouts, we use top k sampling to favor exploration over exploitation. Table C.8 provides an in-depth summary of setting of hyperparameter values along with other implementation details.

Results and Discussion Tables C.10, C.9 presents our benchmarking results with 6 reward functions along with supervised baseline performances on dev and test sets respectively. Our main finding is that warm-started initial policies are crucial for learning to generate coherent sentences with common sense. Without warm-start, policies suffer from reward hacking despite application of repetition penalty and task-specific metrics such as CIDEr etc. Further, we find that RL fine-tuned models obtain very high concept coverage which is also seen in Table C.1.4. Supervised models often tend to miss few concepts in its generation compared to RL methods.

Model Params	value
supervised	batch size: 8 epochs: 4 learning rate: 0.00001 learning rate scheduler: cosine weight decay: 0.01
ppo/ nlpo	steps per update: 1280 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.01 initial kl coeff: 0.001 target kl: 2.0 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
supervised+ ppo (or nlpo)	steps per update: 1280 total number of steps: 128000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.01 initial kl coeff: 0.01 target kl: 1.0 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
decoding	num beams: 5 min length: 5 max new tokens: 20
tokenizer	padding side: left max length: 20

Table C.8: **CommonGen Hyperparams**: Table shows a list of all hyper-parameters and their settings

Appendix C RL4LMs

Tasks	Alg	LM	Reward function	Lexical and Semantic Metrics							
				Rouge-2	Rouge-L	Bleu (n=3)	Bleu (n=4)	Meteor	CIDEr	SPICE	Coverage
Zero-Shot	T5			0.016	0.264	0.029	0.006	0.203	6.200	0.115	91.070
CommonGen	PPO	Rouge-1	T5	0.085 ± 0.008	0.354 ± 0.004	0.161 ± 0.011	0.087 ± 0.009	0.235 ± 0.002	8.673 ± 0.234	0.157 ± 0.001	88.544 ± 2.36
		Rouge-Avg	T5	0.093 ± 0.005	0.351 ± 0.001	0.169 ± 0.032	0.097 ± 0.017	0.224 ± 0.012	8.212 ± 1.329	0.159 ± 0.011	82.584 ± 2.569
		Meteor	T5	0.091 ± 0.008	0.308 ± 0.007	0.166 ± 0.016	0.088 ± 0.013	0.220 ± 0.006	7.251 ± 0.453	0.161 ± 0.007	79.718 ± 2.267
		SPice	T5	0.065 ± 0.003	0.302 ± 0.002	0.115 ± 0.063	0.067 ± 0.041	0.193 ± 0.014	6.571 ± 1.312	0.175 ± 0.011	69.340 ± 3.617
		SPider	T5	0.066 ± 0.003	0.304 ± 0.002	0.132 ± 0.057	0.074 ± 0.036	0.211 ± 0.009	6.877 ± 1.218	0.143 ± 0.017	80.114 ± 4.852
		SPider	T5	0.117 ± 0.005	0.352 ± 0.007	0.224 ± 0.014	0.137 ± 0.011	0.226 ± 0.01	9.162 ± 0.539	0.186 ± 0.006	73.374 ± 6.073
	NLPO	Rouge-1	T5	0.087 ± 0.002	0.339 ± 0.009	0.127 ± 0.048	0.069 ± 0.035	0.213 ± 0.002	6.962 ± 0.883	0.145 ± 0.022	80.89 ± 9.544
		Rouge-Avg	T5	0.095 ± 0.001	0.338 ± 0.002	0.159 ± 0.02	0.093 ± 0.013	0.216 ± 0.009	7.55 ± 0.688	0.153 ± 0.008	77.944 ± 2.770
		Meteor	T5	0.110 ± 0.005	0.332 ± 0.003	0.214 ± 0.007	0.124 ± 0.007	0.235 ± 0.004	8.669 ± 0.164	0.173 ± 0.002	82.007 ± 1.012
		SPice	T5	0.014 ± 0.006	0.242 ± 0.001	0.037 ± 0.011	0.018 ± 0.007	0.156 ± 0.007	4.685 ± 0.283	0.168 ± 0.008	56.998 ± 3.548
		SPider	T5	0.046 ± 0.001	0.241 ± 0.003	0.078 ± 0.028	0.043 ± 0.016	0.143 ± 0.018	3.964 ± 0.792	0.103 ± 0.012	49.606 ± 7.971
		SPider	T5	0.060 ± 0.006	0.258 ± 0.001	0.090 ± 0.008	0.056 ± 0.005	0.151 ± 0.022	4.411 ± 0.837	0.123 ± 0.022	49.230 ± 10.468
	Supervised	T5		0.215 ± 0.001	0.438 ± 0.001	0.444 ± 0.001	0.329 ± 0.001	0.321 ± 0.001	16.385 ± 0.046	0.299 ± 0.001	94.476 ± 0.172
	Supervised + PPO	Rouge-1	T5	0.232 ± 0.002	0.453 ± 0.002	0.454 ± 0.006	0.338 ± 0.006	0.320 ± 0.002	16.233 ± 0.159	0.288 ± 0.004	96.412 ± 0.424
		Rouge-Avg	T5	0.230 ± 0.001	0.450 ± 0.001	0.448 ± 0.005	0.334 ± 0.005	0.319 ± 0.001	16.069 ± 0.167	0.287 ± 0.003	96.116 ± 0.679
		Meteor	T5	0.234 ± 0.002	0.450 ± 0.003	0.462 ± 0.007	0.342 ± 0.007	0.327 ± 0.001	16.797 ± 0.152	0.295 ± 0.001	97.690 ± 0.371
		SPice	T5	0.227 ± 0.004	0.447 ± 0.003	0.450 ± 0.007	0.336 ± 0.008	0.319 ± 0.002	16.208 ± 0.249	0.288 ± 0.003	96.492 ± 0.29
		SPider	T5	0.224 ± 0.003	0.446 ± 0.003	0.427 ± 0.012	0.309 ± 0.01	0.316 ± 0.004	15.497 ± 0.428	0.283 ± 0.005	96.345 ± 0.547
		SPider	T5	0.226 ± 0.003	0.448 ± 0.002	0.436 ± 0.005	0.319 ± 0.004	0.317 ± 0.003	15.678 ± 0.192	0.281 ± 0.003	96.154 ± 0.426
	Supervised + NLPO	Rouge-1	T5	0.229 ± 0.002	0.450 ± 0.001	0.454 ± 0.005	0.338 ± 0.004	0.320 ± 0.003	16.206 ± 0.175	0.289 ± 0.002	96.342 ± 0.572
		Rouge-Avg	T5	0.232 ± 0.003	0.451 ± 0.002	0.458 ± 0.01	0.342 ± 0.009	0.321 ± 0.003	16.351 ± 0.335	0.290 ± 0.005	95.998 ± 0.496
		Meteor	T5	0.231 ± 0.003	0.449 ± 0.002	0.454 ± 0.007	0.334 ± 0.008	0.326 ± 0.002	16.574 ± 0.269	0.292 ± 0.003	97.374 ± 0.457
		SPice	T5	0.223 ± 0.002	0.442 ± 0.001	0.435 ± 0.011	0.321 ± 0.010	0.315 ± 0.004	15.747 ± 0.401	0.283 ± 0.005	96.25 ± 0.313
		SPider	T5	0.226 ± 0.002	0.447 ± 0.004	0.433 ± 0.007	0.315 ± 0.008	0.318 ± 0.003	15.741 ± 0.170	0.285 ± 0.001	96.354 ± 0.971
		SPider	T5	0.226 ± 0.004	0.447 ± 0.003	0.434 ± 0.006	0.316 ± 0.006	0.319 ± 0.002	15.739 ± 0.311	0.284 ± 0.003	96.333 ± 0.644

Table C.9: **CommonGen test evaluation** Table shows official scores obtained from CommonGen hold-out evaluation. The most important result is that RL fine-tuning on a supervised model yields better performance across most metrics especially Coverage which indicates the ratio of concepts covered in generated texts

Tasks	Alg	Reward Function	Top k	LM	Lexical and Semantic Metrics								
					Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BLEU	BertScore	Cider	Spice
Zero-Shot	T5				0.415	0.016	0.270	0.270	0.179	0.0	0.854	0.640	0.231
CommonGen	PPO	Rouge-1	50	T5	0.537 ± 0.004	0.093 ± 0.012	0.380 ± 0.006	0.380 ± 0.006	0.235 ± 0.005	0.016 ± 0.002	0.896 ± 0.001	0.950 ± 0.015	0.318 ± 0.016
		Rouge-Avg	50	T5	0.519 ± 0.0185	0.102 ± 0.007	0.377 ± 0.013	0.376 ± 0.014	0.225 ± 0.024	0.020 ± 0.002	0.897 ± 0.005	0.921 ± 0.102	0.328 ± 0.009
		Meteor	50	T5	0.411 ± 0.009	0.090 ± 0.008	0.304 ± 0.006	0.304 ± 0.006	0.210 ± 0.005	0.029 ± 0.004	0.875 ± 0.007	0.638 ± 0.048	0.259 ± 0.017
		SPice	50	T5	0.439 ± 0.035	0.079 ± 0.045	0.323 ± 0.036	0.323 ± 0.036	0.183 ± 0.022	0.012 ± 0.009	0.891 ± 0.005	0.777 ± 0.140	0.400 ± 0.012
		SPider	50	T5	0.453 ± 0.038	0.081 ± 0.037	0.326 ± 0.033	0.326 ± 0.033	0.203 ± 0.022	0.017 ± 0.009	0.885 ± 0.008	0.770 ± 0.134	0.291 ± 0.036
		SPider	50	T5	0.512 ± 0.008	0.141 ± 0.007	0.388 ± 0.002	0.388 ± 0.003	0.242 ± 0.007	0.032 ± 0.003	0.902 ± 0.001	1.045 ± 0.034	0.380 ± 0.006
	NLPO	Rouge-1	50	T5	0.499 ± 0.012	0.089 ± 0.003	0.328 ± 0.007	0.328 ± 0.007	0.198 ± 0.002	0.021 ± 0.001	0.872 ± 0.005	0.815 ± 0.009	0.305 ± 0.008
		Rouge-Avg	50	T5	0.47 ± 0.01	0.096 ± 0.004	0.312 ± 0.006	0.312 ± 0.006	0.202 ± 0.008	0.025 ± 0.002	0.843 ± 0.013	0.816 ± 0.026	0.299 ± 0.007
		Meteor	50	T5	0.389 ± 0.013	0.1 ± 0.004	0.293 ± 0.008	0.293 ± 0.008	0.226 ± 0.024	0.035 ± 0.004	0.832 ± 0.018	0.691 ± 0.04	0.266 ± 0.016
		SPice	50	T5	0.329 ± 0.015	0.036 ± 0.008	0.247 ± 0.013	0.247 ± 0.013	0.137 ± 0.009	0.006 ± 0.002	0.817 ± 0.024	0.515 ± 0.033	0.323 ± 0.021
		SPider	50	T5	0.515 ± 0.006	0.143 ± 0.008	0.387 ± 0.006	0.308 ± 0.006	0.19 ± 0.001	0.019 ± 0.001	0.865 ± 0.015	0.726 ± 0.018	0.282 ± 0.009
		SPider	50	T5	0.393 ± 0.008	0.086 ± 0.012	0.297 ± 0.007	0.297 ± 0.007	0.183 ± 0.007	0.02 ± 0.003	0.842 ± 0.019	0.717 ± 0.026	0.297 ± 0.019
Supervised	T5		0.503 ± 0.001	0.175 ± 0.001	0.411 ± 0.001	0.411 ± 0.001	0.309 ± 0.001	0.069 ± 0.001	0.929 ± 0.000	1.381 ± 0.011	0.443 ± 0.001		
Supervised + PPO	Rouge-1	50	T5	0.537 ± 0.004	0.198 ± 0.005	0.433 ± 0.002	0.433 ± 0.002	0.314 ± 0.003	0.070 ± 0.002	0.930 ± 0.001	1.426 ± 0.018	0.449 ± 0.001	
	Rouge-Avg	50	T5	0.536 ± 0.001	0.198 ± 0.002	0.433 ± 0.002	0.433 ± 0.002	0.311 ± 0.002	0.070 ± 0.002	0.929 ± 0.001	1.421 ± 0.028	0.446 ± 0.004	
	Meteor	50	T5	0.540 ± 0.005	0.204 ± 0.005	0.436 ± 0.004	0.436 ± 0.004	0.329 ± 0.003	0.076 ± 0.003	0.930 ± 0.001	1.474 ± 0.022	0.447 ± 0.004	
	SPice	50	T5	0.532 ± 0.006	0.194 ± 0.007	0.430 ± 0.005	0.430 ± 0.005	0.311 ± 0.004	0.068 ± 0.003	0.929 ± 0.001	1.415 ± 0.029	0.458 ± 0.001	
	SPider	50	T5	0.530 ± 0.004	0.191 ± 0.003	0.427 ± 0.004	0.427 ± 0.004	0.309 ± 0.008	0.063 ± 0.002	0.928 ± 0.001	1.337 ± 0.040	0.444 ± 0.002	
	SPider	50	T5	0.536 ± 0.002	0.197 ± 0.002	0.430 ± 0.002	0.430 ± 0.002	0.313 ± 0.002	0.064 ± 0.002	0.928 ± 0.001	1.374 ± 0.018	0.445 ± 0.003	
Supervised + NLPO	Rouge-1	50	T5	0.545 ± 0.002	0.197 ± 0.002	0.432 ± 0.001	0.432 ± 0.001	0.31 ± 0.002	0.068 ± 0.001	0.929 ± 0.0	1.41 ± 0.012	0.449 ± 0.001	
	Rouge-Avg	50	T5	0.541 ± 0.003	0.2 ± 0.003	0.435 ± 0.002	0.435 ± 0.002	0.313 ± 0.002	0.07 ± 0.002	0.93 ± 0.001	1.424 ± 0.023	0.447 ± 0.003	
	Meteor	50	T5	0.537 ± 0.003	0.201 ± 0.004	0.431 ± 0.002	0.431 ± 0.002	0.326 ± 0.002	0.074 ± 0.003	0.93 ± 0.0	1.464 ± 0.025	0.448 ± 0.002	
	SPice	50	T5	0.535 ± 0.007	0.193 ± 0.008	0.429 ± 0.005	0.429 ± 0.005	0.3 ± 0.003	0.064 ± 0.002	0.927 ± 0.001	1.333 ± 0.017	0.459 ± 0.003	
	SPider	50	T5	0.533 ± 0.003	0.197 ± 0.004	0.43 ± 0.003	0.43 ± 0.004	0.316 ± 0.004	0.066 ± 0.001	0.929 ± 0.001	1.381 ± 0.014	0.446 ± 0.004	
	SPider	50	T5	0.532 ± 0.006	0.196 ± 0.006	0.431 ± 0.004	0.431 ± 0.004	0.314 ± 0.004	0.066 ± 0.002	0.929 ± 0.0	1.371 ± 0.011	0.448 ± 0.002	

Table C.10: **CommonGen - Lexical and Semantic Metrics on dev set:** Table shows lexical and semantic for best performing models found in each algorithm-reward function combinations along with best performing supervised baseline models. Generated text from these models are submitted to official CommonGen test evaluation to obtain test scores presented in Table C.9

C.1 Experimental Details

Tasks	Alg	Reward Function	Top k	LM	MSTTR	Distinct ₁	Distinct ₂	Diversity Metrics				Mean Output Length
								H ₁	H ₂	Unique ₁	Unique ₂	
CommonGen	Zero-Shot			T5	0.430	0.090	0.335	5.998	7.957	345	1964	8.797
	PPO	Rouge-1	50	T5	0.526 ± 0.020	0.128 ± 0.005	0.518 ± 0.036	6.679 ± 0.132	10.572 ± 0.234	437.4 ± 42.017	2418.8 ± 167.947	7.214 ± 0.374
		Rouge-Avg	50	T5	0.536 ± 0.069	0.141 ± 0.022	0.510 ± 0.056	6.777 ± 0.539	10.348 ± 0.134	458.6 ± 19.734	2244.4 ± 162.855	6.887 ± 1.006
		Meteor	50	T5	0.547 ± 0.012	0.147 ± 0.003	0.529 ± 0.014	7.62 ± 0.127	11.464 ± 0.151	1039.4 ± 63.276	5197.2 ± 280.004	13.660 ± 0.324
		SPice	50	T5	0.546 ± 0.054	0.149 ± 0.019	0.545 ± 0.072	6.721 ± 0.441	10.492 ± 0.330	409.2 ± 41.605	1878.4 ± 167.492	5.706 ± 0.678
		CiDer	50	T5	0.597 ± 0.081	0.195 ± 0.040	0.639 ± 0.106	7.732 ± 0.682	11.131 ± 0.502	777.0 ± 144.676	3350.8 ± 503.419	7.393 ± 0.572
		SPiDer	50	T5	0.482 ± 0.015	0.133 ± 0.003	0.472 ± 0.021	6.372 ± 0.221	10.303 ± 0.228	502.6 ± 33.422	2281.4 ± 252.471	7.489 ± 0.358
	NLPO	Rouge-1	50	T5	0.559 ± 0.01	0.148 ± 0.003	0.555 ± 0.012	7.059 ± 0.067	10.657 ± 0.105	457.9 ± 11.108	2349.6 ± 60.345	6.586 ± 0.094
		Rouge-Avg	50	T5	0.512 ± 0.019	0.146 ± 0.011	0.513 ± 0.012	6.781 ± 0.15	10.424 ± 0.156	484.18 ± 17.303	2357.54 ± 152.113	7.131 ± 0.487
		Meteor	50	T5	0.503 ± 0.003	0.132 ± 0.005	0.471 ± 0.008	7.146 ± 0.192	10.727 ± 0.313	648.05 ± 33.963	3536.0 ± 444.638	11.062 ± 1.301
		SPice	50	T5	0.543 ± 0.023	0.174 ± 0.004	0.568 ± 0.026	7.176 ± 0.212	10.551 ± 0.216	479.45 ± 19.77	2065.8 ± 288.843	5.785 ± 0.431
		CiDer	50	T5	0.550 ± 0.02	0.179 ± 0.005	0.576 ± 0.014	7.286 ± 0.125	10.812 ± 0.089	661.46 ± 21.776	2726.32 ± 71.253	7.13 ± 0.223
		SPiDer	50	T5	0.525 ± 0.024	0.167 ± 0.009	0.537 ± 0.025	6.986 ± 0.262	10.451 ± 0.171	530.14 ± 16.805	2263.4 ± 166.221	6.687 ± 0.372
	Supervised			T5	0.509 ± 0.001	0.101 ± 0.001	0.339 ± 0.001	6.531 ± 0.006	10.079 ± 0.016	503.600 ± 6.530	2158.8 ± 24.514	10.934 ± 0.020
	Supervised + PPO	Rouge-1	50	T5	0.527 ± 0.007	0.112 ± 0.001	0.393 ± 0.004	6.680 ± 0.044	10.289 ± 0.040	498.2 ± 8.931	2317.0 ± 22.609	9.667 ± 0.105
		Rouge-Avg	50	T5	0.526 ± 0.004	0.114 ± 0.002	0.395 ± 0.005	6.682 ± 0.0297	10.274 ± 0.042	506.4 ± 6.829	2326.4 ± 41.778	9.614 ± 0.102
		Meteor	50	T5	0.514 ± 0.004	0.105 ± 0.002	0.378 ± 0.008	6.631 ± 0.053	10.270 ± 0.064	507.0 ± 17.146	2424.6 ± 72.550	10.551 ± 0.271
		SPice	50	T5	0.532 ± 0.008	0.113 ± 0.0038	0.392 ± 0.009	6.736 ± 0.058	10.338 ± 0.057	507.4 ± 14.319	2313.8 ± 27.694	9.742 ± 0.208
		CiDer	50	T5	0.518 ± 0.009	0.110 ± 0.003	0.382 ± 0.006	6.614 ± 0.082	10.166 ± 0.053	490.4 ± 9.457	2295.4 ± 51.554	9.838 ± 0.265
		SpiDer	50	T5	0.524 ± 0.007	0.112 ± 0.001	0.394 ± 0.004	6.673 ± 0.066	10.247 ± 0.066	504.8 ± 7.440	2361.8 ± 20.856	9.761 ± 0.121
	Supervised + NLPO	Rouge-1	50	T5	0.529 ± 0.002	0.114 ± 0.002	0.399 ± 0.005	6.705 ± 0.018	10.301 ± 0.03	498.86 ± 8.594	2311.46 ± 33.451	9.463 ± 0.111
		Rouge-Avg	50	T5	0.530 ± 0.006	0.113 ± 0.002	0.396 ± 0.008	6.708 ± 0.05	10.318 ± 0.074	493.64 ± 10.068	2319.42 ± 55.738	9.596 ± 0.123
		Meteor	50	T5	0.516 ± 0.006	0.106 ± 0.002	0.377 ± 0.008	6.634 ± 0.044	10.26 ± 0.077	506.04 ± 3.502	2401.32 ± 38.569	10.453 ± 0.194
		SPice	50	T5	0.553 ± 0.013	0.12 ± 0.004	0.415 ± 0.014	6.908 ± 0.118	10.445 ± 0.057	508.075 ± 4.669	2343.3 ± 53.274	9.249 ± 0.225
CiDer		50	T5	0.516 ± 0.009	0.108 ± 0.003	0.379 ± 0.01	6.583 ± 0.077	10.165 ± 0.084	490.78 ± 9.734	2304.52 ± 62.068	9.923 ± 0.213	
SPiDer		50	T5	0.521 ± 0.005	0.109 ± 0.002	0.385 ± 0.005	6.623 ± 0.034	10.223 ± 0.049	485.325 ± 5.683	2297.575 ± 21.271	9.798 ± 0.179	

Table C.11: **CommonGen - Diversity Metrics on dev set:** Table shows diversity metrics for best performing models found in each algorithm-reward function combinations along with best performing supervised baseline models. Generated text from these models are submitted to official CommonGen test evaluation to obtain test scores presented in Table C.9

Algorithm	Unique N	Coherence			Commonsense		
		Value	Alpha	Skew	Value	Alpha	Skew
PPO+Supervised	25	4.14	0.073	4.137	4.03	0.137	4.023
NLPO+Supervised	26	4.25	0.036	4.253	4.16	0.002	4.163
Zero Shot	24	2.15	0.391	2.154	2.29	0.342	2.291
PPO	24	2.84	0.16	2.849	3.03	0.081	3.027
Supervised	23	4.39	0.159	4.387	4.21	0.225	4.209
NLPO	24	2	0.335	2.003	2.13	0.265	2.124

Table C.12: **CommonGen Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorff’s alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 300 data points per algorithm.

Group 1	Group 2	Coherence		Commonsense	
		Diff (G2-G1)	<i>p-values</i>	Diff (G2-G1)	<i>p-values</i>
NLPO	PPO	0.847	0.001	0.897	0.001
NLPO	Supervised	2.397	0.001	2.083	0.001
NLPO	NLPO+Supervised	2.257	0.001	2.033	0.001
NLPO	PPO+Supervised	2.143	0.001	1.897	0.001
NLPO	Zero Shot	0.153	0.515	0.157	0.624
PPO	Supervised	1.550	0.001	1.187	0.001
PPO	NLPO+Supervised	1.410	0.001	1.137	0.001
PPO	PPO+Supervised	1.297	0.001	1.000	0.001
PPO	Zero Shot	-0.693	0.001	-0.740	0.001
Supervised	NLPO+Supervised	-0.140	0.601	-0.050	0.900
Supervised	PPO+Supervised	-0.253	0.050	-0.187	0.045
Supervised	Zero Shot	-2.243	0.001	-1.927	0.001
NLPO+Supervised	PPO+Supervised	-0.113	0.008	-0.137	0.007
NLPO+Supervised	Zero Shot	-2.103	0.001	-1.877	0.001
PPO+Supervised	Zero Shot	-1.990	0.001	-1.740	0.001

Table C.13: **CommonGen Human Study Tukey Results:** Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding *p*-values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall *p*-values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.

Instructions (click to expand)

In this HIT you will be presented with a short system generation. Usually the generation will be only a single sentence. Your job is to rate the generation across 2 axes:

- **Fluency/Grammaticality:** *Is the system's generation grammatical, easy-to-read, and fluent?*
- **Commonsense:** *Is the system's generation describing a plausible, realistic, and commonsensical scenario?*

You will be able to rate each of the three axes on a scale from 1 to 5, with 1 being the lowest/worst and 5 the highest/best. The specific scales are:

- **Fluency/Grammaticality:**
 - 5/5 (excellent): The generation is grammatical and fluent.
 - 4/5 (good): The sentence largely makes sense, but there are some small grammar issues/out-of-place words that don't make for the best writing.
 - 3/5 (okay): The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.
 - 2/5 (poor): Even though I can kind-of tell the meaning, it's difficult to read this unnatural sentence.
 - 1/5 (terrible): The generation has severe errors in grammaticality/is almost or completely unreadable.
- **Commonsense**
 - 5/5 (this is reasonable+plausible!): This describes a very coherent/plausible/reasonable situation.
 - 4/5 (mostly reasonable): This could reasonably happen.
 - 3/5 (neutral): This situation might happen, but it's not that likely/it's a bit weird.
 - 2/5 (mostly unreasonable): Most of what's expressed here couldn't happen at all.
 - 1/5 (this wouldn't happen!): This is impossible/nonsensical.

Note: for rating fluency/grammaticality, don't worry about the commonsense axis! There can be grammatical sentences that are nonsensical, and vice versa (see the examples).

Examples (click to expand)

Example 1:
System's generation (rate this!):
The man wore a glove on his hand to open the oyster.

- **Fluency/Grammaticality: 5/5 Why?** The completion is grammatically correct and easy to read.
- **Commonsense: 5/5 Why?** It makes sense that one would wear a glove on their hand to open an oyster.

Example 2:
System's generation (rate this!):
The man wore an oyster on his glove to open his hand.

- **Fluency/Grammaticality: 5/5 Why?** This sentence, even though it doesn't make sense, is grammatically correct and easy to read.
- **Commonsense: 2/5 Why?** You could do what's described in theory, but it makes very close to no sense.

Example 3:
System's generation (rate this!):
Wear glove to open oyster with hand.

- **Fluency/Grammaticality: 3/5 Why?** It's possible to fill-in-the-gaps to make sense of things, but this is not a fluent sentence.
- **Commonsense: 4/5 Why?** The most reasonable reading of this sentence describes wearing a glove to open an oyster, which is reasonable, but it takes a bit of interpolation to get to that.

Coherence/Quality: 3/5
Is the system's generation grammatical, easy-to-read, and fluent?
Bad Excellent

The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.

Commonsense: 3/5
Is the system's generation describing a plausible, realistic, and commonsensical scenario?
this wouldn't happen! this is reasonable+plausible.

This situation might happen, but it's not that likely/it's a bit weird.

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Figure C.4: **CommonGen Human Study Setup:** Instructions, examples, and interface for the CommonGen task.

Human Participant Study

Figure C.4 shows the commongen instructions, examples, and interface used for the human evaluation experiments. Different from the other human evaluations, we didn't provide any prompt because knowing the set of words to be used isn't required for rating either of the axes. Tables C.12, C.13 show averaged results, annotator agreement, and the results of statistical significance tests to determine which models output better generations when rated by humans.

Human Preference Learning Experiments

First, we randomly select prompts from the CommonGen train dataset and sample a single completion from both the Supervised and Supervised+NLPO models. Next, we filter to prompts where both models at least attempted to use all input concepts. This filtration step was conducted because if a model fails to use all concepts, it may generate a more natural/fluent sentence, but, *a priori*, it shouldn't be preferred by crowdworkers; instead of training crowdworkers to prefer sentences with all concepts, we perform this filter. Figure C.5 shows the task presented to the crowdworkers. We then present the prompt and the two completion candidates to 3 unique crowdworkers and ask them to select which one they prefer with respect to commonsense/fluency; We gathered 3 annotations on 417 pairs (Krippendorff $\alpha = .28$), and split into 60/20/20 train/val/test split. We then trained a reward model, T5-11B [298], on the balanced binary classification task of predicting which of the pair was preferred by a majority of 3 annotators, conditioned on the prompt and completion. The resulting model achieved 69.5 test ROC AUC suggesting it indeed captures average human preferences. The model is then used as a reward function. We train Supervised+RL with a METEOR-only reward as a baseline, and compare it to a reward function that uses the fine-tuned T5-11B model. We design the reward function based on the preference model as $r = meteor + pref / (1 + |miss|)$ where *miss* is a set of concepts not covered in the generated text, in an attempt to mimic the data collection process that humans are instructed to follow. This reward function accounts for both the task of using all concepts and also human's preferences for how a sentence should look within the constraints stipulated by the task. Finally, we rerun the same pairwise preference collection procedure—this time sampling from CommonGen test—with human participants to compare the generations from a preference optimized RL policy to the previously best Supervised+NLPO policy. Comparing the METEOR-only to the preference model head-to-head, the generations produced by the human feedback model are preferred in 682 cases, compared to the METEOR-only model which is preferred in 587 cases ($p < 0.01$ the models are equally preferred).

Qualitative Analysis

This section shows sample generations from different algorithms for three randomly picked prompts.

Sample 1

Prompt: generate a sentence with: apron cut hat kitchen sausage

Zero-Shot: generate a sentence with: apron cut hat kitchen sausage.

PPO: sausage in the kitchen on an apron.

NLPO: sausage cut hat cut hat cut hat cut hat cut apron cut hat

Supervised: A man is cutting sausage on an apron in a kitchen.

Supervised + PPO: A man cutting sausage from a hat on an apron in the kitchen.

Supervised + NLPO: A man cuts sausage in a hat on an apron in the kitchen.

Sample 2

Prompt: generate a sentence with: backhand competition player reach shuttlecock

Zero-Shot: backhand competition player reach shuttlecock.

PPO: the ball in the backhand and the competition player in the shuttlecock.

NLPO: backhanded competition player reaches shuttlecock.

Supervised: tennis player reaches for a shuttlecock during a competition.

Supervised + PPO: football player reaches for a shuttlecock in backhand during the competition.

Supervised + NLPO: football player reaches for a shuttlecock after hitting a backhand during the competition.

Sample 3

Prompt: generate a sentence with: bowl combine cucumber onion pasta

Instructions (click to expand)

Thanks for your participation and work on this HIT!

In this task you will be presented with two sentences, each containing similar words. Your job is to **pick the sentence that you prefer**. While the judgment is ultimately subjective, consider the following factors:

- **Most important:** Which sentence makes the most sense? Which describes a plausible, realistic, and commonsensical scenario more effectively?
- Which generation is more grammatical, easy-to-read, and fluent?

In general, be forgiving of slight grammatical errors. If one conceptually makes much more sense than the other, you should prefer it even if there are slight issues with readability. However, if there are severe readability issues that effect comprehensibility, feel free to select the other option.

Thanks again for your efforts, we appreciate your work!

Please take a moment to read both choices. Select the more commonsensical, complete, and grammatical option. If they are both bad, still do your best to pick the one you prefer.

● A	● B
footballer receives instruction from football player when taking on a squad of people	footballer receives instruction from football player during a training session.

Figure C.5: **CommonGen Pairwise Study Setup:** Instructions and interface for the pairwise CommonGen HIT.

Zero-Shot: cucumber onion pasta.

PPO: the cucumber onion pasta in a small bowl.

NLPO: the in a bowl Combine cucumber onion pasta in a large bowl. Toss pasta

Supervised: A bowl containing pasta, tomatoes, cucumbers, and onions.

Supervised + PPO: A bowl containing pasta topped with cucumbers, onions, and peppers.

Supervised + NLPO: A bowl containing a mixture of pasta, cucumber, and onion.

C.1.5 CNN/DM

Setup As a representative of the summarization task, we consider CNN/DM dataset consisting of long news articles and their highlights written by news authors. The dataset consists of 287k training, 13k validation and 11k test examples. We trained RL methods using 3 different automated metrics, namely Rouge-1, Rouge-avg and Meteor. We chose T5 as our base LM as it is pre-trained in a unified text-to-text framework and relishes Zero-Shot capabilities. For decoding, we use multinomial sampling with a temperature of 0.7 for all the models.

Model Params	value
supervised	batch size: 16 epochs: 2 learning rate: 0.0001 learning rate scheduler: cosine weight decay: 0.1
ppo/ nlpo	steps per update: 5120 total number of steps: 512000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 rollouts top k: sweep of (50,100) top mask ratio: 0.9 target update iterations: sweep of (10, 20, 30)
supervised+ppo/ nlpo	steps per update: 5120 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.0 initial kl coeff: 0.01 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 value function coeff: 0.5 rollouts top k: sweep of (50,100) top mask ratio: 0.9 target update iterations: sweep of (10, 20, 30)
decoding	sampling: True temperature: 0.7 min length: 50 max new tokens: 100
tokenizer	padding side: left truncation side: right max length: 512

Table C.14: **CNN/DM Hyperparams**: Table shows a list of all hyper-parameters and their settings

Results and Discussion

Tables C.15, C.16 presents benchmarking results on test set reporting a wide range of metrics: lexical, semantic, factual correctness and diversity metrics. As baselines, we report lead-3 which selects first three sentences as the summary, Zero-Shot and a supervised model. PPO and NLPO models are on par with supervised performance on several metrics including Rouge-2, Rouge-L, and Bleu. On fine-tuning on top of supervised model, performance improves consistently on all metrics indicating that RL fine-tuning is beneficial. Another interesting finding is that, RL fine-tuned models are factually consistent as measured by SummaCZS metric. For ablations on PPO params, NLPO params, we refer to Tables C.17,C.18.

C.1 Experimental Details

Tasks	Alg	Reward Function	LM	Lexical and Semantic Metrics					Factual Consistency		
				Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BLEU	BertScore	SummaCZS
CNN/DM	Lead-3			0.401	0.175	0.250	0.363	0.333	0.099	0.874	0.993
	Zero-Shot		T5	0.372	0.145	0.247	0.311	0.256	0.077	0.864	0.654
	PPO	Rouge-1	T5	0.410	0.182	0.283	0.349	0.276	0.095	0.876	0.622
		Rouge-Avg	T5	0.396	0.176	0.273	0.338	0.270	0.095	0.874	0.622
		Meteor	T5	0.408	0.178	0.276	0.342	0.301	0.109	0.873	0.527
	NLPO	Rouge-1	T5	0.404	0.180	0.278	0.344	0.275	0.096	0.875	0.636
		Rouge-Avg	T5	0.404	0.177	0.279	0.344	0.274	0.094	0.874	0.586
		Meteor	T5	0.405	0.180	0.277	0.343	0.292	0.108	0.872	0.578
	Supervised		T5	0.411	0.177	0.276	0.343	0.309	0.108	0.876	0.654
	Supervised + PPO	Rouge-1	T5	0.417	0.189	0.294	0.358	0.278	0.101	0.882	0.722
		Rouge-Avg	T5	0.425	0.194	0.297	0.363	0.296	0.114	0.882	0.728
		Meteor	T5	0.426	0.194	0.293	0.361	0.316	0.125	0.880	0.726
	Supervised + NLPO	Rouge-1	T5	0.421	0.193	0.297	0.361	0.287	0.108	0.882	0.740
		Rouge-Avg	T5	0.424	0.193	0.296	0.363	0.295	0.115	0.882	0.743
		Meteor	T5	0.429	0.194	0.293	0.361	0.319	0.124	0.880	0.743

Table C.15: CNN/Daily Mail test evaluation (Lexical and Semantic metrics): Table presents a wide range of metrics: lexical, semantic and factual correctness. As baselines, we report lead-3 which selects first three sentences as the summary, Zero-Shot and a supervised model. PPO and NLPO models are on par with supervised performance on several metrics including Rouge-2, Rouge-L, and Bleu. On fine-tuning on top of supervised model, performance improves consistently on all metrics indicating that RL fine-tuning is beneficial. Another interesting finding is that, RL fine-tuned models are factually consistent as measured by SummaCZS metric.

Tasks	Alg	Reward Function	LM	Diversity Metrics							Mean Output Length
				MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂	
CNN/DM	Lead-3			0.750	0.0482	0.386	10.481	16.631	21465	273153	84
	Zero-Shot		T5	0.725	0.061	0.414	10.285	16.183	19113	193999	55
	PPO	Rouge-1	T5	0.760	0.068	0.464	10.661	16.437	18189	191383	47
		Rouge-Avg	T5	0.773	0.071	0.490	10.830	16.664	19478	209140	48
		Meteor	T5	0.765	0.060	0.447	10.699	16.688	20528	234386	61
	NLPO	Rouge-1	T5	0.771	0.069	0.480	10.789	16.618	18677	201971	48
		Rouge-Avg	T5	0.765	0.066	0.476	10.744	16.620	18179	206368	50
		Meteor	T5	0.772	0.064	0.471	10.802	16.766	20212	231038	56
	Supervised		T5	0.727	0.057	0.401	10.459	16.410	21096	230343	68
	Supervised + PPO	Rouge-1	T5	0.750	0.070	0.459	10.595	16.389	18184	184220	46
		Rouge-Avg	T5	0.747	0.066	0.445	10.589	16.458	18939	200617	52
		Meteor	T5	0.741	0.059	0.420	10.532	16.491	20395	224432	63
	Supervised + NLPO	Rouge-1	T5	0.748	0.067	0.446	10.528	16.313	18204	185561	48
		Rouge-Avg	T5	0.744	0.065	0.443	10.570	16.444	18747	201705	53
		Meteor	T5	0.745	0.059	0.422	10.574	16.516	20358	226801	63

Table C.16: CNN/Daily Mail test evaluation (Diversity Metrics): Table presents a wide range of diversity metrics on test set

Alg	Reward Function	Top k	Lexical and Semantic Metrics						
			Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BLEU	BertScore
PPO	Rouge-1	50	0.404	0.181	0.280	0.346	0.273	0.095	0.874
		100	0.412	0.186	0.286	0.354	0.276	0.094	0.876
	Rouge-Avg	50	0.401	0.177	0.276	0.342	0.271	0.092	0.873
		100	0.399	0.179	0.275	0.342	0.270	0.094	0.874
	Meteor	50	0.413	0.182	0.279	0.348	0.301	0.110	0.873
		100	0.409	0.179	0.276	0.345	0.296	0.108	0.871
Supervised+PPO	Rouge-1	50	0.414	0.190	0.293	0.358	0.272	0.097	0.881
		100	0.420	0.193	0.295	0.362	0.277	0.100	0.881
	Rouge-Avg	50	0.426	0.196	0.298	0.366	0.294	0.114	0.881
		100	0.427	0.196	0.298	0.366	0.294	0.113	0.881
	Meteor	50	0.429	0.197	0.297	0.367	0.306	0.122	0.881
		100	0.432	0.199	0.297	0.367	0.317	0.131	0.879

Table C.17: **PPO Ablation/Model Selection:** Evaluation of PPO models on validation set with different reward functions and top k values for rollouts. For each alg-reward combo, best model (top k) is chosen.

Alg	Reward Function	Top k (rollout)	Top p (Action mask)	target update n_{iters}	Lexical and Semantic Metrics						
					Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BLEU	BertScore
NLPO	Rouge-1	50	0.9	10	0.400	0.178	0.275	0.343	0.269	0.094	0.872
				20	0.396	0.173	0.274	0.340	0.257	0.082	0.873
				30	0.396	0.174	0.273	0.339	0.265	0.091	0.872
		100	10	0.407	0.177	0.279	0.347	0.265	0.085	0.875	
			20	0.406	0.182	0.281	0.347	0.273	0.094	0.874	
			30	0.405	0.180	0.279	0.347	0.269	0.091	0.875	
	Rouge-Avg	50	0.9	10	0.400	0.180	0.276	0.343	0.271	0.096	0.873
				20	0.349	0.147	0.241	0.298	0.237	0.078	0.858
				30	0.393	0.173	0.272	0.336	0.267	0.092	0.870
		100	10	0.396	0.174	0.274	0.339	0.265	0.088	0.872	
			20	0.406	0.179	0.280	0.347	0.272	0.092	0.874	
			30	0.400	0.178	0.279	0.344	0.266	0.087	0.874	
Meteor	50	0.9	10	0.404	0.177	0.274	0.343	0.286	0.102	0.872	
			20	0.406	0.180	0.276	0.343	0.292	0.107	0.871	
			30	0.401	0.172	0.271	0.337	0.288	0.099	0.870	
	100	10	0.405	0.178	0.276	0.343	0.294	0.107	0.870		
		20	0.406	0.176	0.276	0.343	0.291	0.106	0.872		
		30	0.409	0.184	0.280	0.348	0.291	0.108	0.873		
Supervised + NLPO	Rouge-1	50	0.9	10	0.425	0.196	0.299	0.366	0.285	0.106	0.882
				20	0.417	0.191	0.295	0.360	0.276	0.100	0.881
				30	0.418	0.192	0.296	0.361	0.278	0.101	0.881
		100	10	0.424	0.196	0.299	0.366	0.286	0.106	0.882	
			20	0.423	0.196	0.299	0.365	0.289	0.110	0.881	
			30	0.420	0.193	0.296	0.362	0.279	0.102	0.881	
	Rouge-Avg	50	0.9	10	0.426	0.197	0.298	0.367	0.294	0.115	0.881
				20	0.425	0.196	0.298	0.366	0.292	0.112	0.881
				30	0.424	0.194	0.297	0.365	0.287	0.107	0.881
		100	10	0.424	0.196	0.298	0.365	0.291	0.113	0.881	
			20	0.428	0.198	0.300	0.368	0.296	0.115	0.882	
			30	0.429	0.199	0.300	0.369	0.296	0.116	0.882	
Meteor	50	0.9	10	0.430	0.197	0.294	0.364	0.320	0.130	0.879	
			20	0.432	0.198	0.297	0.367	0.318	0.130	0.880	
			30	0.423	0.191	0.293	0.361	0.297	0.116	0.879	
	100	10	0.435	0.200	0.298	0.369	0.320	0.131	0.881		
		20	0.433	0.198	0.297	0.368	0.319	0.130	0.879		
		30	0.434	0.200	0.297	0.369	0.324	0.132	0.879		

Table C.18: **NLPO Ablation/Model Selection:** Evaluation of NLPO models on validation set with different reward functions, top k values for rollouts and target update iterations. For each alg-reward combo, best model is chosen

Algorithm	Unique N	Coherence			Quality		
		Value	Alpha	Skew	Value	Alpha	Skew
PPO+Supervised	22	4.21	0.198	4.224	3.97	0.256	3.98
NLPO+Supervised	19	4.3	0.26	4.308	3.98	0.089	4
Zero Shot	17	3.73	0.1	3.757	3.69	0.25	3.722
Supervised	19	4.25	0.116	4.241	3.99	0.2	3.986
NLPO	17	4.03	0.13	4.042	3.83	0.191	3.832
PPO	21	3.94	0.111	3.945	3.76	0.129	3.767
Human	19	3.89	0.277	3.902	3.77	0.029	3.769

Table C.19: **CNN/DM Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorff’s alpha showing inter-annotator agreement, and Skew. For each model a total of 50 samples were drawn randomly from the test set and rated by 3 annotators each, each resulting in 150 data points per algorithm.

Group 1	Group 2	Coherence		Quality	
		Diff (G2-G1)	<i>p-values</i>	Diff (G2-G1)	<i>p-values</i>
Human	NLPO	0.147	0.755	0.060	0.900
Human	NLPO+Supervised	0.413	0.001	0.213	0.047
Human	PPO	0.053	0.900	-0.007	0.900
Human	PPO+Supervised	0.327	0.024	0.200	0.544
Human	Supervised	0.360	0.008	0.220	0.043
Human	Zero Shot	-0.160	0.679	-0.080	0.900
NLPO	NLPO+Supervised	0.267	0.012	0.153	0.008
NLPO	PPO	-0.093	0.900	-0.067	0.900
NLPO	PPO+Supervised	0.180	0.564	0.140	0.860
NLPO	Supervised	0.213	0.361	0.160	0.754
NLPO	Zero Shot	-0.307	0.044	-0.140	0.860
NLPO+Supervised	PPO	-0.360	0.008	-0.220	0.043
NLPO+Supervised	PPO+Supervised	-0.087	0.009	-0.013	0.009
NLPO+Supervised	Supervised	-0.053	0.009	0.007	0.900
NLPO+Supervised	Zero Shot	-0.573	0.001	-0.293	0.012
PPO	PPO+Supervised	0.273	0.106	0.207	0.508
PPO	Supervised	0.307	0.044	0.227	0.394
PPO	Zero Shot	-0.213	0.361	-0.073	0.900
PPO+Supervised	Supervised	0.033	0.900	0.020	0.900
PPO+Supervised	Zero Shot	-0.487	0.001	-0.280	0.155
Supervised	Zero Shot	-0.520	0.001	-0.300	0.101

Table C.20: **CNN/DM Human Study Tukey Results:** Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding *p-values*. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall *p-values* showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.

Instructions (click to expand)

In this HIT you will be presented with a news article an automatically-generated summary of that article. Your job is to rate the the system generation across 2 axes:

- **Coherence/Fluency:** *Is the system's generation grammatical, easy-to-read, and well-written?*
- **Summary Quality:** *Does the system's generation meaningfully capture the main points in the article?*

You will be able to rate each of the three axes on a scale from 1 to 5, with **1 being the lowest/worst** and **5 the highest/best**. The specific scales are:

- **Coherence/Fluency:**
 - **5/5 (excellent):** The summary itself is grammatical, fluent, and reasonable.
 - 4/5 (good): The summary generally makes sense, but there are minor grammatical errors or topical shifts that don't make for the best writing.
 - 3/5 (okay): I can see why this summary was generated, and it's somewhat readable, but there are problems that can't be ignored.
 - 2/5 (poor): Some parts of the summary could make sense, but it's unnatural, illogical, or quite hard to read.
 - **1/5 (terrible):** The summary has nothing to do with the article, and/or there are severe errors in grammaticality or fluency.
- **Summary Quality**
 - **5/5 (very good):** The summary correctly and completely captures the key points of the article.
 - 4/5 (mostly good): The summary captures most of the key points from the article.
 - 3/5 (neutral): While there are no egregious inconsistencies, there are major missing key points, or some partly incorrect summarizations.
 - 2/5 (mostly bad): The summary mentions some things from the article, but there are few reasonable points.
 - **1/5 (awful):** The summary has nothing to do with the article or severely contradicts it.

You don't need to read the entire article, but --- please do skim it to get a sense of the main points and how it relates to the summary. In particular, you should focus on two things when evaluating summary quality:

- *Does the summary cover the main points of the story?*
- *Are the facts/references made in the summary represented in the article? Or are facts misconstrued or made-up?*

In early testing doing this HIT ourselves, we found it useful to not display the sliders until after a time delay, so we've made it so the sliders are hidden for 15 seconds. They should appear automatically.

Article:

\$(prompt)

System's summary (rate this!):

\$(machine_completion)

Please take time to read the system's summary and to skim the article briefly --- then, rate the system's summary on the form below (appears in ~15s...). For "Summary quality" --- please reference the article to check: 1) if the summary contains the key points; and 2) if the specific details mentioned in the summary are correct.

Coherence/Fluency: 3/5

Is the system's generation *grammatical, easy-to-read, and well-written?*

Bad Excellent

I can see why this summary was generated, and it's somewhat readable, but there are problems that can't be ignored.

Summary Quality: 3/5

Does the system's generation *meaningfully capture the main points in the article?*

Terrible Summary Great Summary

While there are no egregious inconsistencies, there are major missing key points, or some partly incorrect summarizations.

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Submit

Figure C.6: CNN/DM Human Study Setup: Instructions and interface for the summarization task.

Human Participant Study

Figure C.6 shows the summarization instructions and interface used for the human evaluation experiments. Participants weren't required to read the entire article, but to encourage some reading, a minimum time on the window of 15s was enforced via hiding the sliders. Tables C.19, C.20 show averaged results, annotator agreement, and the results of statistical significance tests to determine which models output better generations when rated by humans.

Qualitative Analysis

We show sample generations from each of the algorithms for three randomly picked prompts below.

Sample 1

Prompt: Manchester City are confident UEFA's punishment for breaching financial fairplay regulations will be lifted this summer which would allow them to bid for stellar names like Raheem Sterling, Gareth Bale, Kevin de Bruyne and Ross Barkley. City boss Manuel Pellegrini has been hampered over the past year by UEFA restricting them to a net transfer spend of 49million in each window and keeping the club's overall wage bill to its current level of 205million-a-year. UEFA's settlement with City published in May stated those penalties would remain in place until the end of the 2015/16 season but the club's latest financial figures showed drastically-reduced losses of 23million which they feel proves they are now compliant with FFP

regulations. Manuel Pellegrini is hoping that the financial restrictions imposed by UEFA for a breach of FFP rules will be lifted at the end of this season. Manchester City have been limited to a net spend of 49million in the last two transfer windows - they spent 25m bringing Wilfried Bony in from Swansea in January. Ahead of Monday night's trip to Crystal Palace, Pellegrini was certainly talking like a man excited at the prospect of signing 'crack' players this summer. "I think that next season we don't have any restrictions so we will be in the same position that all the other English clubs have," said Pellegrini. "It's important. You have so many strong teams here in England and in Champions League, you can not allow them to keep the advantage every year; having less players to put in your squad or spending less money. We spend money, of course we always spend money, but they spent more." Manchester United, Barcelona, Liverpool and Arsenal have all paid more in transfer fees in the past 12 months than City who were traditionally Europe's biggest spenders after the club was taken over by Abu Dhabi owners in 2008. Uefa also ordered City to play with a reduced squad from 25 players to 21 in the Champions League this season and while that restriction has now ended, any time reduction in the penalties on spending and wages is more controversial. Arsenal have paid more in transfer fees than City in the last 12 months, including 30m on Alexis Sanchez. The document published last May by UEFA's Club Financial Control Body investigative chamber explicitly said City's financial penalties would run for two seasons at least and there has been no official deviation from that decision.

The published statement said at the time: "Manchester City agrees to significantly limit spending in the transfer market for the seasons 2014/15 and 2015/16. It means City will have to argue their case with Uefa that as they have been financially compliant over the past year, they deserve to be free of restrictions moving forward. They have successfully argued their case with UEFA before. Last summer they persuaded the governing body to allow them to bypass the normal quota of eight homegrown players as their Champions League squad had been reduced. Eliaquim Mangala joined the champions from Porto for 32m last summer. The reigning Premier League champions have only paid money for Fernando, Willie Caballero, Eliaquim Mangala and Wilfried Bony in the last two transfer windows and that was part-paid by the sales of Javi Garcia and Jack Rodwell. Pellegrini admitted they weren't in a position to compete with neighbours United for established world stars like Angel di Maria because of the FFP penalties. It has cost City on the pitch. Even if they win their remaining eight Premier League fixtures their points tally will be lower than last season and they crashed out once again at the last-16 stage of the Champions League. Pellegrini and director of football Txiki Begiristain both accept the squad needs replenishing and they want to be in the market for the top available players. Manchester City have been heavily linked with a move for former Chelsea midfielder Kevin de Bruyne. The club also want to be involved in any race for Liverpool contract rebel Raheem Sterling. They have been strongly linked with former Chelsea winger Kevin de Bruyne who has been one of the best players in the Bundesliga with Wolfsburg. And if Liverpool do allow Sterling to leave, City want to be involved in the chase for one of England's most outstanding talents to help fit in with quotas for homegrown players. "If you are the manager or a player of a big team, you are always open to criticism because everyone expects a high level of performance," he said. "The major issue with our team always seems to be the money but this season I think we were the team that spent less than any other team. That is the weird thing. For this club trying to be a big team in so few years, maybe we are paying the cost for that." Since the transfer penalties were introduced, City have spent 91million on players in the last two windows (Net spend 68million). That compares to Manchester United 145.5million, Liverpool 113million, Arsenal 92.5million and Chelsea 82.6million. Over the same time period Barcelona spent 118.3million on players and Real Madrid 81.2million though they also broke the world transfer record for Gareth Bale the previous summer.

Zero-Shot: manuel Pellegrini hoping UEFA's punishment for breaching financial fairplay regulations will be lifted this summer. city have been restricted to a net transfer spend of 49million in each window. a total of 23million has been lost in the last year.

References: Manchester City have been restricted to a net transfer spend of 49m. Club's also had to keep overall wage bill to its current level of 205m. Punishments imposed by UEFA for breaching Financial Fair Play rules. The spending restrictions were set for this season and the next one. But City are confident they will be lifted early after their compliance.

PPO: Manuel Pellegrini is hoping UEFA's punishment for breaching financial fairplay rules will be lifted this summer. Manchester City have been limited to a net spend of 49million in each transfer window. the club also keep the club's overall wage bill to its current level of 205million-a-year.

NLPO: Manchester City are confident UEFA's punishment for breaching financial fairplay regulations will be lifted this summer. that would allow them to bid for stellar names like Raheem Sterling, Kevin de Bruyne and Ross Barkley.

Supervised: Manchester City have been limited to a net transfer spend of 49million in each window. City's overall wage bill is 205m-a-year due to a breach of UEFA rules. City are confident the financial restrictions will be lifted at the end of the current season in order to bid for stars such as Raheem Sterling. Manuel Pellegrini is hoping that the financial restrictions will be lifted at the end of the 2015/16 season

Supervised + PPO: Manchester City have been restricted to a net spend of 49million in each window and kept their overall wage bill at 205m. Manuel Pellegrini believes these restrictions will be lifted at the end of the season. Arsenal have spent more in transfer fees than City in the last 12 months, including 30m on Alexis Sanchez.

Supervised + NLPO: Manchester City are confident UEFA's punishment for a breach of financial fairplay regulations will be lifted this summer. The club have been restricted to a net transfer spend of 49million over the past two window. They aim to bid for stars like Raheem Sterling, Gareth Bale and Kevin de Bruyne. City lost 23m on transfers in the last 12 months due to financial restrictions imposed by UEFA for breach

Sample 2

Prompt: (CNN)Larry Johnson remembers the fear and feeling of helplessness from being on the SkyWest Airlines flight that made an emergency landing in Buffalo, New York. "I thought we were done," he said Thursday, one day later. "There was no doubt in my mind that we weren't going to land." Johnson was flying with his brother, his girlfriend and his 8-month-old son when he says a flight attendant came over the speaker asking for someone who was medically trained to help with a sick passenger. Minutes later, Johnson says, the attendant announced there was a pressurization problem and told passengers to prepare for the emergency landing. "You're going to feel dizzy and woozy and lightheaded, and you're not going to be able to breathe," Johnson recalled the flight attendant saying. The Federal Aviation Administration on Wednesday initially reported a pressurization problem with SkyWest Flight 5622, and said it would investigate. It later issued a statement that did not reference any pressurization issues. "It was like being trapped and you couldn't do anything. You have no control over it. Your body just felt pressured," Johnson said. Johnson said his girlfriend was seeing spots and that "when I was blinking, it was really bright. I could barely see." According to Marissa Snow, spokeswoman for SkyWest, three passengers reported a loss of consciousness while on the flight. Fifteen passengers were examined by medical personnel at the Buffalo airport, but no one was transported to the hospital, she said. The spokeswoman said that maintenance personnel found no indication of a pressurization problem with the aircraft, an Embraer E170, and that the airline continues to investigate the cause. An official with the National Transportation Safety Board told CNN that the agency is in communication with the FAA and SkyWest to gather information on the incident to better understand what took place.

Mary Cunningham was the nurse on board who attended to the first sick passenger. "There was a woman, very lethargic, very nauseous, turning gray, just not looking good," Cunningham said. After administering oxygen to the woman, Cunningham, a nurse at Yale-New Haven Hospital in Connecticut, returned to her seat. Soon after, Cunningham said, she was called up again by the flight attendant after another woman near the first started to lose consciousness. "It was something that was affecting oxygen in that area," which was near the middle of the plane, she said. Cunningham said she had to return to her seat after feeling lightheaded. There were 84 passengers on board the plane, including nine crew members. Flight 5622 was originally scheduled to fly from Chicago to Hartford, Connecticut. The plane descended 28,000 feet in three minutes. "It would feel like a roller coaster -- when you're coming over the top and you're going down," CNN aviation analyst Mary Schiavo said, describing how such a descent would feel. "You know that these pilots knew they were in a very grave and very serious situation." Johnson, flying to visit his parents with their only grandchild, was glad that roller coaster ride ended safely. "It was a shaky ride. It wasn't

anything I would like to experience again," he said. CNN's Jean Casarez, Josh Gaynor, Dana Ford and Stephanie Gallman contributed to this report.

Zero-Shot: "you're going to feel dizzy and woozy and lightheaded," Johnson says. "it was like being trapped and you couldn't do anything," Johnson says. "the plane was trying to get back on the ground," passenger says of fear.

References: Three passengers report a loss of consciousness on SkyWest flight .But officials say there is no evidence of a pressurization problem .

PPO: SkyWest flight 5312 made emergency landing in Buffalo, new york. passengers were told they were not going to be able to breathe, passenger recalls attendant. the faa initially reported a pressurization problem with the flight 5622.

NLPO: SkyWest flight 5622 made emergency landing in buffalo, new york.Larry Johnson says flight attendant announced there was a pressurization problem. three passengers reported loss of consciousness while on flight; no one transported to hospital.

Supervised: The FAA says it is investigating the cause of the emergency landing. SkyWest Flight 5622 made an emergency landing in Buffalo, New York. The flight was carrying a family of three people. The flight attendant told the passengers to prepare for the landing.

Supervised + PPO: "I thought we were done," the father of one passenger says. The flight attendant announced there was a pressurization problem. The flight made an emergency landing in Buffalo, New York. Three passengers reported a loss of consciousness while on the flight.

Supervised + NLPO: SkyWest Airlines flight made an emergency landing in Buffalo, New York, on Thursday. "There was no doubt in my mind that we weren't going to land," passenger says. "You're going to feel dizzy and woozy and lightheaded...you're not going to be able to breathe," he says."

Sample 3

Prompt: For Inverness the latest chapter in their remarkable story featured not one, but two, plot-changing twists. One featured penalty Celtic didn't get for 2-0 in which Josh Meekings handball should have also led to a sending off. The other the spot kick they did, followed by a red card for Craig Gordon. 'I've not seen it yet, but going by the reaction of the Celtic players we got away with a penalty and a sending off and that was probably the turning point in the game,' acknowledged Caley manager John Hughes after. Inverness's Josh Meekings appears to get away with a handball on the line in their win over Celtic . Caley boss John Hughes says the break, which could have meant a penalty and red card, was a turning point . 'I've not spoken to Josh. I haven't seen it - but going by the media it was definitely a hand ball. We look at the referee behind the line and all that and I know Ronny will feel aggrieved - because I certainly would. 'But it's part and parcel of football and you need a wee bit of luck to beat Celtic. 'This was their biggest game of the season because they will go on and win the league and if they had beaten us today there was a good chance they would have gone on and won the Scottish Cup. 'But when Marley Watkins was clipped by Craig Gordon and they were down to 10 men that was advantage Inverness. ' We weren't going to give Celtic the ball back, they had to come and get it and we had to be patient. 'When big Edward put us into the lead we thought it was going to be our day on the back of things that had happened. 'Celtic equalised with another free kick but it's typical of Inverness that we don't do anything easy. 'We do it the hard way and we came up with the winner through David Raven.' Hughes hauled Raven, his Scouse defender, from his backside as extra-time beckoned. Offended by the sight of one of his players resting he had a message to impart. Caley players celebrate after upsetting Celtic in a Scottish Cup semi-final 3-2 thriller . Celtic, depleted by games and absentees, were virtually on their knees after a relentless programme of midweek games. In last season's League Cup Final Inverness had been passive and unambitious prior to losing on penalties. This was no time to repeat the mistake. 'I tried to emphasise to the players they would never have a better time to go on and beat Celtic, down to 10 men in the semi final of a cup. We needed to go for it,' Hughes said. 'Before Raven scored at the back post I was looking to change it. I was going to bring on another winger, Aaron Doran, and put him in the full-back position over on the right, but more advanced so he could take their left back on. Thankfully I didn't do that and David Raven came up with the goal. Virgil Van Dijk (centre) fired Celtic into an early lead with a superb free-kick in the 18th minute . 'I didn't realise this is the first time the club have been in the final of the Scottish Cup and that's a remarkable achievement given it was only formed 20 years ago. 'It is a

great story isn't it? It's an absolutely fantastic story. It is 20 odd years since the amalgamation. We are a small provincial club up there in the Highlands. 'We have lost a real inspirational skipper in Richie Foran right from the start of the season. He has never played. We have had to adjust to that. 'We had to sell Billy McKay, our top goalscorer, at Christmas. We have had to go again and adjust. I am a very humble guy and I am grateful and thankful that injuries have never caught up with us.'

There is remarkable irony in the fact Falkirk will be the opponents for the final. A former Bairns captain, he was manager of the club in 2009 when they lost to Rangers at Hampden. Former Falkirk captain and manager John Hughes will take on his former club in the final. 'I had a lot of great times at Falkirk. So much so that it is possibly my favourite time in my playing career. I am still friendly with an awful lot of the characters who were in that dressing room. Neil Oliver is a good friend of mine from my Falkirk days. He comes along and sits on the bench and gives me a wee hand out. 'That is the spirit that we had at that club. I have met some great guys. Brian Rice, my ex-coach, Davie Weir, Ian McCall, the list is endless. I was just talking the other day about that day at Falkirk. There are times even now when I see. 'I have a real special, special feel for the place. I am not kidded Falkirk is a right good side, trust me on that. No side goes to Tynecastle and beats Hearts without being a good side. No side beats Hibs the way they were playing without being a good side. You might say we are favourites I say it is even stevens and I am just looking forward to when the days comes.' David Raven (second right) celebrates with his Inverness team-mates following his extra-time winner. An Inverness and Falkirk final would not have been the choice of sponsors William Hill, you suspect. Vast swathes of Hampden's west stand were empty, a 12.15 kick-off for the benefit of broadcasters proving unhelpful. There will be no ticket rows this time. Not over a final unlikely to raise a crowd much higher than 35,000 in a stadium of 52,000. The contract with Hampden plc is signed, with no question of moving it. If the bitter amalgamation of 21 years ago produced a fine football club it also disenfranchised many of those who followed the old ones. 'I hope this encourages the next generation to follow Inverness,' added Hughes. 'In the peer groups it is still Celtic and Rangers they follow but I hope the next ones will want to support their local club. ' If we can add another few hundred or thousand to the support then we are playing our part.

Zero-Shot: Inverness beat Celtic 2-0 in their opening game of the season. Josh Meekings was the subject of a handball after the break. the break could have meant a penalty and a red card for Craig Gordon.

References: Josh Meekings handball should have been a penalty and a red card. If the penalty was awarded and converted Celtic would have gone to 2-0. Caley manager John Hughes admitted his side were fortunate in win. Virgil van Dijk scored Celtic's opener with a superb free-kick. Celtic keeper Craig Gordon was sent off early in the second half. Greg Tansey, Edward Ofere and finally Daven Raven scored for Caley.

PPO: Josh Meekings handball could have meant sending off for Celtic in extra-time win. John Hughes says the break could have meant an penalty and red card. Caley boss also admits he would feel aggrieved had Celtic won the championship.

NLPO: Caley beat Celtic 2-0 in their Scottish Cup clash. Josh Meekings handball could have led to a penalty and red card. John Hughes says the break was probably the turning point in the game. defender does not believe defensive striker can get away with being shown the yellow card - despite playing in same half we never scored.

Supervised: Inverness beat Celtic 2-0 in their Scottish Cup clash. Josh Meekings appeared to get away with a handball on the line. Another player was sent off for a spot kick which ended in a red card. Caley manager John Hughes says the break could have meant a penalty. Hughes says he 'would feel aggrieved' after the break.

Supervised + PPO: Caley beat Celtic 1-0 in their Scottish Cup encounter at the Etihad. Josh Meekings' handball on the line should have also seen a sending off. But Craig Gordon got away with a spot kick and sending off. John Hughes said it was a turning point in the game for Inverness.

Supervised + NLPO: Josh Meekings appears to get away with a handball after Inverness's win. Inverness were down to 10 men when Craig Gordon was sent off. John Hughes says the break could have meant a penalty and red card. Hughes says Celtic will go on and win the Scottish Cup with win

C.1.6 ToTTo

Setup ToTTo [294] is a controlled table-to-text generation task in which the goal is to produce one-sentence description of highlighted table cells. For training RL methods, we consider 5 different reward functions: BLEU, SacreBLEU, METEOR, PARENT and a combination of Meteor and PARENT. We chose T5 as our base LM here too, as they are more suitable for structure to text tasks. For decoding, we use beam search during inference and for generating rollouts, we use top k sampling. Other implementation details are captured in Table C.21.

Model Params	value
supervised	batch size: 8 epochs: 4 learning rate: 0.0001 learning rate scheduler: constant with warm up weight decay: 0.1
ppo/nlpo	steps per update: 2560 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 2.0 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 0 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
supervised+ ppo (or nlpo)	steps per update:2560 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.0000005 entropy coefficient: 0.0 initial kl coeff: 0.01 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 50 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
decoding	num beams: 5 min length: 10 max new tokens: 50
tokenizer	padding side: left truncation side: right max length: 512

Table C.21: **ToTTO Hyperparams**: Table shows a list of all hyper-parameters and their settings

Results and Discussion

Tables C.22, C.23, C.24 presents our benchmarking results with 5 reward functions along with supervised baseline performances on dev and test sets respectively. Similar to other tasks, our main finding is that warm-started initial policies are crucial for learning to generate descriptions from highlighted cells. Without warm-start, policies suffer from reward hacking and resulting in

sub-optimal solutions despite application of task-specific metrics such as PARENT etc. We find that Supervised+NLPO method outperforms all models on ToTTo leaderboard in terms of PARENT metric.

Tasks	Alg	LM	Reward function	Lexical and Semantic Metrics					SacreBleu			Factual Consistency PARENT			
				Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BertScore	Overall	Overlap	Non-Overlap	Overall	Overlap	Non-Overlap
Zero-Shot	T5			0.131	0.055	0.127	0.127	0.057	0.805	0.038	0.042	0.034	0.118	0.119	0.116
Supervised	T5			0.410	0.279	0.388	0.388	0.223	0.953	0.458	0.533	0.387	0.586	0.633	0.540
ToTTo	PPO	bleu		0.274	0.138	0.249	0.249	0.139	0.844	0.068	0.071	0.066	0.251	0.250	0.251
		sacrebleu		0.341	0.166	0.300	0.300	0.165	0.858	0.09	0.094	0.086	0.300	0.299	0.300
		meteor		0.322	0.157	0.286	0.286	0.173	0.888	0.147	0.163	0.133	0.358	0.367	0.350
	NLPO	parent		0.268	0.125	0.251	0.251	0.119	0.890	0.150	0.158	0.143	0.337	0.332	0.342
		meteor + parent		0.266	0.128	0.251	0.251	0.130	0.886	0.165	0.175	0.155	0.348	0.346	0.350
		bleu		0.267	0.134	0.24	0.24	0.137	0.84	0.068	0.071	0.065	0.238	0.239	0.237
	Supervised + PPO	sacrebleu		0.341	0.168	0.297	0.297	0.183	0.863	0.089	0.093	0.085	0.32	0.324	0.317
		meteor		0.322	0.157	0.286	0.286	0.173	0.888	0.147	0.163	0.133	0.358	0.367	0.350
		parent		0.283	0.132	0.264	0.264	0.133	0.894	0.163	0.174	0.153	0.36	0.357	0.364
		meteor + parent		0.299	0.14	0.276	0.276	0.142	0.896	0.171	0.181	0.161	0.369	0.365	0.372
		bleu		0.408	0.283	0.388	0.388	0.222	0.954	0.477	0.549	0.405	0.596	0.644	0.550
		sacrebleu		0.395	0.275	0.378	0.378	0.211	0.955	0.477	0.554	0.401	0.577	0.621	0.535
Supervised + NLPO	meteor		0.410	0.282	0.389	0.389	0.223	0.954	0.469	0.540	0.398	0.593	0.642	0.547	
	parent		0.401	0.277	0.382	0.382	0.215	0.953	0.470	0.543	0.394	0.598	0.647	0.550	
	meteor + parent		0.406	0.281	0.386	0.387	0.220	0.954	0.473	0.544	0.399	0.600	0.648	0.553	
	bleu		0.410	0.283	0.388	0.388	0.222	0.954	0.476	0.548	0.404	0.597	0.644	0.552	
	sacrebleu		0.397	0.276	0.38	0.38	0.214	0.955	0.477	0.555	0.401	0.581	0.628	0.535	
	meteor		0.411	0.283	0.389	0.39	0.224	0.954	0.474	0.547	0.403	0.6	0.649	0.554	
Supervised + NLPO	parent		0.405	0.28	0.386	0.386	0.219	0.954	0.469	0.541	0.398	0.598	0.645	0.552	
	meteor + parent		0.405	0.28	0.386	0.386	0.219	0.954	0.474	0.547	0.398	0.598	0.646	0.552	

Table C.22: **ToTTo dev evaluation (Lexical, Semantic and Factual Consistency Metrics)**: Table shows lexical, semantic and factual correctness metric scores of algorithms with different reward functions on dev set. Without supervised pre-training, both PPO and NLPO results in sub-optimal solutions, with NLPO better than PPO. With supervised pre-training, PPO and NLPO achieve better scores across all metrics showing RL fine-tuning is beneficial. Most importantly, RL fine-tuned models produce more factually correct text as seen in higher PARENT scores. Another observation, fine-tuning with a task-specific metric PARENT is better than training just on task-agnostic lexical metrics

Tasks	Alg	LM	Reward function	Diversity Metrics							
				MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂	Mean Output Length
	Zero-Shot	T5		0.428	0.084	0.238	6.703	9.933	8387	26490	19.964
	Supervised	T5		0.715	0.162	0.511	9.995	14.468	15168	54706	17.791
	PPO	T5	bleu	0.403	0.091	0.308	10.659	14.511	7536	34232	28.545
		T5	sacrebleu	0.469	0.121	0.407	11.071	14.880	10138	48195	26.612
		T5	meteor	0.625	0.136	0.482	10.189	14.910	12346	54925	21.484
		T5	parent	0.764	0.202	0.646	11.068	14.988	13068	50313	13.035
		T5	meteor + parent	0.702	0.181	0.594	10.096	14.432	14422	55770	15.354
	NLPO	T5	bleu	0.448	0.1	0.359	11.259	14.623	9029	47209	28.472
		T5	sacrebleu	0.494	0.111	0.373	11.007	15.032	9455	43379	27.977
ToTTo		T5	meteor	0.625	0.136	0.482	10.189	14.910	12346	54925	21.484
		T5	parent	0.824	0.223	0.691	11.493	15.127	14344	55542	14.204
		T5	meteor + parent	0.779	0.214	0.674	11.072	15.275	14939	58737	15.141
		Supervised + PPO	T5	bleu	0.722	0.167	0.525	10.080	14.524	15203	54724
		T5	sacrebleu	0.728	0.174	0.539	10.086	14.518	14846	52327	16.063
		T5	meteor	0.718	0.165	0.516	10.037	14.467	15182.0	54446	17.542
		T5	parent	0.732	0.174	0.545	10.209	14.660	15379.0	55421	16.826
		T5	meteor + parent	0.727	0.170	0.532	10.143	14.586	15330	55211	17.185
	Supervised + NLPO	T5	bleu	0.721	0.167	0.524	10.077	14.532	15213	54948	17.408
		T5	sacrebleu	0.729	0.174	0.54	10.124	14.544	14940	52986	16.334
		T5	meteor	0.727	0.171	0.536	10.156	14.612	15341	55292	17.637
		T5	parent	0.716	0.165	0.519	10.019	14.5	15218	54793	17.095
		T5	meteor + parent	0.727	0.171	0.536	10.156	14.612	15341	55292	17.095

Table C.23: **ToTTo dev evaluation (Diversity Metrics)** : Table shows diversity metrics of algorithms with different reward functions on dev set.

Tasks	Alg	LM	Reward function	Lexical and Semantic Metrics						Factual Consistency		
				SacreBleu			BLEURT			PARENT		
				Overall	Overlap	Non-Overlap	Overall	Overlap	Non-Overlap	Overall	Overlap	Non-Overlap
ToTTo	Zero-Shot	T5		0.036	0.040	0.032	-1.392	-1.387	-1.397	0.116	0.119	0.112
	PPO	T5	bleu	0.065	0.067	0.063	-1.074	-1.045	-1.098	0.246	0.246	0.244
		T5	sacrebleu	0.086	0.090	0.083	-0.979	-0.955	-1.003	0.293	0.292	0.294
		T5	meteor	0.144	0.155	0.132	-0.769	-0.713	-0.826	0.356	0.361	0.351
		T5	parent	0.146	0.153	0.128	-0.721	-0.688	-0.753	0.336	0.335	0.339
		T5	meteor + parent	0.161	0.169	0.152	-0.891	-0.861	-0.922	0.345	0.342	0.348
	NLPO	T5	bleu	0.062	0.065	0.059	-1.077	-1.057	-1.097	0.235	0.236	0.233
		T5	sacrebleu	0.085	0.088	0.083	-0.945	-0.917	-0.972	0.314	0.315	0.313
		T5	meteor	0.102	0.108	0.097	-1.044	-1.009	-1.079	0.329	0.328	0.330
		T5	parent	0.159	0.166	0.152	-0.710	-0.675	-0.745	0.357	0.351	0.363
		T5	meteor + parent	0.166	0.175	0.158	-0.704	-0.668	-0.740	0.365	0.362	0.368
	Supervised	T5		0.457	0.535	0.377	0.204	0.327	0.081	0.583	0.631	0.534
	Supervised + PPO	T5	bleu	0.473	0.548	0.395	0.200	0.323	0.078	0.590	0.638	0.542
		T5	sacrebleu	0.474	0.557	0.389	0.209	0.340	0.077	0.573	0.620	0.525
		T5	meteor	0.468	0.541	0.392	0.203	0.325	0.082	0.590	0.638	0.542
		T5	parent	0.469	0.547	0.388	0.175	0.300	0.050	0.595	0.641	0.549
		T5	meteor + parent	0.473	0.547	0.392	0.192	0.314	0.069	0.595	0.642	0.549
	Supervised + NLPO	T5	bleu	0.475	0.548	0.399	0.208	0.330	0.085	0.593	0.639	0.546
		T5	sacrebleu	0.475	0.557	0.392	0.208	0.335	0.081	0.577	0.625	0.529
		T5	meteor	0.468	0.541	0.392	0.201	0.322	0.079	0.594	0.641	0.546
T5		parent	0.474	0.550	0.392	0.192	0.315	0.068	0.596	0.643	0.550	
T5		meteor + parent	0.471	0.546	0.393	0.204	0.326	0.081	0.592	0.640	0.544	

Table C.24: **ToTTo test evaluation:** Table shows lexical, semantic and factual correctness metric scores of algorithms with different reward functions on hold-out test set. Without supervised pre-training, both PPO and NLPO results in sub-optimal solutions, with NLPO better than PPO. With supervised pre-training, PPO and NLPO achieve better scores across all metrics showing RL fine-tuning is beneficial. Most importantly, RL fine-tuned models produce more factually consistent text as seen in higher PARENT scores. Another observation, fine-tuning with a task-specific metric PARENT is better than training on task-agnostic lexical rewards

Algorithm	Unique N	Coherence			Correctness		
		Value	Alpha	Skew	Value	Alpha	Skew
Zero Shot	25	1.63	0.718	1.642	1.93	0.503	1.946
PPO+Supervised	24	4.57	0.221	4.579	4.48	0.098	4.483
PPO	26	2.75	0.427	2.753	3.23	0.214	3.227
NLPO	28	2.25	0.401	2.247	2.61	0.419	2.613
Supervised	24	4.59	0.173	4.592	4.54	0.189	4.537
NLPO+Supervised	26	4.58	0.244	4.601	4.57	0.144	4.581

Table C.25: **ToTTo Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorff’s alpha showing inter-annotator agreement, and Skew. For each model a total of 50 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 150 data points per algorithm.

Group 1	Group 2	Coherence		Correctness	
		Diff (G2-G1)	<i>p-values</i>	Diff (G2-G1)	<i>p-values</i>
PPO	NLPO	-0.507	0.001	-0.613	0.001
PPO	NLPO+Supervised	1.827	0.001	1.340	0.001
PPO	Supervised	1.833	0.001	1.313	0.001
PPO	PPO+Supervised	1.813	0.001	1.253	0.001
PPO	Zero Shot	-1.120	0.001	-1.293	0.001
NLPO	NLPO+Supervised	2.333	0.001	1.953	0.001
NLPO	Supervised	2.340	0.001	1.927	0.001
NLPO	PPO+Supervised	2.320	0.001	1.867	0.001
NLPO	Zero Shot	-0.613	0.001	-0.680	0.001
NLPO+Supervised	Supervised	0.007	0.9	-0.027	0.009
NLPO+Supervised	PPO+Supervised	-0.013	0.009	-0.087	0.009
NLPO+Supervised	Zero Shot	-2.947	0.001	-2.633	0.001
Supervised	PPO+Supervised	-0.020	0.009	-0.060	0.009
Supervised	Zero Shot	-2.953	0.001	-2.607	0.001
PPO+Supervised	Zero Shot	-2.933	0.001	-2.547	0.001

Table C.26: **ToTTo Human Study Tukey Results:** Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding p -values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall p -values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.

Human Participant Study

Figure C.7 shows the ToTTo instructions, example, and interface used for the human evaluation experiments. We made small modifications to the original code release’s HTML renderer to make the tables display in our HITs. Tables C.25, C.26 show averaged results, annotator agreement, and the results of statistical significance tests to determine which models output better generations when rated by humans.

Qualitative Analysis

We show sample generations from each of the algorithms for three randomly picked prompts below.

Sample 1

Prompt: `<table> <cell> August 1, 1992 </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> <cell> Allentown <col_header> City </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> <cell> Allentown Fairgrounds Grandstand <col_header> Country </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> </table><page_title> Under My Spell Tour </page_title> <section_title> Tour dates </section_title> <table> <cell> August 1, 1992 <col_header> Date </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> <cell> Allentown <col_header> City </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> <cell> Allentown Fairgrounds Grandstand <col_header> Country </col_header> <col_header> North America </col_header> <col_header> Australasia </col_header> <col_header> Asia </col_header> <col_header> North America </col_header> </cell> </table>`

Zero-Shot: `table> cell> Under My Spell Tour /section_title> Tour dates /cell> cell> August 1, 1992 col_header> Date /col_header>`

PPO: Under My Spell Tour, August 1, 1992. August 1, 1992.

NLPO: Under My Spell Tour dates Under My Spell Tour dates Allentown Fairgrounds Grandstand August 1, 1992. Under My Spell Tour dates Under My Spell Tour dates Allentown Fairgrounds Grandstand August 1, 1992 table>

Supervised: The Under My Spell Tour began on August 1, 1992 at the Allentown Fairgrounds Grandstand in Allentown.

Supervised + PPO: On August 1, 1992, Under My Spell Tour was held at the Allentown Fairgrounds Grandstand in Allentown.

Supervised + NLPO: On August 1, 1992, Under My Spell Tour was held at the Allentown Fairgrounds Grandstand in Allentown.

Sample 2

Prompt: `<table> <cell> Philip IV the Fair, the Iron King (Philippe) <col_header> Name </col_header> </cell> <cell> 5 October 1285 <col_header> King from </col_header> </cell> <cell> King of France and of Navarre (Roi de France et de Navarre) <col_header> Title </col_header> </cell> </table><page_title> List of French monarchs </page_title> <section_title> House of Capet (987԰) </section_title> <table> <cell> Philip IV the Fair, the Iron King (Philippe) <col_header> Name </col_header> </cell> <cell> 5 October 1285 <col_header> King from </col_header> </cell> <cell> King of France and of Navarre (Roi de France et de Navarre) <col_header> Title </col_header> </cell> </table>`

Zero-Shot: `> </table><page_title> List of French monarchs <`

PPO: Philip IV the Fair, the Iron King (Philippe) 5 October 1285. Title: King of France and of Navarre (Roi de France et de Navarre)

NLPO: King (Philippe) Philip IV the Fair, the Iron King (Philippe) 5 October 1285

Supervised: Philip IV the Fair, the Iron King (Philippe) was King of France and Navarre from 1285.

Supervised + PPO: Philip IV the Fair, the Iron King was King of France and Navarre from 1285.

Supervised + NLPO: Philip IV the Fair, the Iron King (Philippe) was King of France and Navarre (Roi de France et de Navarre) in 1285.

Instructions (click to expand)

Thank you for your participation in this and other similar HITS!

Please take a moment to familiarize yourself with this new HIT by reading the instructions/examples, because things have changed a bit. Thanks again for your work!

In this HIT you will be presented with a table from Wikipedia, with a few cells highlighted in yellow. Along with the table, you will be provided some metadata like the title of the Wikipedia article/section. Based on this table, you will also be given a system generation, which aims to capture/summarize/describe the Your job is to rate the generation across 2 axes:

- Fluency/Grammaticality:** *Is the system's generation grammatical, easy-to-read, and fluent?*
- Correctness/Specificity:** *Does the generation correctly describe a fact from the table and does that fact come from the highlighted cells?*

You will be able to rate each of the three axes on a scale from 1 to 5, with 1 being the lowest/worst and 5 the highest/best. The specific scales are:

- Fluency/Grammaticality:**
 - 5/5 (excellent):** The generation is grammatical and fluent.
 - 4/5 (good):** The sentence largely makes sense, but there are some small grammar issues/out-of-place words that don't make for the best writing.
 - 3/5 (okay):** The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.
 - 2/5 (poor):** Even though I can kind-of tell the meaning, it's difficult to read this unnatural sentence.
 - 1/5 (terrible):** The generation has severe errors in grammaticality/is almost or completely unreadable.
- Correctness/Specificity:**
 - 5/5 (correct and based on the highlighted cells):** The generation correctly describes the information conveyed in the highlighted cells.
 - 4/5 (mostly reasonable):** The generation mostly describes the information in the highlighted cells with only small deviations.
 - 3/5 (neutral):** The generation is somewhat plausible/relevant, but it's not as specific to the highlighted cells or correct as it could be.
 - 2/5 (mostly unreasonable):** I see why this could be generated given the table/cells, but it doesn't make much sense.
 - 1/5 (wrong/nonsense/irrelevant):** The generation doesn't seem to apply to the cells/tables at all, or doesn't make any sense.

Notes:

- For Fluency/Grammaticality, don't worry about correctness!* There can be grammatical sentences that do not describe the associated table, and vice versa (see the examples).
- For Correctness/Specificity, consider both the correctness of the statement given the table, and also its specificity to the highlighted cells;* don't give 5/5 if the generation applies better to unhighlighted cells.
- For Correctness/Specificity, it's okay if the generation references the metadata like the title;* --- points should be deducted for "specificity" if there are other table cells that are referenced more directly.
- A handful of tables are quite large! Still apply the same rating criteria, even if the tables have many rows.

Example 3:
Table from Wikipedia:

Party	Candidate	Votes	%
Democratic	Edwin M. Capps	1,714	52.3
Republican	Daniel C. Reed	1,493	45.6
Socialist Labor	John Helpingstine	70	2.1
Total votes		3,277	100

System's generation (rate this!):

Daniel C. Reed is a candidate 45.6 percent of the 1899 mayoral election results in San Diego, California. 45.6 --- Daniel C. Reed 1899 mayoral election result. cell

- Fluency/Grammaticality: 2/5 Why?** It's possible to fill-in-the-gaps to make sense of things, but this is not a fluent sentence.
- Correctness/Specificity: 3/5 Why?** The sentence does seem to correctly copy the highlighted cells --- but it's not a correct description of the cells, it's just mindlessly copying.

Example 2:

Table from Wikipedia:

No.	Date	Venue	Cap	Opponent	Score	Result	Competition
1	28 March 2009	Santiago Bernabéu Stadium, Madrid, Spain	2	Turkey	1–0	1–0	2010 FIFA World Cup qualification
2	12 August 2009	Philip II Arena, Skopje, Macedonia	8	North Macedonia	2–2	3–2	Friendly
3	5 September 2009	Estadio Riazor, A Coruña, Spain	9	Belgium	3–0	5–0	2010 FIFA World Cup qualification
4	14 October 2009	Bilino Polje, Zenica, Bosnia and Herzegovina	12	Bosnia and Herzegovina	1–0	5–2	
5	13 June 2016	Stadium Municipal, Toulouse, France	78	Czech Republic	1–0	1–0	UEFA Euro 2016

System's generation (rate this!):

Piqué scored an international goal at Stadium Municipal.

- Fluency/Grammaticality: 4/5 Why?** The sentence is grammatically correct, though the term "international goal" is a bit awkward --- "a goal on the international stage" would have been better.
- Correctness/Specificity: 3/5 Why?** It might be correct, but the focus of the sentence is the stadium where the player has scored, but that cell is not highlighted.

Table from Wikipedia:

\$prompt

System's generation (rate this!):

\$(machine_completion)

Coherence/Quality: 3/5

Is the system's generation grammatical, easy-to-read, and fluent?



The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.

Correctness/Specificity: 3/5

Does the generation correctly describe a fact from the table, and does that fact come from the highlighted cells? (wrong/nonsense/irrelevant)



The generation is somewhat plausible/relevant, but it's not as specific to the highlighted cells or correct as it could be.

(Optional) Please let us know if anything was unclear, if you experienced any issues, or if you have any other feedback for us.

Submit

Figure C.7: ToTTo Human Study Setup: Instructions, two examples, and interface for the ToTTo table description task.

Sample 3

Prompt: <table> <cell> Benjamin A. Smith II (D) <col_header> Vacator </col_header> </cell> <cell> Successor elected November 6, 1962. <col_header> Reason for change </col_header> </cell> <cell> Ted Kennedy (D) <col_header> Successor </col_header> </cell> <cell> November 7, 1962 <col_header> Date of successor's formal installation </col_header> </cell> </table><page_title> 87th United States Congress </page_title> <section_title> Senate </section_title> > <table> <cell> Benjamin A. Smith II (D) <col_header> Vacator </col_header> </cell> <cell> Successor elected November 6, 1962. <col_header> Reason for change </col_header> </cell> <cell> Ted Kennedy (D) <col_header> Successor </col_header> </cell> <cell> November 7, 1962 <col_header> Date of successor's formal installation </col_header> </cell> </table>

Zero-Shot: cell> Ted Kennedy (D) col_header> Vacator /col_header> /cell> cell> November 7, 1962 col_header> Reason for change

PPO: Ted Kennedy (D) succeeding November 6, 1962 as Vacator of the 87th United States Congress.

NLPO: Ted Kennedy (D) Successor elected November 6, 1962. table> cell Benjamin A. Smith II (D) col_header Reason for change: Date of successor's formal installation November 7, 1962.

Supervised: Benjamin A. Smith II (D) served as senate until November 6, 1962 which was later served by Ted Kennedy (D) from November 7, 1962.

Supervised + PPO: Benjamin A. Smith II (D) served until November 6, 1962 and Ted Kennedy (D) succeeded him from November 7, 1962.

Supervised + NLPO: Benjamin A. Smith II (D) served until November 6, 1962 and Ted Kennedy (D) succeeded him from November 7, 1962.

C.1.7 NarrativeQA

Setup NarrativeQA [296] deals with task of generating answers to questions about a given story. For training RL methods, we consider 2 traditional lexical rewards namely Rouge Combined and Rouge-L-Max. We chose T5-base as the base LM since it has been shown to do well at question answering in prior work [309]. We note that the supervised models we use are trained on the UnifiedQA dataset, which contains other QA datasets, and is shown by [309] to outperform supervised fine-tuning only on NarrativeQA. Hyperparams for our models can be found in Table C.27.

Results and Discussion

Tables C.28 and C.29 presents our benchmarking results with 2 reward functions along with supervised baseline performances on the NarrativeQA test set. Similar to other methods, our main finding is that warm-started initial policies are crucial for learning to generate answers that successfully use the input context.

Qualitative Results

We show sample generations from each of the algorithms for three randomly picked prompts below.

Sample 1

Prompt: who is mark hunter? mark hunter (slater), a high school student in a sleepy suburb of phoenix, arizona, starts an fm pirate radio station that broadcasts from the basement of his parents' house. mark is a loner, an outsider, whose only outlet for his teenage angst and aggression is his unauthorized radio station. his pirate station's theme song is "everybody knows" by leonard cohen and there are glimpses of cassettes by such alternative musicians as the jesus and mary chain, camper van beethoven, primal scream, soundgarden, ice-t, bad brains, concrete blonde, henry rollins, and the pixies. by day, mark is seen as a loner, hardly talking to anyone around him; by night, he expresses his outsider views about what is wrong with american society. when he speaks his mind about what is going on at his school and in the community, more and more of his fellow students tune in to hear his show.nobody knows the true identity of "hard harry" or "happy harry hard-on," as mark refers to himself, until nora diniro (mathis), a fellow student, tracks him down and confronts him the day

Model Params	value
ppo/nlpo	steps per update: 5120 total number of steps: 512000 batch size: 64 epochs per update: 5 learning rate: 0.000002 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 1.0 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 50 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
supervised+ ppo (or nlpo)	steps per update:2560 total number of steps: 512000 batch size: 64 epochs per update: 5 learning rate: 0.0000005 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 50 value function coeff: 0.5 top mask ratio: 0.9 target update iterations: 20
decoding	num beams: 4 max new tokens: 50
tokenizer	padding side: left truncation side: right max length: 512

Table C.27: NarQA Hyperparams: Table shows a list of all hyper-parameters and their settings

after a student named malcolm commits suicide after harry attempts to reason with him. the radio show becomes increasingly popular and influential after harry confronts the suicide head-on, exhorting his listeners to do something about their problems instead of surrendering to them through suicide. at the crescendo of his yelled speech, an overachieving student named paige woodward (who has been a constant listener) jams her various medals and accolades into a microwave and turns it on. she then sits, watching the awards cook until the microwave explodes, injuring her. while this is happening, other students act out in cathartic release. eventually, the radio show causes so much trouble in the community that the fcc is called in to investigate. during the fracas, it is revealed that the school's principal (annie ross) has been expelling "problem students," namely, students with below-average standardized test scores, in an effort to boost the district's test scores while still keeping their names on the rolls (a criminal offense) in order to retain government funding. realizing he has started something huge, mark decides it is up to him to end it. he dismantles his radio station and attaches it to his mother's old jeep, creating a mobile transmitter so his position can't be triangulated. pursued by the police and the fcc, nora drives the jeep around while mark broadcasts. the harmonizer he uses to

Appendix C RL4LMs

Tasks	Alg	Reward Function	LM	Lexical and Semantic Metrics							
				Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Rouge-LMax	Meteor	BLEU	BertScore
NarQA	Zero Shot		T5	0.095	0.022	0.084	0.084	0.117	0.095	0.009	0.835
	PPO	Rouge Combined	T5	0.101	0.025	0.088	0.088	0.122	0.099	0.01	0.837
		Rouge-L Max	T5	0.099	0.025	0.087	0.087	0.122	0.099	0.01	0.835
	NLPO	Rouge Combined	T5	0.097	0.023	0.085	0.085	0.118	0.098	0.009	0.836
		Rouge-L Max	T5	0.102	0.026	0.089	0.089	0.124	0.1	0.01	0.837
	Supervised		T5	0.378	0.190	0.367	0.367	0.581	0.099	0.209	0.931
	Supervised + PPO	Rouge Combined	T5	0.38	0.177	0.371	0.371	0.585	0.09	0.229	0.931
		Rouge-L Max	T5	0.368	0.18	0.36	0.36	0.585	0.083	0.239	0.931
	Supervised + NLPO	Rouge Combined	T5	0.398	0.21	0.393	0.373	0.589	0.096	0.24	0.971
		Rouge-L Max	T5	0.381	0.194	0.383	0.383	0.588	0.093	0.243	0.932

Table C.28: **Evaluation of NarrativeQA (Lexical and Semantic Metrics)**: Reference Metrics, supervised is based on UnifiedQA [309].

Tasks	Alg	Reward Function	LM	Diversity Metrics							
				MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂	Mean Output Length
NarQA	Zero Shot		T5	0.415	0.026	0.097	9.641	13.468	1880	11495	31.688
	PPO	Rouge Combined	T5	0.462	0.03	0.125	9.759	13.789	2522	17806	32.352
		Rouge-L Max	T5	0.439	0.029	0.119	9.653	13.618	2292	15816	31.479
	NLPO	Rouge Combined	T5	0.418	0.025	0.096	9.652	13.528	1816	10980	32.117
		Rouge-L Max	T5	0.445	0.029	0.118	9.776	13.75	2181	14569	31.555
	Supervised		T5	0.609	0.156	0.534	9.807	13.657	3250	14995	4.923
	Supervised + PPO	Rouge Combined	T5	0.640	0.174	0.559	10.132	13.547	3326	13785	4.353
		Rouge-L Max	T5	0.641	0.187	0.576	10.201	13.452	3287	12436	3.913
	Supervised + NLPO	Rouge Combined	T5	0.679	0.185	0.595	10.304	13.694	3371	15067	4.728
		Rouge-L Max	T5	0.645	0.187	0.59	10.2	13.397	3287	12171	3.889

Table C.29: **Evaluation of NarrativeQA (Diversity Metrics)**: Reference Metrics, supervised is based on UnifiedQA [309].

disguise his voice breaks, and with no time left to fix it, mark decides to broadcast his final message as himself. they finally drive up to the crowd of protesting students, and mark tells them that the world belongs to them and that they should make their own future. the police step in and arrest mark and nora. as they are taken away, mark reminds the students to "talk hard." as the film ends, the voices of other students (and even one of the teachers) speak as intros for their own independent stations, which can be heard broadcasting across the country.

Zero-Shot: hunter (slater) is a high school student in a sleepy suburb of phoenix, arizona. he starts an fm pirate radio station that broadcasts from the basement of his

References: he is a high school student in phoenix 2) loner and outsider student with a radio station

PPO: hunter (slater), a high school student in a sleepy suburb of phoenix, arizona, starts an fm pirate radio station that broadcasts from the basement of his parents' house

NLPO: hunter (slater), a high school student in a sleepy suburb of phoenix, arizona, starts an fm pirate radio station that broadcasts from the basement of his parents' house

Supervised: a high school student

Supervised + PPO: a high school student

Supervised + NLPO: a high school student

Sample 2

Prompt: what was the principle doing with the problem students? mark hunter (slater), a high school student in a sleepy suburb of phoenix, arizona, starts an fm pirate radio station

that broadcasts from the basement of his parents' house. mark is a loner, an outsider, whose only outlet for his teenage angst and aggression is his unauthorized radio station. his pirate station's theme song is "everybody knows" by leonard cohen and there are glimpses of cassettes by such alternative musicians as the jesu and mary chain, camper van beethoven, primal scream, soundgarden, ice-t, bad brains, concrete blonde, henry rollins, and the pixies. by day, mark is seen as a loner, hardly talking to anyone around him; by night, he expresses his outsider views about what is wrong with american society. when he speaks his mind about what is going on at his school and in the community, more and more of his fellow students tune in to hear his show. nobody knows the true identity of "hard harry" or "happy harry hard-on," as mark refers to himself, until nora diniro (mathis), a fellow student, tracks him down and confronts him the day after a student named malcolm commits suicide after harry attempts to reason with him. the radio show becomes increasingly popular and influential after harry confronts the suicide head-on, exhorting his listeners to do something about their problems instead of surrendering to them through suicide. at the crescendo of his yelled speech, an overachieving student named paige woodward (who has been a constant listener) jams her various medals and accolades into a microwave and turns it on. she then sits, watching the awards cook until the microwave explodes, injuring her. while this is happening, other students act out in cathartic release. eventually, the radio show causes so much trouble in the community that the fcc is called in to investigate. during the fracas, it is revealed that the school's principal (annie ross) has been expelling "problem students," namely, students with below-average standardized test scores, in an effort to boost the district's test scores while still keeping their names on the rolls (a criminal offense) in order to retain government funding. realizing he has started something huge, mark decides it is up to him to end it. he dismantles his radio station and attaches it to his mother's old jeep, creating a mobile transmitter so his position can't be triangulated. pursued by the police and the fcc, nora drives the jeep around while mark broadcasts. the harmonizer he uses to disguise his voice breaks, and with no time left to fix it, mark decides to broadcast his final message as himself. they finally drive up to the crowd of protesting students, and mark tells them that the world belongs to them and that they should make their own future. the police step in and arrest mark and nora. as they are taken away, mark reminds the students to "talk hard." as the film ends, the voices of other students (and even one of the teachers) speak as intros for their own independent stations, which can be heard broadcasting across the country.

Zero-Shot: mark hunter (slater) starts an fm pirate radio station that broadcasts from the basement of his parents' house. by day, mark is seen as a loner, an outsider, whose only outlet

References: 1) expelling them. 2) expelling the students

PPO: mark hunter (slater) starts an fm pirate radio station that broadcasts from the basement of his parents' house. mark is a loner, an outsider, whose only outlet for his teenage ang

NLPO: mark hunter (slater) starts an fm pirate radio station that broadcasts from the basement of his parents' house. mark is a loner, an outsider, whose only outlet for his teenage ang

Supervised: he was taking them to a psychiatric hospital.

Supervised + PPO: addressing them

Supervised + NLPO: he was expelling them

Sample 3

Prompt: what happens to the people maskull meets in his travels to various lands? maskull, a man longing for adventures, accepts an invitation from krag, an acquaintance of his friend nightspore, to travel to tormance after a seance. the three set off in a crystal ship from an abandoned observatory in scotland but maskull awakens to find himself alone on tormance. in every land he passes through he usually meets only one or two persons; these meetings often (though not always) end in the death of those he meets, either at his own hand or by that of another. he learns of his own impending death, meets krag again, and dies shortly after learning that he is in fact nightspore himself. the book concludes with a final revelation from krag (who claims to be known on earth as "pain") to nightspore about the origin of the universe. the author turns out to support a variation of the doctrine of the demiurge, somewhat similar to that propounded by some gnostics. all of the characters and lands are types used to convey the author's critique of several philosophical systems. on tormance, most such viewpoints or ways of life are accompanied by corresponding new bodily sense organs or modifications of the same, thus each distinct weltanschauung landscape has its corresponding sensorium.

Zero-Shot: what happens to the people maskull meets in his travels to various lands? n maskull, a man longing for adventures, accepts an invitation from krag to travel to tormance after

References: 1) they usually die by his hand or anothers. 2) they die
PPO: what happens to the people maskull meets in his travels to various lands? n maskull, a man longing for adventures, accepts an invitation from krag to travel to tormance after
NLPO: maskull meets krag, an acquaintance of nightspore, to travel to tormance after a seance. maskull awakens to find himself alone on tormance. in every land
Supervised: they die either from his own hand or from another person's death.
Supervised + PPO: they end up dying either at his own hand or by another's.
Supervised + NLPO: they end up dying either at his own hand or by another's

C.1.8 Neural Machine Translation

Setup We benchmark on two datasets: 1) IWSLT-17 [310] contains transcriptions of TED talks in many languages. 2) WMT-16: We pick two languages, English and German, and frame this task similarly to other machine translation tasks—requiring the models to translate from English to German. We train models on 4 rewards: SacreBLEU, chRF, TER, and BertScore.

Results and Discussion

Tables C.31, C.32 presents our benchmarking results with 4 reward functions along with supervised baseline performances on test set. Our main finding is that NLPO + Supervised performs better than PPO and supervised models.

Model Params	value
supervised	batch size: 64 epochs: 5 learning rate: 0.00001 learning rate scheduler: constant weight decay: 0.1
ppo/nlpo	steps per update: 5120 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.0.000001 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 10 value function coeff: 0.5 top mask ratio: 0.5 target update iterations: 20
supervised+ ppo (or nlpo)	steps per update:2560 total number of steps: 256000 batch size: 64 epochs per update: 5 learning rate: 0.0000005 entropy coefficient: 0.0 initial kl coeff: 0.001 target kl: 0.2 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 10 value function coeff: 0.5 top mask ratio: 0.5 target update iterations: 20
decoding	num beams: 4 length penalty: 0.6 max new tokens: 128
tokenizer	padding side: left truncation side: right max length: 128

Table C.30: NMT Hyperparams: Table shows a list of all hyper-parameters and their settings

Appendix C RL4LMs

Datasets			Lexical and Semantic Metrics										
Alg	LM	Reward Function	Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	BLEU	SacreBLEU	chRf	TER	BertScore	
WMT16	Zero-Shot	T5	0.635	0.414	0.591	0.591	0.483	0.294	0.348	0.613	0.543	0.882	
	PPO	T5	SacreBLEU	0.636	0.415	0.591	0.591	0.482	0.294	0.348	0.614	0.539	0.882
		T5	chRF	0.635	0.414	0.591	0.591	0.481	0.291	0.346	0.612	0.540	0.882
		T5	TER	0.638	0.416	0.595	0.594	0.484	0.294	0.350	0.616	0.534	0.883
		T5	BertScore	0.637	0.417	0.593	0.593	0.479	0.294	0.347	0.613	0.534	0.882
	NLPO	T5	SacreBLEU	0.635	0.415	0.592	0.592	0.484	0.297	0.352	0.615	0.542	0.882
		T5	chRF	0.634	0.413	0.59	0.59	0.481	0.291	0.345	0.612	0.540	0.882
		T5	TER	0.633	0.412	0.59	0.59	0.477	0.286	0.341	0.608	0.540	0.881
		T5	BertScore	0.622	0.397	0.58	0.581	0.458	0.269	0.323	0.591	0.546	0.876
	Supervised	T5		0.635	0.411	0.590	0.590	0.482	0.294	0.350	0.617	0.540	0.882
	Supervised + PPO	T5	SacreBLEU	0.640	0.416	0.595	0.595	0.487	0.298	0.355	0.620	0.533	0.883
		T5	chRF	0.640	0.416	0.596	0.596	0.486	0.298	0.354	0.621	0.532	0.883
		T5	TER	0.637	0.414	0.594	0.594	0.483	0.295	0.352	0.618	0.533	0.882
		T5	BertScore	0.637	0.413	0.593	0.594	0.482	0.294	0.350	0.616	0.533	0.882
	Supervised + NLPO	T5	SacreBLEU	0.642	0.419	0.596	0.596	0.497	0.297	0.355	0.621	0.533	0.888
		T5	chRF	0.636	0.412	0.592	0.592	0.492	0.293	0.349	0.617	0.534	0.886
T5		TER	0.637	0.414	0.594	0.594	0.491	0.292	0.349	0.615	0.531	0.886	
T5		BertScore	0.64	0.417	0.598	0.598	0.499	0.287	0.349	0.62	0.538	0.887	
IWSLT2017	Zero-Shot	T5	0.619	0.386	0.588	0.587	0.445	0.254	0.308	0.577	0.573	0.870	
	PPO	T5	SacreBLEU	0.621	0.383	0.587	0.587	0.448	0.243	0.296	0.575	0.583	0.869
		T5	chRF	0.622	0.385	0.590	0.590	0.448	0.248	0.301	0.578	0.575	0.870
		T5	TER	0.623	0.384	0.591	0.591	0.443	0.246	0.303	0.572	0.568	0.869
		T5	BertScore	0.533	0.326	0.507	0.507	0.321	0.143	0.174	0.406	0.573	0.839
	NLPO	T5	SacreBLEU	0.624	0.385	0.59	0.59	0.45	0.245	0.299	0.578	0.578	0.87
		T5	chRF	0.624	0.386	0.59	0.59	0.451	0.248	0.302	0.581	0.576	0.87
		T5	TER	0.622	0.384	0.59	0.59	0.443	0.246	0.303	0.573	0.57	0.869
		T5	BertScore	0.611	0.377	0.58	0.58	0.425	0.239	0.291	0.555	0.573	0.866
	Supervised	T5		0.638	0.400	0.610	0.609	0.461	0.280	0.337	0.593	0.538	0.878
	Supervised + PPO	T5	SacreBLEU	0.640	0.407	0.610	0.610	0.465	0.277	0.332	0.596	0.542	0.877
		T5	chRF	0.639	0.406	0.609	0.609	0.464	0.277	0.331	0.596	0.543	0.877
		T5	TER	0.637	0.406	0.609	0.609	0.457	0.274	0.331	0.589	0.535	0.876
		T5	BertScore	0.612	0.381	0.585	0.585	0.418	0.240	0.291	0.548	0.559	0.867
	Supervised + NLPO	T5	SacreBLEU	0.641	0.418	0.614	0.614	0.474	0.289	0.343	0.597	0.535	0.877
		T5	chRF	0.643	0.418	0.621	0.621	0.464	0.291	0.345	0.596	0.539	0.877
T5		TER	0.639	0.419	0.621	0.621	0.471	0.289	0.346	0.593	0.535	0.877	
T5		BertScore	0.633	0.401	0.606	0.606	0.448	0.267	0.323	0.580	0.537	0.875	

Table C.31: **WMT-16 and IWSLT test evaluation - lexical and semantic:** Table shows lexical, semantic metrics for RL algorithms with different reward functions bench-marked against supervised baseline models

C.1 Experimental Details

Tasks	Diversity Metrics											
	Alg	Reward Function	LM	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂	Mean Output Length	
WMT16	Zero-Shot	T5		0.732	0.193	0.675	10.100	14.561	7290	33691	20.533	
	PPO	T5	SacreBLEU	0.738	0.198	0.687	10.166	14.613	7503	34140	20.375	
		T5	chRF	0.738	0.196	0.687	10.175	14.611	7376	34116	20.337	
		T5	TER	0.736	0.196	0.683	10.132	14.588	7447	33977	20.356	
		T5	BertScore	0.736	0.195	0.685	10.129	14.574	7272	33477	20.035	
	NLPO	T5	SacreBLEU	0.735	0.193	0.68	10.125	14.592	7395	34276	20.672	
		T5	chRF	0.738	0.196	0.686	10.164	14.606	7399	34056	20.351	
		T5	TER	0.74	0.2	0.694	10.204	14.63	7522	34234	20.151	
		T5	BertScore	0.739	0.2	0.698	10.194	14.608	7203	33169	19.482	
	Supervised	T5		0.729	0.190	0.669	10.048	14.530	7205	33430	20.622	
	Supervised + PPO	T5	SacreBLEU	0.732	0.191	0.674	10.080	14.552	7222	33723	20.605	
		T5	chRF	0.735	0.192	0.677	10.093	14.569	7319	33923	20.586	
		T5	TER	0.732	0.192	0.676	10.079	14.553	7265	33635	20.441	
		T5	BertScore	0.732	0.192	0.677	10.082	14.550	7187	33385	20.305	
	Supervised + NLPO	T5	SacreBLEU	0.734	0.191	0.675	10.089	14.568	7308	33941	20.686	
		T5	chRF	0.735	0.194	0.681	10.112	14.571	7372	33814	20.348	
		T5	TER	0.737	0.194	0.682	10.105	14.566	7243	33482	20.159	
		T5	BertScore	0.737	0.227	0.742	10.042	14.179	5438	22574	12.63	
	IWSLT2017	Zero-Shot	T5		0.662	0.097	0.4700	9.276	14.526	8312	52947	18.739
		PPO	T5	SacreBLEU	0.657	0.095	0.464	9.230	14.498	8285	53000	19.069
T5			chRF	0.660	0.096	0.468	9.253	14.526	8243	53142	18.912	
T5			TER	0.659	0.097	0.474	9.244	14.536	8129	51914	18.268	
T5			BertScore	0.673	0.120	0.541	9.288	14.388	6642	37267	11.602	
NLPO		T5	SacreBLEU	0.656	0.094	0.463	9.207	14.483	8240	52822	19.043	
		T5	chRF	0.658	0.095	0.464	9.233	14.502	8230	53167	19.073	
		T5	TER	0.661	0.098	0.476	9.271	14.552	8223	52438	18.344	
		T5	BertScore	0.667	0.102	0.491	9.31	14.576	8134	50740	17.162	
Supervised		T5		0.655	0.095	0.467	9.210	14.492	7970	51430	18.440	
Supervised + PPO		T5	SacreBLEU	0.654	0.094	0.461	9.176	14.467	8061	51840	18.803	
		T5	chRF	0.656	0.094	0.464	9.202	14.497	8054	52198	18.794	
		T5	TER	0.658	0.097	0.475	9.239	14.529	7969	51255	18.048	
		T5	BertScore	0.665	0.102	0.495	9.270	14.524	7495	47629	16.051	
Supervised + NLPO		T5	SacreBLEU	0.655	0.094	0.461	9.182	14.469	8073	51797	18.777	
		T5	chRF	0.657	0.095	0.466	9.212	14.503	8074	52056	18.642	
		T5	TER	0.658	0.096	0.473	9.231	14.521	8025	51389	18.133	
		T5	BertScore	0.661	0.098	0.482	9.26	14.547	7852	50672	17.496	

Table C.32: **WMT-16 and IWSLT test evaluation - diversity metrics** Table shows diversity metrics for RL algorithms with different reward functions bench-marked against supervised baseline models

Qualitative Results

We show sample generations from each of the algorithms for three randomly picked prompts from IWSLT below.

Sample 1

Prompt: translate English to German: Several years ago here at TED, Peter Skillman introduced a design challenge called the marshmallow challenge.

Zero-Shot: Vor einigen Jahren hat Peter Skillman hier bei TED eine Design-Herausforderung namens Marshmallow Challenge vorgestellt.

PPO: Vor einigen Jahren, hier bei TED, stellte Peter Skillman einen Design-Wettbewerb namens "Die Marshmallow-Herausforderung" vor

NLPO: Vor einigen Jahren hier bei TED fÄijhrte Peter Skillman eine Design-Herausforderung namens Marshmallow Challenge ein.

Supervised: Vor einigen Jahren hier bei TED fÄijfchrte Peter Skillman eine Design-Herausforderung namens Marshmallow Challenge ein.

Supervised + PPO: Vor einigen Jahren hier bei TED, fÄijhrte Peter Skillman eine Design-Herausforderung namens Marshmallow-Herausforderung ein.

Supervised + NLPO: Vor einigen Jahren hier bei TED fÄijchrte Peter Skillman eine Design-Herausforderung ein, die Marshmallow-Herausforderung.

Sample 2

Prompt: translate English to German: They spend the majority of their time assembling the sticks into ever-growing structures.

Zero-Shot: Sie verbringen den grÄüÄsten Teil ihrer Zeit damit, die StÄücke in immer wachsende Strukturen zu montieren.

References: Sie investieren einen GroÄsteil ihrer Zeit in den Zusammenbau stÄndig wachsender Strukturen

PPO: Sie verbringen den GroÄsteil ihrer Zeit damit, die StÄücke in immer grÄüÄser werdende Strukturen zu montieren.

NLPO: Sie verbringen den GroÄsteil ihrer Zeit, die StÄücke in immer wachsende Strukturen zu montieren.

Supervised: Sie verbringen den GroÄsteil ihrer Zeit damit, die StÄücke in immer wachsende Strukturen zu montieren.

Supervised + PPO: Sie verbringen den GroÄsteil ihrer Zeit damit, die StÄücke in immer wachsende Strukturen zu montieren.

Supervised + NLPO: Sie verbringen den GroÄsteil ihrer Zeit damit, die StÄücke in immer wachsende Strukturen zu montieren.

Sample 3

Prompt: translate English to German: And of course there are teams that have a lot more "ta-da" structures, and among the best are recent graduates of kindergarten.

Zero-Shot: Und natÄijrlich gibt es Teams, die viel mehr "ta-da" Strukturen haben, und zu den besten gehÄüren jÄijngste Absolventen des Kindergartens.

References: Und natÄijrlich gibt es auch Teams die viel mehr "ta-da"-Gebilde haben und unter den Besten sind frische Absolventen des Kindergartens.

PPO: Und natÄijrlich gibt es Teams, die viel mehr "ta-da" Strukturen, und unter den besten sind jÄijngste Absolventen des Kindergartens.

NLPO: Und natÄijrlich gibt es Teams, die viel mehr "ta-da" Strukturen haben, und unter den besten sind jÄijngste Absolventen des Kindergartens

Supervised: Und natÄijrlich gibt es Teams, die viel mehr "ta-da"-Strukturen haben, und unter den besten sind jÄijngste Absolventen des Kindergartens.

Supervised + PPO: Und natÄijrlich gibt es Teams, die viel mehr "ta-da"-Strukturen haben, und unter den besten sind jÄijngste Absolventen des Kindergartens.

Supervised + NLPO: Und natÄijlich gibt es Teams, die viel mehr "ta-da"-Strukturen haben, und unter den besten sind jÄijngste Absolventen des Kindergartens.

C.1.9 Dially Dialog

Setup We consider DailyDialog [297] as the test bed for the dialogue generation task. The dataset includes conversations written by human on various topics. In addition, each utterance contains labels

of intent and emotional information. For simplicity, we focus only on generating the next utterance, given the dialogue context. We chose a context window of size 5, which results in 35k training, 3k and 3k utterances. The input to the model is dialogue history in which utterances are concatenated using a <EOU> token. We picked GPT-2 as the LM as they are more suited for text continuation than encoder-decoder LMs. For a fair comparison, we use top-k sampling with $k = 20$ as the decoding method for all methods. For RL methods, we use a linear combination of meteor score and intent match score (whether the generated text’s intent matches with the reference’s intent) as the reward function. The coefficients for meteor and intent are chosen based on both lexical scores and intent accuracy on the validation set. For this purpose, we trained an intent classifier (fine-tuned RoBERTa [311]) that classifies given text into intent categories such as *inform*, *question*, *directive* and *commisive*, etc. Table C.33 provides a summary of hyperparameters and implementation details.

Results and Discussion

Tables C.34 and C.35 presents our benchmarking results of RL methods along with supervised baseline performances on test sets. Our main finding is that RL methods generally achieve better intent accuracy and automatic metric scores, in particular NLPO variants perform better than all other methods.

Human Participant Study

Figure C.8 shows the Daily Dialogue instructions and interface used for the human evaluation experiments. Tables C.36, C.37 show averaged results, annotator agreement, and the results of statistical significance tests to determine which models output better generations when rated by humans.

Model Params	value
ppo/nlpo	steps per update: 1280 total number of steps: 128000 batch size: 64 epochs per update: 5 learning rate: 0.000001 entropy coefficient: 0.0 initial kl coeff: 0.2 target kl: 0.5 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 20 value function coeff: 0.5 meteor coeff: 0.25 intent coeff: 0.75 top mask ratio: 0.9 target update iterations: 20
supervised+ ppo (or nlpo)	steps per update:1280 total number of steps: 64000 batch size: 64 epochs per update: 5 learning rate: 0.000001 entropy coefficient: 0.0 initial kl coeff: 0.2 target kl: 0.5 discount factor: 0.99 gae lambda: 0.95 clip ratio: 0.2 rollouts top k : 20 value function coeff: 0.5 meteor coeff: 0.5 0.25 intent coeff: 0.5 0.75 top mask ratio: 0.9 target update iterations: 20
decoding	top k: 20 min length: 2 max new tokens: 50
tokenizer	padding side: left truncation side: right max length: 128

Table C.33: **DailyDialog Hyperparams:** Table shows a list of all hyper-parameters and their settings

Tasks	Lexical and Semantic Metrics										
	Alg	Reward Function	LM	Rouge-1	Rouge-2	Rouge-L	Rouge-LSum	Meteor	SacreBLEU	BertScore	Intent Accuracy
Dialog	Zero Shot		GPT-2	0.157	0.012	0.131	0.131	0.191	0.066	0.854	0.427
	Supervised		GPT-2	0.162	0.020	0.138	0.138	0.186	0.064	0.855	0.437
	PPO	Meteor + Intent	GPT-2	0.168	0.012	0.142	0.142	0.221	0.085	0.861	0.474
	NLPO	Meteor + Intent	GPT-2	0.169	0.013	0.142	0.142	0.221	0.087	0.860	0.490
	Supervised + PPO	Meteor + Intent	GPT-2	0.169	0.021	0.144	0.144	0.198	0.071	0.857	0.455
	Supervised + NLPO	Meteor + Intent	GPT-2	0.171	0.020	0.146	0.146	0.205	0.074	0.858	0.454

Table C.34: **Evaluation of Daily Dialog:** Table shows lexical, semantic metrics for RL algorithms bench-marked against supervised baseline models

Tasks	Diversity Metrics										
	Alg	Reward Function	LM	MSTTR	Distinct ₁	Distinct ₂	H ₁	H ₂	Unique ₁	Unique ₂	Mean Output Length
Dialog	Zero Shot		GPT-2	0.608	0.055	0.316	7.787	11.831	1574	12327	18.685
	Supervised		GPT-2	0.635	0.065	0.342	8.051	12.119	1925	13952	18.919
	PPO	Meteor + Intent	GPT-2	0.581	0.058	0.310	7.653	11.437	1719	12156	18.538
	NLPO	Meteor + Intent	GPT-2	0.568	0.059	0.309	7.630	11.351	1718	11946	18.397
	Supervised + PPO	Meteor + Intent	GPT-2	0.626	0.068	0.348	8.056	12.015	1983	14170	18.829
	Supervised + NLPO	Meteor + Intent	GPT-2	0.624	0.070	0.349	8.044	11.990	2051	14213	18.763

Table C.35: **Evaluation of Daily Dialog:** Table shows diversity metrics for RL algorithms bench-marked against supervised baseline models

Algorithm	Unique N	Coherence			Quality		
		Value	Alpha	Skew	Value	Alpha	Skew
Zeroshot	31	3.84	0.225	4.181	3.2	0.125	3.352
NLPO	30	4.18	0.114	4.17	3.35	0.159	3.318
PPO	32	4.18	0.112	4.032	3.32	0.163	3.478
Supervised+PPO	31	3.99	0.148	4.133	3.48	0.166	3.58
Supervised+NLPO	31	4.13	0.186	3.953	3.58	0.178	3.597
Supervised	31	3.96	0.249	3.834	3.59	0.236	3.196

Table C.36: **DailyDialog Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorff’s alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, each resulting in 300 data points per algorithm.

Group 1	Group 2	Coherence		Quality	
		Diff (G2-G1)	<i>p-values</i>	Diff (G2-G1)	<i>p-values</i>
NLPO	PPO	-0.003	0.900	-0.030	0.900
NLPO	Supervised	-0.227	0.043	0.238	0.020
NLPO	Supervised+NLPO	-0.050	0.900	0.234	0.022
NLPO	Supervised+PPO	-0.194	0.013	0.127	0.803
NLPO	Zero Shot	-0.345	0.001	-0.154	0.655
PPO	Supervised	-0.224	0.049	0.268	0.010
PPO	Supervised+NLPO	-0.047	0.900	0.264	0.011
PPO	Supervised+PPO	-0.191	0.144	0.157	0.636
PPO	Zero Shot	-0.341	0.001	-0.124	0.822
Supervised	Supervised+NLPO	0.177	0.021	-0.003	0.900
Supervised	Supervised+PPO	0.033	0.900	-0.110	0.896
Supervised	Zero Shot	-0.117	0.645	-0.391	0.002
Supervised+NLPO	Supervised+PPO	-0.144	0.444	-0.107	0.009
Supervised+NLPO	Zero Shot	-0.294	0.002	-0.388	0.003
Supervised+PPO	Zero Shot	-0.151	0.390	-0.281	0.008

Table C.37: **DailyDialog Human Study Tukey Results:** Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding *p*-values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall *p*-values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.

Qualitative Analysis

We show sample generations from each of the algorithms for three randomly picked prompts below.

Sample 1

Prompt: Right . And China is the homeland of tea . <EOU> Yes , Chinese people love drinking tea so much . Some even claim they can't live without tea . <EOU> Do you know there are several catagories of Chinese tea ? <EOU> Yes , I believe there are green teas , black teas and scented teas . Any Others ? <EOU> Well , have you ever heard of Oulong tea and compressed tea ? <EOU>

References: Oh , yeah . Oulong tea is good for one's health . isn't it ?

Zero-Shot: I know that Oulong tea can be enjoyed with very few meals, and there are very few

PPO: I have, but it's not as strong as Oulong tea.

NLPO: I haven't. I don't have tea.

Supervised: Yes. I hear they do good. And I heard this Taijiquan Tea is very good

Supervised + PPO: Actually, I have had Oulong tea, but I haven't had any for awhile now.

Supervised + NLPO: No, I've never heard of that.

Sample 2

Prompt: I think I might have a dahl . <EOU> what's in a dahl ? <EOU> it's got chickpeas and vegetables in a spicy curry sauce with rice . <EOU> that sounds nice . Do you want to share some kebabs as well ? <EOU> ok . How about some lamb kebabs ? <EOU>

References: that's my favourite . Do you want to have some wine or beer ?

Zero-Shot: they're very salty

PPO: well you might want to take a peek.

NLPO: well I like that.

Supervised: ok, here they are. I'm so thirsty. I can't wait until the next bus to

Supervised + PPO: I've tried that a few times, but they're too dry and I don't want any of

Supervised + NLPO: oh, ok. That's a good choice. I'll take it.

Sample 3

Prompt: Really ? Have you had an auto-reverse break before ? <EOU> Yes , twice now . I think it's too high-tech , and so it's the first part that breaks . <EOU> Hmm . You know , the problem might be dust . Did you clean the unit often ? <EOU> Well . No , I didn't clean it. But still , I only had the last one two months , and it broke . I don't trust auto-reverse . <EOU> Alright . Well . We have many units without auto-reverse. Here is a good unit . Very reliable. It has an AM / FM band , built-in microphone , recording capability , and no auto-reverse . <EOU>

References: How much is it ?

Zero-Shot: Oh. So you don't have the AM / FM band?

PPO: You said AM / FM. Are you sure about that?

NLPO: Well. Well. I had no idea when it was, and didn't care.

Supervised: Good. Well, what else I need?

Supervised + PPO: But I think it lacks the CD player or its CD player accessories.

Supervised + NLPO: Cool! So what's the trouble with it?

Instructions (click to expand)

Thank you for your participation in this and other similar HITS!

Please take a moment to familiarize yourself with this new HIT by reading the instructions/examples, because things have changed a bit. Thanks again for your work!

In this HIT you will be presented with a dialogue consisting of a conversation between two people. You will also be given a system generation, which aims to contain the next line of the conversation. Your job is to rate the system generation, across 2 axes:

- **Fluency/Grammaticality:** *Is the system's generation grammatical, easy-to-read, and fluent?*
- **Quality/Coherence:** *Is the next utterance coherent, reasonable, and the type of thing a person might say, in the context of the dialogue history?*

You will rate each of the two axes on a scale from 1 to 5, with 1 being the lowest/worst and 5 the highest/best. The specific scales are:

- **Fluency/Grammaticality:**
 - **5/5 (excellent):** The generation is grammatical and fluent.
 - **4/5 (good):** The sentence largely makes sense, but there are some small grammar issues/out-of-place words that don't make for the best writing.
 - **3/5 (okay):** The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.
 - **2/5 (poor):** Even though I can kind-of tell the meaning, it's difficult to read this unnatural sentence.
 - **1/5 (terrible):** The generation has severe errors in grammaticality/is almost or completely unreadable.
- **Quality/Coherence:**
 - **5/5 (perfectly coherent, interesting):** The generated next utterance is very relevant and coherent with the dialogue context; a human might say this.
 - **4/5 (mostly relevant):** The generation is relevant but not perfect given the dialogue context.
 - **3/5 (neutral):** The generation is somewhat plausible/relevant.
 - **2/5 (mostly irrelevant):** I see why this could be generated but it doesn't make much sense.
 - **1/5 (wrong/nonsense/irrelevant):** The generation doesn't seem to apply to the dialogue at all or doesn't make any sense.

Dialogue context (read me first!):

Person 1: It doesn't work.

Person 2: What happened?

Person 1: I don't know.

Person 2: Did you call the repairman?

Person 1: Of course.

System's generation (rate this!):

Person 2: I'm sorry. They couldn't find the problem.

Fluency/Grammaticality:

3/5

Is the system's generation grammatical, easy-to-read, and fluent?

Bad Excellent



The grammar is okay and it's possible to read, but it definitely doesn't sound like a human wrote it.

Quality/Coherence: 3/5

Is the next utterance coherent, reasonable, and the type of thing a person might say, in the context of the dialogue history?

Bad Excellent



The generation is somewhat plausible/relevant.

Figure C.8: **DailyDialog Human Study Setup:** Instructions and interface for the Daily Dialogue task.

Bibliography

- [1] M. Abadi, “TensorFlow: Learning Functions at Scale”, *Proceedings of International Conference on Functional Programming*, 2016 (cit. on p. 1).
- [2] A. Paszke et al., *Pytorch: An Imperative Style, High-Performance Deep Learning Library*, *Advances in Neural Information Processing Systems* **32** (2019) (cit. on p. 1).
- [3] A. Krizhevsky, I. Sutskever and G. E. Hinton, *Imagenet Classification with Deep Convolutional Neural Networks*, *Communications of ACM* **60** (2017) (cit. on p. 1).
- [4] C. Farabet, C. Couprie, L. Najman and Y. LeCun, *Learning Hierarchical Features for Scene Labeling*, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35** (2012) (cit. on p. 1).
- [5] J. J. Tompson, A. Jain, Y. LeCun and C. Bregler, *Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation*, *Advances in Neural Information Processing Systems* **27** (2014) (cit. on p. 1).
- [6] C. Szegedy et al., “Going Deeper with Convolutions”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015 (cit. on p. 1).
- [7] G. Hinton et al., *Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of Four Research Groups*, *IEEE Signal Processing Magazine* **29** (2012) (cit. on p. 1).
- [8] T. Mikolov, A. Deoras, D. Povey and L. Burget, “Strategies for Training Large Scale Neural Network Language Models”, *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011 (cit. on p. 1).
- [9] T. Ciodaro, D. Deva, J. De Seixas and D. Damazio, “Online Particle Detection with Neural Networks Based on Topological Calorimetry Information”, *Journal of Physics: Conference Series*, vol. 368, 2012 (cit. on p. 1).
- [10] M. K. Leung, H. Y. Xiong, L. J. Lee and B. J. Frey, *Deep learning of the Tissue-Regulated Splicing Code*, *Bioinformatics* **30** (2014) (cit. on p. 1).
- [11] H. Y. Xiong et al., *The Human Splicing Code Reveals New Insights into the Genetic Determinants of Disease*, *Science* **347** (2015) (cit. on p. 1).
- [12] J. Ma, R. P. Sheridan, A. Liaw, G. E. Dahl and V. Svetnik, *Deep Neural Nets as a Method for Quantitative Structure–Activity Relationships*, *Journal of Chemical Information and Modeling* **55** (2015) (cit. on p. 1).

- [13] C. Adam-Bourdarios et al., “The Higgs Boson Machine Learning Challenge”, *NIPS Workshop on High-Energy Physics and Machine Learning*, 2015 (cit. on p. 1).
- [14] A. W. Senior et al., *Improved Protein Structure Prediction using Potentials from Deep Learning*, *Nature* **577** (2020) (cit. on p. 1).
- [15] J. Jumper et al., *Highly Accurate Protein Structure Prediction with AlphaFold*, *Nature* **596** (2021) (cit. on p. 1).
- [16] I. Goodfellow et al., “Generative Adversarial Nets”, *Advances in Neural Information Processing Systems*, vol. 27, 2014 (cit. on p. 1).
- [17] D. P. Kingma and M. Welling, *Auto-Encoding Variational Bayes*, arXiv preprint arXiv:1312.6114 (2013) (cit. on p. 1).
- [18] R. Rombach, A. Blattmann, D. Lorenz, P. Esser and B. Ommer, “High-Resolution Image Synthesis with Latent Diffusion Models”, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022 (cit. on p. 1).
- [19] J. J. Park, P. Florence, J. Straub, R. Newcombe and S. Lovegrove, “Deepsdf: Learning Continuous Signed Distance Functions for Shape Representation”, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019 (cit. on p. 1).
- [20] A. Radford et al., *Language Models are Unsupervised Multitask Learners*, OpenAI Blog **1** (2019) (cit. on pp. 1, 3).
- [21] OpenAI, *GPT-4 Technical Report*, 2023, arXiv: 2303.08774 [cs.CL] (cit. on p. 1).
- [22] A. Ramesh et al., “Zero-shot Text-to-Image Generation”, *International Conference on Machine Learning*, 2021 (cit. on p. 1).
- [23] U. Singer et al., *Make-a-video: Text-to-video Generation without Text-video Data*, arXiv preprint arXiv:2209.14792 (2022) (cit. on p. 1).
- [24] A. Agostinelli et al., *Musiclm: Generating Music from Text*, arXiv preprint arXiv:2301.11325 (2023) (cit. on p. 1).
- [25] L. A. Ramshaw and M. P. Marcus, *Text Chunking using Transformation-based Learning*, *Natural Language Processing using Very Large Corpora* (1999) (cit. on p. 2).
- [26] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, 2018 (cit. on pp. 2, 100).
- [27] U. Germann, M. Jahr, K. Knight, D. Marcu and K. Yamada, *Fast and Optimal Decoding for Machine Translation*, *Artificial Intelligence* **154** (2004) (cit. on p. 2).
- [28] H. Daumé, J. Langford and D. Marcu, *Search-based Structured Prediction*, *Machine Learning* **75** (2009) (cit. on pp. 2, 5, 80, 90).
- [29] R. Kneser and H. Ney, “Improved Backing-off for M-gram Language Modeling”, *International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, 1995 (cit. on p. 2).

-
- [30] F. Jelinek and R. L. Mercer, “Interpolated Estimation of Markov Source Parameters from Sparse Data”, *Proceedings of Workshop on Pattern Recognition in Practice*, 1980 (cit. on p. 2).
- [31] Y. Bengio, R. Ducharme and P. Vincent, *A Neural Probabilistic Language Model*, *Advances in Neural Information Processing Systems* **13** (2000) (cit. on p. 2).
- [32] T. Mikolov, K. Chen, G. Corrado and J. Dean, *Efficient Estimation of Word Representations in Vector Space*, arXiv preprint arXiv:1301.3781 (2013) (cit. on pp. 2, 26).
- [33] P. Bojanowski, E. Grave, A. Joulin and T. Mikolov, *Enriching Word Vectors with Subword Information*, *Transactions of the Association for Computational Linguistics* **5** (2017) (cit. on pp. 2, 26).
- [34] J. Pennington, R. Socher and C. D. Manning, “Glove: Global vectors for word representation”, *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014 (cit. on pp. 2, 26).
- [35] A. Akbik, D. Blythe and R. Vollgraf, “Contextual String Embeddings for Sequence Labeling”, *Proceedings of International Conference on Computational Linguistics*, 2018 (cit. on p. 3).
- [36] M. Peters et al., *Deep Contextualized Word Representations*, arXiv preprint arXiv:1802.05365 **12** (2018) (cit. on pp. 3, 26).
- [37] J. Howard and S. Ruder, *Universal Language Model Fine-tuning for Text Classification*, arXiv preprint arXiv:1801.06146 (2018) (cit. on pp. 3, 27).
- [38] I. Sutskever, O. Vinyals and Q. V. Le, “Sequence to Sequence Learning with Neural Networks”, *Proceedings of Neural Information Processing Systems*, 2014 (cit. on pp. 3, 13, 55).
- [39] D. Bahdanau, K. Cho and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, arXiv preprint arXiv:1409.0473 (2014) (cit. on pp. 3, 13).
- [40] M.-T. Luong, H. Pham and C. D. Manning, *Effective Approaches to Attention-based Neural Machine Translation*, arXiv preprint arXiv:1508.04025 (2015) (cit. on p. 3).
- [41] K. Xu et al., “Show, Attend and Tell: Neural Image Caption Generation with Visual Attention”, *Proceedings of International conference on machine learning*, 2015 (cit. on p. 3).
- [42] A. Vaswani et al., “Attention is All You Need”, *Proceedings of Advances in Neural Information Processing Systems*, 2017 (cit. on pp. 3, 80).
- [43] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv preprint arXiv:1810.04805 (2018) (cit. on pp. 3, 27, 80, 85).
- [44] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever et al., *Improving Language Understanding by Generative Pre-training*, (2018) (cit. on p. 3).

- [45] C. Raffel et al., *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, *Journal of Machine Learning Research* **21** (2020) (cit. on p. 3).
- [46] L. Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback”, *Proceedings of Neural Information Processing Systems*, 2022 (cit. on pp. 3, 6, 87, 89, 90, 92, 100).
- [47] H. Touvron et al., *Llama: Open and Efficient Foundation Language Models*, arXiv preprint arXiv:2302.13971 (2023) (cit. on p. 3).
- [48] S. Bubeck et al., *Sparks of Artificial General Intelligence: Early experiments with GPT-4*, arXiv preprint arXiv:2303.12712 (2023) (cit. on p. 3).
- [49] M. Hausknecht and P. Stone, *Deep Recurrent Q-Learning for Partially Observable MDPs*, 2017, arXiv: 1507.06527 [cs.LG] (cit. on p. 3).
- [50] C. Romac and V. Béraud, *Deep Recurrent Q-Learning vs Deep Q-Learning on a simple Partially Observable Markov Decision Process with Minecraft*, 2019, arXiv: 1903.04311 [cs.LG] (cit. on p. 3).
- [51] R. Ramamurthy, C. Bauckhage, K. Buza and S. Wrobel, “Using Echo State Networks for Cryptography”, *Proceedings of International Conference on Artificial Neural Networks*, 2017, URL: https://doi.org/10.1007/978-3-319-68612-7_75 (cit. on pp. 4, 8, 18, 28, 197).
- [52] R. Ramamurthy, R. Stenzel, R. Sifa, A. Ladi and C. Bauckhage, “Echo State Networks for Named Entity Recognition”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30493-5_11 (cit. on pp. 4, 8, 26, 197).
- [53] R. Ramamurthy, C. Bauckhage, R. Sifa and S. Wrobel, “Policy Learning Using SPSA”, *Proceedings of International Conference on Artificial Neural Networks*, 2018, URL: https://doi.org/10.1007/978-3-030-01424-7_1 (cit. on pp. 4, 8, 34, 197).
- [54] G. Tesauro et al., *Temporal Difference Learning and TD-Gammon*, *Communications of the ACM* **38** (1995) (cit. on p. 4).
- [55] S. Singh, D. Litman, M. Kearns and M. Walker, *Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System*, *Journal of Artificial Intelligence Research* **16** (2002) (cit. on p. 4).
- [56] A. Y. Ng et al., “Autonomous Inverted Helicopter Flight via Reinforcement Learning”, *Experimental Robotics IX: International Symposium on Experimental Robotics*, 2006 (cit. on p. 4).
- [57] V. Mnih et al., *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602 (2013) (cit. on p. 4).
- [58] D. Silver et al., *Mastering the Game of Go without Human Knowledge*, *Nature* **550** (2017) (cit. on pp. 4, 34, 48).
- [59] M. Campbell, A. J. Hoane Jr and F.-h. Hsu, *Deep Blue*, *Artificial intelligence* **134** (2002) (cit. on p. 4).

-
- [60] D. Ferrucci et al., *Building Watson: An Overview of the DeepQA Project*, AI magazine **31** (2010) (cit. on p. 4).
- [61] C. Berner et al., *Dota 2 with Large Scale Deep Reinforcement Learning*, arXiv preprint arXiv:1912.06680 (2019) (cit. on p. 4).
- [62] S. Levine, C. Finn, T. Darrell and P. Abbeel, *End-to-end Training of Deep Visuomotor Policies*, Journal of Machine Learning Research **17** (2016) (cit. on p. 4).
- [63] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz and D. Quillen, *Learning Hand-eye Coordination for Robotic Grasping with Deep Learning and Large-scale Data Collection*, International Journal of Robotics Research **37** (2018) (cit. on p. 4).
- [64] O. M. Andrychowicz et al., *Learning Dexterous In-hand Manipulation*, International Journal of Robotics Research **39** (2020) (cit. on p. 4).
- [65] N. C. Luong et al., *Applications of Deep Reinforcement Learning in Communications and Networking: A Survey*, IEEE Communications Surveys & Tutorials **21** (2019) (cit. on p. 4).
- [66] V. Talpaert et al., *Exploring Applications of Deep Reinforcement Learning for Real-world Autonomous Driving Systems*, arXiv preprint arXiv:1901.01536 (2019) (cit. on p. 5).
- [67] M. M. Afsar, T. Crump and B. Far, *Reinforcement Learning Based Recommender Systems: A Survey*, ACM Computing Surveys **55** (2022) (cit. on p. 5).
- [68] N. Lazic et al., *Data Center Cooling using Model-predictive Control*, Advances in Neural Information Processing Systems **31** (2018) (cit. on p. 5).
- [69] R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Novelty-Guided Reinforcement Learning via Encoded Behaviors”, *Proceedings of International Joint Conference on Neural Networks*, 2020, URL: <https://doi.org/10.1109/IJCNN48605.2020.9206982> (cit. on pp. 5, 8, 47, 48, 197).
- [70] R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Guided Reinforcement Learning via Sequence Learning”, *Proceedings of International Conference on Artificial Neural Networks*, 2020, URL: https://doi.org/10.1007/978-3-030-61616-8_27 (cit. on pp. 5, 8, 47, 48, 197).
- [71] R. Ramamurthy, C. Bauckhage, R. Sifa, J. Schücker and S. Wrobel, “Leveraging Domain Knowledge for Reinforcement Learning Using MMC Architectures”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30484-3_48 (cit. on pp. 5, 8, 63, 197).
- [72] F. Maes, L. Denoyer and P. Gallinari, *Structured Prediction with Reinforcement Learning*, Machine learning **77** (2009) (cit. on pp. 5, 80, 83).
- [73] C.-Y. Lin, “Rouge: A Package for Automatic Evaluation of Summaries”, *Text Summarization Branches Out*, 2004 74 (cit. on pp. 5, 92).

- [74] S. Ryang and T. Abekawa, “Framework of Automatic Text Summarization using Reinforcement Learning”, *Proceedings of Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012 (cit. on pp. 6, 79, 80).
- [75] Y. Gao, C. M. Meyer and I. Gurevych, *APRIL: Interactively Learning to Summarise by Combining Active Preference Learning and Reinforcement Learning*, arXiv preprint arXiv:1808.09658 (2018) (cit. on pp. 6, 79, 80).
- [76] Y.-C. Chen and M. Bansal, “Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting”, *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2018 (cit. on pp. 6, 79, 80).
- [77] S. Wang et al., *R3: Reinforced Reader-Ranker for Open-Domain Question Answering*, arXiv preprint arXiv:1709.00023 (2017) (cit. on pp. 6, 79, 80).
- [78] X. Yuan et al., *Interactive Language Learning by Question Answering*, arXiv preprint arXiv:1908.10909 (2019) (cit. on pp. 6, 79, 80).
- [79] K. Narasimhan, A. Yala and R. Barzilay, *Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning*, arXiv preprint arXiv:1603.07954 (2016) (cit. on pp. 6, 79, 80).
- [80] G. Brockman et al., *OpenAI Gym*, arXiv:1606.01540 (2016) (cit. on pp. 6, 38, 64, 66, 80, 91).
- [81] M. Johnson, K. Hofmann, T. Hutton and D. Bignell, “The Malmo Platform for Artificial Intelligence Experimentation.”, *Proceedings of International Joint Conference on Artificial Intelligence*, 2016 (cit. on pp. 6, 80).
- [82] M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling, *The Arcade Learning Environment: An Evaluation Platform for General Agents*, *Journal of Artificial Intelligence Research* **47** (2013) (cit. on pp. 6, 80).
- [83] M.-A. Côté et al., “Textworld: A Learning Environment for Text-based games”, *Workshop on Computer Games*, 2018 (cit. on pp. 6, 80, 81).
- [84] M. Chevalier-Boisvert et al., *BabyAI: First Steps Towards Grounded Language Learning with a Human in the Loop*, arXiv preprint arXiv:1810.08272 (2018) (cit. on pp. 6, 80, 81).
- [85] D. M. Ziegler et al., *Fine-tuning Language Models from Human Preferences*, arXiv preprint arXiv:1909.08593 (2019) (cit. on pp. 6, 90, 92).
- [86] N. Stiennon et al., *Learning to Summarize with Human Feedback*, *Advances in Neural Information Processing Systems* **33** (2020) (cit. on pp. 6, 90).
- [87] L. Choshen, L. Fox, Z. Aizenbud and O. Abend, “On the Weaknesses of Reinforcement Learning for Neural Machine Translation”, *Proceedings of International Conference on Learning Representations*, 2020 (cit. on pp. 6, 89, 90).

-
- [88] J. Kreutzer, S. Riezler and C. Lawrence, “Offline Reinforcement Learning from Human Feedback in Real-World Sequence-to-Sequence Tasks”, *Proceedings of Workshop on Structured Prediction for NLP*, 2021 (cit. on pp. 6, 89).
- [89] R. Ramamurthy, R. Sifa and C. Bauckhage, “NLP Gym – A Toolkit for Evaluating RL agents on Natural Language Processing Tasks”, *Proceedings of Wordplay Workshop in NeurIPS 2020*, URL: <https://doi.org/10.48550/arXiv.2011.08272> (cit. on pp. 6, 8, 80, 197).
- [90] R. Ramamurthy et al., “Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”, *Proceedings of International Conference on Learning Representations*, 2023, URL: <https://openreview.net/forum?id=8aHzds2uUyB> (cit. on pp. 6, 8, 88, 197).
- [91] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, “Bleu: A Method for Automatic Evaluation of Machine Translation”, *Proceedings of Annual meeting of the Association for Computational Linguistics*, 2002 (cit. on pp. 6, 88, 92).
- [92] I. Sutskever, J. Martens and G. E. Hinton, “Generating Text with Recurrent Neural Networks”, *Proceedings of International Conference on Machine Learning*, 2011 (cit. on p. 13).
- [93] A. Graves, *Generating Sequences with Recurrent Neural Networks*, arXiv preprint arXiv:1308.0850 (2013) (cit. on p. 13).
- [94] R. Pascanu, T. Mikolov and Y. Bengio, “On the Difficulty of Training Recurrent Neural Networks”, *Proceedings of International Conference on Machine Learning*, 2013 (cit. on p. 13).
- [95] K. Doya, “Bifurcations in the Learning of Recurrent Neural Networks”, *Proceedings of International Symp. on Circuits and Systems*, 1992 (cit. on p. 13).
- [96] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, *Neural computation* **9** (1997) (cit. on pp. 13, 26).
- [97] K. Cho et al., *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, arXiv preprint arXiv:1406.1078 (2014) (cit. on pp. 13, 57).
- [98] H. Jäger, *The “Echo State” Approach to Analysing and Training Recurrent Neural Networks*, tech. rep. 148, GMD, 2001 (cit. on pp. 13, 18, 27, 30).
- [99] W. Maass, T. Natschläger and H. Markram, *Real-time Computing Without Stable States: A New Framework for Neural Computation based on Perturbations*, *Neural computation* **14** (2002) (cit. on pp. 13, 27).
- [100] C. Gallicchio and A. Micheli, *Deep Echo State Network (deepesn): A Brief Survey*, arXiv preprint arXiv:1712.04323 (2017) (cit. on p. 13).
- [101] M. Lukoševičius, “A Practical Guide to Applying Echo State Networks”, *Neural Networks: Tricks of the Trade*, vol. 7700, 2012 (cit. on pp. 15, 17, 28, 30, 31, 38).
- [102] H. Jaeger and H. Haas, *Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication*, *science* **304** (2004) (cit. on pp. 17, 28).

- [103] M. Clark and D. Blank, “A Neural-Network Based Cryptographic System”, *Proceedings of Midwest Artificial Intelligence and Cognitive Science Conference*, 1998 (cit. on p. 18).
- [104] I. Kanter, W. Kinzel and E. Kanter, *Secure Exchange of Information by Synchronization of Neural Networks*, *Europhysics Letters* **57** (2002) (cit. on p. 18).
- [105] A. Klimov, A. Mityagin and A. Shamir, “Analysis of Neural Cryptography”, *Proceedings of Int. Conf. on Theory and Application of Cryptology and Information Security*, 2002 (cit. on p. 18).
- [106] M. Abadi and D. G. Andersen, *Learning to Protect Communications with Adversarial Neural Cryptography*, arXiv:1610.06918 (2016) (cit. on p. 18).
- [107] C. Li, S. Li, D. Zhang and G. Chen, “Chosen-Plaintext Cryptanalysis of a Clipped-Neural-Network-Based Chaotic Cipher”, *Proceedings of International Symposium on Neural Networks*, 2005 (cit. on p. 18).
- [108] S. Lian, *A Block Cipher Based on Chaotic Neural Networks*, *Neurocomputing* **72** (2009) (cit. on p. 18).
- [109] X.-Y. Wang, L. Yang, R. Liu and A. Kadir, *A Chaotic Image Encryption Algorithm Based on Perceptron Model*, *Nonlinear Dynamics* **62** (2010) (cit. on p. 18).
- [110] W. Yu and J. Cao, *Cryptography Based on Delayed Chaotic Neural Networks*, *Physics Letters A* **356** (2006) (cit. on p. 18).
- [111] T. Zhou, X. Liao and Y. Chen, “A Novel Symmetric Cryptography Based on Chaotic Signal Generator and a Clipped Neural Network”, *Proceedings of International Symposium on Neural Networks*, 2004 (cit. on p. 18).
- [112] C. E. Shannon, *Communication Theory of Secrecy Systems*, *Bell Labs Technical Journal* **28** (1949) (cit. on p. 24).
- [113] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition”, *Proceedings of the Conference on Natural Language Learning at HLT-NAACL*, 2003 (cit. on p. 26).
- [114] K. Toutanova, D. Klein, C. D. Manning and Y. Singer, “Feature-rich Part-of-Speech Tagging with a Cyclic Dependency Network”, *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 2003 (cit. on p. 26).
- [115] A. Conneau and D. Kiela, *Senteval: An Evaluation Toolkit for Universal Sentence Representations*, arXiv preprint arXiv:1803.05449 (2018) (cit. on p. 26).
- [116] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, *Squad: 100,000+ Questions for Machine Comprehension of Text*, arXiv preprint arXiv:1606.05250 (2016) (cit. on p. 26).

-
- [117] S. R. Bowman, G. Angeli, C. Potts and C. D. Manning, *A Large Annotated Corpus for Learning Natural Language Inference*, arXiv preprint arXiv:1508.05326 (2015) (cit. on p. 26).
- [118] A. Williams, N. Nangia and S. R. Bowman, *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*, arXiv preprint arXiv:1704.05426 (2017) (cit. on p. 26).
- [119] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman, *Indexing by Latent Semantic Analysis*, *Journal of the American Society for Information Science* **41** (1990) (cit. on p. 26).
- [120] A. Akbik et al., “FLAIR: An Easy-to-use Framework for State-of-the-art NLP”, *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, 2019 (cit. on p. 26).
- [121] R. Socher et al., “Recursive Deep Models for Semantic Compositionality over a Sentiment Treebank”, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2013 (cit. on p. 26).
- [122] N. Kalchbrenner, E. Grefenstette and P. Blunsom, *A Convolutional Neural Network for Modelling Sentences*, arXiv preprint arXiv:1404.2188 (2014) (cit. on p. 26).
- [123] Y. Kim, *Convolutional Neural Networks for Sentence Classification*, 2014, arXiv: 1408.5882 [cs.CL] (cit. on p. 26).
- [124] Q. Le and T. Mikolov, “Distributed Representations of Sentences and Documents”, *Proceedings of International conference on machine learning*, 2014 1188 (cit. on p. 27).
- [125] R. Kiros et al., “Skip-thought Vectors”, *Proceedings of Advances in Neural Information Processing Systems*, 2015 (cit. on p. 27).
- [126] A. Conneau, D. Kiela, H. Schwenk, L. Barrault and A. Bordes, *Supervised Learning of Universal Sentence Representations from Natural Language Inference Data*, arXiv preprint arXiv:1705.02364 (2017) (cit. on pp. 27, 29).
- [127] N. Reimers and I. Gurevych, *Sentence-bert: Sentence Embeddings using Siamese Bert-Networks*, arXiv preprint arXiv:1908.10084 (2019) (cit. on p. 27).
- [128] J. Wieting and D. Kiela, *No Training Required: Exploring Random Encoders for Sentence Classification*, arXiv preprint arXiv:1901.10444 (2019) (cit. on pp. 27, 29).
- [129] A. Conneau, G. Kruszewski, G. Lample, L. Barrault and M. Baroni, *What You Can Cram Into a Single Vector: Probing Sentence Embeddings for Linguistic Properties*, arXiv preprint arXiv:1805.01070 (2018) (cit. on p. 27).
- [130] E. B. Baum, *On the Capabilities of Multilayer Perceptrons*, *Journal of complexity* **4** (1988) (cit. on p. 27).

- [131] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin et al.,
“Feed Forward Neural Networks with Random Weights”,
Proceedings of International Conference on Pattern Recognition, 1992 (cit. on p. 27).
- [132] G.-B. Huang, Q.-Y. Zhu and C.-K. Siew,
Extreme Learning Machine: Theory and Applications, *Neurocomputing* **70** (2006)
(cit. on p. 27).
- [133] C. Gallicchio, A. Micheli and L. Pedrelli,
Deep Echo State Networks for Diagnosis of Parkinson’s Disease,
arXiv preprint arXiv:1802.06708 (2018) (cit. on p. 28).
- [134] X. Hinaut and P. F. Dominey,
“On-line Processing of Grammatical Structure using Reservoir Computing”,
Proceedings of International Conference on Artificial Neural Networks, 2012 (cit. on p. 28).
- [135] S. L. Frank,
Learn More by Training Less: Systematicity in Sentence Processing by Recurrent Networks,
Connection science **18** (2006) (cit. on p. 28).
- [136] D. Nadeau and S. Sekine, *A Survey of Named Entity Recognition and Classification*,
Linguisticae Investigationes **30** (2007) (cit. on p. 28).
- [137] V. Yadav and S. Bethard,
A Survey on Recent Advances in Named Entity Recognition from Deep Learning Models,
arXiv preprint arXiv:1910.11470 (2019) (cit. on p. 28).
- [138] D. Benikova, C. Biemann, M. Kisselew and S. Pado,
“Germeval 2014 Named Entity Recognition Shared Task: Companion Paper”,
Workshop Proceedings of KONVENS Conference, 2014 (cit. on p. 29).
- [139] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov,
Bag of Tricks for Efficient Text Classification, arXiv preprint arXiv:1607.01759 (2016)
(cit. on pp. 30, 84, 85, 109).
- [140] A. Akbik, D. Blythe and R. Vollgraf, “Contextual String Embeddings for Sequence Labeling”,
Proceedings of International Conference on Computational Linguistics, 2018 (cit. on p. 31).
- [141] Q. Ma, L. Shen and G. W. Cottrell,
Deep-esn: A Multiple Projection-Encoding Hierarchical Reservoir Computing Framework,
arXiv preprint arXiv:1711.05255 (2017) (cit. on p. 32).
- [142] L.-J. Lin, *Reinforcement Learning for Robots Using Neural Networks*,
tech. rep. CMU-CS-93-103, Carnegie-Mellon University, 1993 (cit. on pp. 33, 34).
- [143] D. Bertsekas, *Neuro-Dynamic Programming*, vol. 1, 1996 (cit. on p. 34).
- [144] R. Sutton, A. Barto et al., *Reinforcement Learning: An Introduction*, 1998 (cit. on p. 34).
- [145] V. Mnih et al., *Human-level Control Through Deep Reinforcement Learning*,
Nature **518** (2015) (cit. on pp. 34, 48, 64).
- [146] V. Mnih et al., “Asynchronous Methods for Deep Reinforcement Learning”,
Proceedings of International Conference on Machine Learning, 2016 (cit. on pp. 34, 91).

-
- [147] R. Sutton, *Introduction to Reinforcement Learning with Function Approximation*, Tutorial at the Conf. on Neural Information Processing Systems (2015) (cit. on p. 34).
- [148] T. Salimans, J. Ho, X. Chen and I. Sutskever, *Evolution Strategies as a Scalable Alternative to Reinforcement Learning*, arXiv:1703.03864 (2017) (cit. on pp. 34, 40, 49, 52).
- [149] H. Mania, A. Guy and B. Recht, *Simple Random Search Provides a Competitive Approach to Reinforcement Learning*, arXiv:1803.07055 (2018) (cit. on p. 34).
- [150] F. P. Such et al., *Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning*, arXiv:1712.06567 (2017) (cit. on p. 34).
- [151] Y. Duan, X. Chen, R. Houthoofd, J. Schulman and P. Abbeel, “Benchmarking Deep Reinforcement Learning for Continuous Control”, *Proceedings of International Conference on Machine Learning*, 2016 (cit. on p. 35).
- [152] K. C. Chatzidimitriou and P. A. Mitkas, “A NEAT Way for Evolving Echo State Networks”, *Proceedings of European Conference on Artificial Intelligence*, 2010 (cit. on p. 35).
- [153] J. Schmidhuber, D. Wierstra, M. Gagliolo and F. Gomez, *Training Recurrent Networks by Evolino*, *Neural Computation* **19** (2007) (cit. on p. 35).
- [154] F. Jiang, H. Berry and M. Schoenauer, “Supervised and Evolutionary Learning of Echo State Networks”, *Proceedings of International Conference on Parallel Problem Solving From Nature*, 2008 (cit. on p. 35).
- [155] B. Schrauwen, M. Wardermann, D. Verstraeten, J. J. Steil and D. Stroobandt, *Improving Reservoirs using Intrinsic Plasticity*, *Neurocomputing* **71** (2008) (cit. on p. 35).
- [156] P. Koprinkova-Hristova, “Three Approaches to Train Echo State Network Actors of Adaptive Critic Design”, *Proceedings of International Conference on Artificial Neural Networks*, 2016 (cit. on p. 35).
- [157] J. C. Spall, *Multivariate Stochastic Approximation using a Simultaneous Perturbation Gradient Approximation*, *IEEE Transactions on Automatic Control* **37** (1992) (cit. on p. 35).
- [158] H. Robbins and S. Monro, *A Stochastic Approximation Method*, *Annals of Mathematical Statistics* **22** (1951) (cit. on p. 36).
- [159] P. Mirowski et al., *Learning to Navigate in Complex Environments*, arXiv preprint arXiv:1611.03673 (2016) (cit. on p. 48).
- [160] Y. Zhu et al., “Target-Driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning”, *Proceedings of International Conference on Robotics and Automation*, 2017 (cit. on p. 48).
- [161] S. Gu, E. Holly, T. Lillicrap and S. Levine, “Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-policy Updates”, *Proceedings of International Conference on Robotics and Automation*, 2017 (cit. on p. 48).

- [162] J. Schulman, S. Levine, P. Abbeel, M. Jordan and P. Moritz, “Trust Region Policy Optimization”, *Proceedings of International Conference on Machine Learning*, 2015 (cit. on pp. 48, 52, 91).
- [163] A. Y. Ng, D. Harada and S. Russell, “Policy Invariance under Reward Transformations: Theory and Application to Reward Shaping”, *Proceedings of International Conference on Machine Learning*, 1999 (cit. on p. 48).
- [164] N. Heess et al., *Emergence of Locomotion Behaviours in Rich Environments*, arXiv preprint arXiv:1707.02286 (2017) (cit. on p. 48).
- [165] K. D. Asis, J. F. Hernandez-Garcia, G. Z. Holland and R. S. Sutton, *Multi-step Reinforcement Learning: A Unifying Algorithm*, arXiv preprint arXiv:1703.01327 (2017) (cit. on p. 48).
- [166] C. Sherstan et al., *Directly Estimating the Variance of the λ -return using Temporal-Difference Methods*, arXiv preprint arXiv:1801.08287 (2018) (cit. on p. 48).
- [167] D. Amodei et al., *Concrete problems in AI Safety*, arXiv preprint arXiv:1606.06565 (2016) (cit. on p. 48).
- [168] J. Randlev and P. Alstrm, “Learning to Drive a Bicycle Using Reinforcement Learning and Shaping”, *Proceedings of International Conference on Machine Learning*, 1998 (cit. on pp. 48, 65).
- [169] A. Y. Ng, S. J. Russell et al., “Algorithms for Inverse Reinforcement Learning”, *Proceedings of International Conference on Machine Learning*, 2000 (cit. on p. 49).
- [170] H. B. Suay, T. Brys, M. E. Taylor and S. Chernova, “Learning from Demonstration for Shaping Through Inverse Reinforcement Learning”, *Proceedings of International Conference on Autonomous Agents & Multiagent Systems*, 2016 (cit. on pp. 49, 65).
- [171] R. Loftin et al., *Learning Behaviors via Human-delivered Discrete Feedback: Modeling Implicit Feedback Strategies to speed up Learning*, *Proceedings of Conference on Autonomous agents and Multiagent systems* **30** (2016) (cit. on pp. 49, 65).
- [172] B. Peng et al., “A Need for Speed: Adapting Agent Action Speed to Improve Task Learning from Non-Expert Humans”, *Proceedings of International Conference on Autonomous Agents & Multiagent Systems*, 2016 (cit. on pp. 49, 65).
- [173] G. Ostrovski, M. G. Bellemare, A. van den Oord and R. Munos, *Count-Based Exploration with Neural Density Models*, 2017, arXiv: 1703.01310 [cs.AI] (cit. on pp. 49, 50).
- [174] D. Pathak, P. Agrawal, A. A. Efros and T. Darrell, “Curiosity-Driven Exploration by Self-supervised Prediction”, *Proceedings of International Conference on Machine Learning*, 2017 (cit. on pp. 49, 50).
- [175] I. Rechenberg, “Evolutionsstrategien”, *Simulationmethoden in der Medizin und Biologie*, 1978 83 (cit. on pp. 49, 52).

-
- [176] J. Lehman and K. O. Stanley, “Evolving a diversity of Virtual Creatures through Novelty Search and Local Competition”, *Proceedings of International Conference on Genetic and Evolutionary Computation*, 2011 (cit. on pp. 49, 65).
- [177] J. Lehman and K. O. Stanley, *Abandoning Objectives: Evolution through the Search for Novelty Alone*, *Evolutionary computation* **19** (2011) (cit. on p. 49).
- [178] E. Conti et al., *Improving Exploration in Evolution Strategies for Deep Reinforcement Learning via a Population of Novelty-Seeking Agents*, arXiv preprint arXiv:1712.06560 (2017) (cit. on pp. 49, 53, 65).
- [179] J. Schmidhuber, *Formal Theory of Creativity, Fun, and Intrinsic Motivation (1990–2010)*, *IEEE Transactions on Autonomous Mental Development* **2** (2010) (cit. on p. 50).
- [180] M. G. Bellemare et al., *Unifying Count-Based Exploration and Intrinsic Motivation*, arXiv preprint arXiv:1606.01868 (2016) (cit. on p. 50).
- [181] R. Houthoofd et al., *VIME: Variational Information Maximizing Exploration*, 2016, arXiv: 1605.09674 [cs.LG] (cit. on p. 50).
- [182] B. Eysenbach, A. Gupta, J. Ibarz and S. Levine, *Diversity is All You Need: Learning Skills without a Reward Function*, arXiv preprint arXiv:1802.06070 (2018) (cit. on p. 50).
- [183] H. Tang et al., “Exploration: A study of Count-Based Exploration for Deep Reinforcement Learning”, *Proceedings of Advances in Neural Information Processing Systems*, 2017 (cit. on p. 50).
- [184] B. C. Stadie, S. Levine and P. Abbeel, *Incentivizing Exploration In Reinforcement Learning With Deep Predictive Models*, 2015, arXiv: 1507.00814 [cs.AI] (cit. on p. 50).
- [185] M. Chevalier-Boisvert, L. Willems and S. Pal, *Minimalistic Gridworld Environment for OpenAI Gym*, <https://github.com/maximecb/gym-minigrid>, 2018 (cit. on p. 50).
- [186] D. Silver et al., “Deterministic Policy Gradient Algorithms”, *Proceedings of International Conference on Machine Learning*, 2014 (cit. on p. 52).
- [187] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.*, PhD thesis: Thesis, Technical University of Berlin, Department of Process Engineering, 1971 (cit. on p. 52).
- [188] N. Srivastava, E. Mansimov and R. Salakhudinov, “Unsupervised Learning of Video Representations using Lstms”, *Proceedings of International Conference on Machine Learning*, 2015 (cit. on p. 55).
- [189] A. Makhzani and B. Frey, *K-sparse Autoencoders*, arXiv preprint arXiv:1312.5663 (2013) (cit. on p. 55).
- [190] R. Ramamurthy, *pytorch-optimize is a simple Black-box Optimisation Framework*, version 0.0.1, URL: <https://github.com/rajcsw/pytorch-optimize> (cit. on p. 57).

- [191] F. Maes, R. Fonteneau, L. Wehenkel and D. Ernst, “Policy Search in a Space of Simple Closed-form Formulas: Towards Interpretability of Reinforcement Learning”, *Proceedings of International Conference on Discovery Science*, 2012 (cit. on pp. 63, 64).
- [192] A. Verma, V. Murali, R. Singh, P. Kohli and S. Chaudhuri, *Programmatically Interpretable Reinforcement Learning*, arXiv preprint arXiv:1804.02477 (2018) (cit. on pp. 63, 64).
- [193] M. G. Bellemare, Y. Naddaf, J. Veness and M. Bowling, *The Arcade Learning Environment: An Evaluation Platform for General Agents*, *Journal of Artificial Intelligence Research* **47** (2013) (cit. on p. 64).
- [194] Y. Tassa et al., *DeepMind Control Suite*, arXiv preprint arXiv:1801.00690 (2018) (cit. on pp. 64, 66).
- [195] A. Gupta, C. Devin, Y. Liu, P. Abbeel and S. Levine, *Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning*, arXiv preprint arXiv:1703.02949 (2017) (cit. on p. 64).
- [196] M. Wulfmeier, I. Posner and P. Abbeel, *Mutual Alignment Transfer Learning*, arXiv preprint arXiv:1707.07907 (2017) (cit. on p. 64).
- [197] A. A. Rusu et al., *Sim-to-real Robot Learning from Pixels with Progressive Nets*, arXiv preprint arXiv:1610.04286 (2016) (cit. on p. 64).
- [198] F. Zhang, J. Leitner, Z. Ge, M. Milford and P. Corke, *Sim-to-real Transfer of Visuo-motor Policies for reaching in clutter*, arXiv preprint arXiv:1709.05746 (2017) (cit. on p. 64).
- [199] U. Steinkühler and H. Cruse, *A Holistic Model for an Internal Representation to control the movement of a Manipulator with redundant degrees of freedom*, *Biological Cybernetics* **79** (1998) (cit. on pp. 65, 66, 68).
- [200] H. Cruse, T. Kindermann, M. Schumm, J. Dean and J. Schmitz, *Walknet—a biologically inspired network to control six-legged walking*, *Neural networks* **11** (1998) (cit. on p. 65).
- [201] H. Cruse, *A Recurrent Network for Landmark-based Navigation*, *Biological cybernetics* **88** (2003) (cit. on p. 65).
- [202] M. J. Mataric, “Reward Functions for Accelerated Learning”, *Machine Learning Proceedings 1994*, 1994 (cit. on p. 65).
- [203] M. Dorigo and M. Colombetti, *Robot Shaping: Developing Autonomous Agents through Learning*, *Artificial intelligence* **71** (1994) (cit. on p. 65).
- [204] J. K. Pugh, L. B. Soros and K. O. Stanley, *Quality Diversity: A New Frontier for Evolutionary Computation*, *Frontiers in Robotics and AI* **3** (2016) (cit. on p. 65).
- [205] A. Cully, J. Clune, D. Tarapore and J.-B. Mouret, *Robots That Can Adapt Like Animals*, *Nature* **521** (2015) (cit. on p. 65).

-
- [206] J. Ho and S. Ermon, “Generative Adversarial Imitation Learning”, *Proceedings of Advances in Neural Information Processing Systems*, 2016 (cit. on p. 65).
- [207] Y. Zhu et al., *Reinforcement and Imitation Learning for Diverse Visuomotor Skills*, arXiv preprint arXiv:1802.09564 (2018) (cit. on p. 65).
- [208] A. Rajeswaran et al., *Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations*, arXiv preprint arXiv:1709.10087 (2017) (cit. on p. 65).
- [209] M. E. Taylor, H. B. Suay and S. Chernova, “Integrating Reinforcement Learning with Human Demonstrations of Varying Ability”, *Proceedings of International Conference on Autonomous Agents and Multiagent Systems*, 2011 (cit. on p. 65).
- [210] Z. Wang and M. E. Taylor, “Improving Reinforcement Learning with Confidence-Based Demonstrations”, *Proceedings of International Conference on Artificial Intelligence*, 2017 (cit. on p. 65).
- [211] A. Rosenfeld, M. E. Taylor and S. Kraus, “Leveraging Human Knowledge in Tabular Reinforcement Learning: a study of human subjects”, *Proceedings of International Joint Conference on Artificial Intelligence*, 2017 (cit. on p. 65).
- [212] C. Devin, A. Gupta, T. Darrell, P. Abbeel and S. Levine, “Learning Modular Neural Network Policies for Multi-Task and Multi-Robot Transfer”, *Proceedings of International Conference on Robotics and Automation*, 2017 (cit. on p. 65).
- [213] I. Clavera, D. Held and P. Abbeel, “Policy Transfer via Modularity and Reward Guiding”, *Proceedings of Intelligent Robots and Systems*, 2017 (cit. on p. 65).
- [214] M. Müller, A. Dosovitskiy, B. Ghanem and V. Koltun, *Driving Policy Transfer via Modularity and Abstraction*, arXiv preprint arXiv:1804.09364 (2018) (cit. on p. 65).
- [215] A. Akbik, D. Blythe and R. Vollgraf, “Contextual String Embeddings for Sequence Labeling”, *Proceedings of International Conference on Computational Linguistics*, 2018 (cit. on pp. 80, 109).
- [216] J. Straková, M. Straka and J. Hajic, “Neural Architectures for Nested NER through Linearization”, *Proceedings of International Conference on Computational Linguistics*, 2019 (cit. on p. 80).
- [217] R. Nallapati, F. Zhai and B. Zhou, *Summarunner: A Recurrent Neural Network based Sequence model for Extractive Summarization of Documents*, arXiv preprint arXiv:1611.04230 (2016) (cit. on p. 80).
- [218] M. Zhong et al., *Extractive Summarization as Text Matching*, arXiv preprint arXiv:2004.08795 (2020) (cit. on p. 80).
- [219] K. M. Hermann et al., “Teaching Machines to Read and Comprehend”, *Proceedings of Advances in Neural Information Processing Systems*, 2015 (cit. on pp. 80, 95).
- [220] Z. Yang et al., “Xlnet: Generalized Autoregressive Pretraining for Language Understanding”, *Proceedings of Advances in Neural Information Processing Systems*, 2019 (cit. on p. 80).

- [221] K. Narasimhan, T. Kulkarni and R. Barzilay, *Language Understanding for Text-based Games using Deep Reinforcement Learning*, arXiv preprint arXiv:1506.08941 (2015) (cit. on pp. 80, 90).
- [222] P. Ammanabrolu et al., *How to Motivate Your Dragon: Teaching Goal-Driven Agents to Speak and Act in Fantasy Worlds*, 2020, arXiv: 2010.00685 [cs.CL] (cit. on p. 80).
- [223] P. Dhariwal et al., *OpenAI Baselines*, <https://github.com/openai/baselines>, 2017 (cit. on pp. 81, 82).
- [224] Hill, Ashley and Raffin, Antonin and Ernestus, Maximilian and Gleave, Adam and Kanervisto, Anssi and Traore, Rene and Dhariwal, Prafulla and Hesse, Christopher and Klimov, Oleg and Nichol, Alex and Plappert, Matthias and Radford, Alec and Schulman, John and Sidor, Szymon and Wu, Yuhuai, *Stable Baselines*, <https://github.com/hill-a/stable-baselines>, 2018 (cit. on pp. 81, 82, 84).
- [225] E. Liang et al., “RLlib: Abstractions for Distributed Reinforcement Learning”, *Proceedings of International Conference on Machine Learning*, 2018 (cit. on pp. 81, 82).
- [226] K. M. Hermann et al., *Grounded Language Learning in a Simulated 3D World*, arXiv preprint arXiv:1706.06551 (2017) (cit. on p. 81).
- [227] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal and R. Salakhutdinov, *Gated-attention Architectures for Task-Oriented Language Grounding*, arXiv preprint arXiv:1706.07230 (2017) (cit. on p. 81).
- [228] M. J. Hausknecht, P. Ammanabrolu, M.-A. Côté and X. Yuan, “Interactive Fiction Games: A Colossal Adventure”, *Proceedings of AAAI Conference on Artificial Intelligence*, 2020 (cit. on p. 81).
- [229] J. Urbanek et al., *Learning to Speak and Act in a Fantasy Text Adventure Game*, arXiv preprint arXiv:1903.03094 (2019) (cit. on p. 81).
- [230] T. Khot, P. Clark, M. Guerquin, P. Jansen and A. Sabharwal, *QASC: A Dataset for Question Answering via Sentence Composition*, arXiv (2019) arXiv (cit. on pp. 83, 85).
- [231] D. Khashabi et al., *UnifiedQA: Crossing Format Boundaries With a Single QA System*, 2020 (cit. on p. 83).
- [232] G. Tsoumakas and I. Katakis, *Multi-label Classification: An Overview*, *International Journal of Data Warehousing and Mining (IJDWM)* **3** (2007) (cit. on p. 83).
- [233] W. Chen, T.-Y. Liu, Y. Lan, Z.-M. Ma and H. Li, “Ranking Measures and Loss Functions in Learning to Rank”, *Proceedings of Advances in Neural Information Processing Systems*, 2009 (cit. on p. 83).
- [234] E. F. Sang and F. De Meulder, *Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition*, arXiv preprint cs/0306050 (2003) (cit. on p. 84).
- [235] N. Silveira et al., “A Gold Standard Dependency Corpus for English”, *Proceedings of International Conference on Language Resources and Evaluation*, 0204 (cit. on p. 84).

-
- [236] B. Heinzerling and M. Strube, “BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages”, *Proceedings of International Conference on Language Resources and Evaluation*, 2018 (cit. on pp. 84, 85, 109).
- [237] P. Clark et al., *Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge*, arXiv:1803.05457v1 (2018) (cit. on p. 85).
- [238] D. D. Lewis, *Reuters-21578*, <http://www.daviddlewis.com/resources/testcollections/reuters21578/> (cit. on p. 85).
- [239] P. Yang et al., “SGM: Sequence Generation Model for Multi-label Classification”, *Proceedings of International Conference on Computational Linguistics*, 2018 (cit. on p. 85).
- [240] J. Wu et al., *Recursively Summarizing Books with Human Feedback*, arXiv preprint arXiv:2109.10862 (2021) (cit. on pp. 87, 89, 90, 92, 100, 122).
- [241] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger and Y. Artzi, *Bertscore: Evaluating Text Generation with Bert*, arXiv preprint arXiv:1904.09675 (2019) (cit. on pp. 88, 92).
- [242] T. Sellam, D. Das and A. Parikh, “BLEURT: Learning Robust Metrics for Text Generation”, *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 2020 (cit. on pp. 88, 92).
- [243] S. Banerjee and A. Lavie, “METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments”, *Proceedings of ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and Summarization*, 2005 (cit. on pp. 88, 92).
- [244] M. Strathern, ‘Improving ratings’: *Audit in the British University System*, *European Review* **5** (1997) (cit. on p. 89).
- [245] S. Bengio, O. Vinyals, N. Jaitly and N. Shazeer, *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*, *Advances in Neural Information Processing Systems* **28** (2015) (cit. on p. 90).
- [246] D. Duckworth, A. Neelakantan, B. Goodrich, L. Kaiser and S. Bengio, *Parallel Scheduled Sampling*, arXiv preprint arXiv:1906.04331 (2019) (cit. on p. 90).
- [247] T. Mihaylova and A. F. Martins, *Scheduled Sampling for Transformers*, arXiv preprint arXiv:1906.07651 (2019) (cit. on p. 90).
- [248] K. Goyal, C. Dyer and T. Berg-Kirkpatrick, *Differentiable Scheduled Sampling for Credit Assignment*, arXiv preprint arXiv:1704.06970 (2017) (cit. on p. 90).
- [249] G. Lampouras and A. Vlachos, “Imitation Learning for Language Generation from Unaligned Data”, *Proceedings of International Conference on Computational Linguistics: Technical Papers*, 2016 (cit. on p. 90).

- [250] K.-W. Chang, A. Krishnamurthy, A. Agarwal, H. Daumé III and J. Langford, “Learning to Search Better than your Teacher”, *Proceedings of International Conference on Machine Learning*, 2015 (cit. on p. 90).
- [251] R. Leblond, J.-B. Alayrac, A. Osokin and S. Lacoste-Julien, *SEARNN: Training RNNs with Global-local Losses*, arXiv preprint arXiv:1706.04499 (2017) (cit. on p. 90).
- [252] S. Ross, G. Gordon and D. Bagnell, “A Reduction of Imitation Learning and Structured Prediction to No-regret Online Learning”, *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2011 (cit. on p. 90).
- [253] F. Huszár, *How (not) to Train Your Generative Model: Scheduled Sampling, Likelihood, Adversary?*, arXiv preprint arXiv:1511.05101 (2015) (cit. on p. 90).
- [254] A. Agarwal, N. Jiang, S. M. Kakade and W. Sun, *Reinforcement learning: Theory and algorithms*, CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep (2019) 10 (cit. on p. 90).
- [255] G. Swamy, S. Choudhury, J. A. Bagnell and S. Wu, “Of Moments and Matching: A Game-theoretic Framework for Closing the Imitation Gap”, *Proceedings of International Conference on Machine Learning*, 2021 (cit. on p. 90).
- [256] Y. Wu et al., *Google’s Neural Machine Translation System: Bridging the Gap Between Human and Machine Translation*, arXiv preprint arXiv:1609.08144 (2016) (cit. on p. 90).
- [257] S. Kiegeand and J. Kreutzer, “Revisiting the Weaknesses of Reinforcement Learning for Neural Machine Translation”, *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2021 (cit. on p. 90).
- [258] R. Paulus, C. Xiong and R. Socher, *A Deep Reinforced Model for Abstractive Summarization*, arXiv preprint arXiv:1705.04304 (2017) (cit. on p. 90).
- [259] J. Li et al., “Deep Reinforcement Learning for Dialogue Generation”, *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2016 (cit. on pp. 90, 122).
- [260] L. Zhou, K. Small, O. Rokhlenko and C. Elkan, *End-to-End Offline Goal-Oriented Dialog Policy Learning via Policy Gradient*, CoRR **abs/1712.02838** (2017) (cit. on p. 90).
- [261] N. Jaques et al., “Human-centric Dialog Training via Offline Reinforcement Learning”, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, 2020 3985 (cit. on p. 90).
- [262] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross and V. Goel, “Self-critical Sequence Training for Image Captioning”, *Proceedings of Computer Vision and Pattern Recognition*, 2017 (cit. on p. 90).
- [263] R. Y. Pang and H. He, “Text Generation by Learning from Demonstrations”, *Proceedings of International Conference on Learning Representations*, 2021 (cit. on p. 90).

-
- [264] M. Hausknecht, P. Ammanabrolu, M.-A. Côté and X. Yuan, “Interactive Fiction Games: A Colossal Adventure”, *Proceedings of AAAI Conference on Artificial Intelligence*, 2020 (cit. on p. 90).
- [265] M. Ranzato, S. Chopra, M. Auli and W. Zaremba, “Sequence Level Training with Recurrent Neural Networks”, *Proceedings of International Conference on Learning Representations*, 2016 (cit. on pp. 90, 91, 93).
- [266] C. Snell, I. Kostrikov, Y. Su, M. Yang and S. Levine, *Offline RL for Natural Language Generation with Implicit Language Q Learning*, arXiv preprint arXiv:2206.11871 (2022) (cit. on p. 90).
- [267] X. Lu et al., *Quark: Controllable Text Generation with Reinforced Unlearning*, arXiv e-prints (2022) arXiv (cit. on pp. 90, 100).
- [268] L. Chen et al., “Decision Transformer: Reinforcement Learning via Sequence Modeling”, *Proceedings of Advances in Neural Information Processing Systems*, 2021 (cit. on p. 90).
- [269] R. Nakano et al., *WebGPT: Browser-assisted Question-answering with Human Feedback*, arXiv preprint arXiv:2112.09332 (2021) (cit. on p. 90).
- [270] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, *Proximal Policy Optimization Algorithms*, arXiv preprint arXiv:1707.06347 (2017) (cit. on pp. 90, 91).
- [271] R. J. Williams, *Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning*, *Machine learning* **8** (1992) (cit. on p. 91).
- [272] D. Bahdanau et al., *An actor-critic Algorithm for Sequence Prediction*, arXiv preprint arXiv:1607.07086 (2016) (cit. on p. 91).
- [273] P. Ammanabrolu and M. Hausknecht, “Graph Constrained Reinforcement Learning for Natural Language Action Spaces”, *Proceedings of International Conference on Learning Representations*, 2020 (cit. on pp. 91, 93).
- [274] A. Martin et al., “Learning Natural Language Generation with Truncated Reinforcement Learning”, *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022 (cit. on p. 91).
- [275] T. Zahavy, M. Haroush, N. Merlis, D. J. Mankowitz and S. Mannor, *Learn What Not to Learn: Action Elimination with Deep Reinforcement Learning*, *Proceedings of Advances in Neural Information Processing Systems* **31** (2018) (cit. on pp. 91, 93).
- [276] P. Ammanabrolu, L. Jiang, M. Sap, H. Hajishirzi and Y. Choi, “Aligning to Social Norms and Values in Interactive Narratives”, *Proceedings of North American Chapter of the Association for Computational Linguistics*, 2022 (cit. on p. 91).

- [277] T. Wolf et al., “Transformers: State-of-the-art Natural Language Processing”, *Proceedings of Conference on empirical methods in natural language processing: system demonstrations*, 2020 (cit. on p. 91).
- [278] A. Raffin et al., *Stable-baselines3: Reliable Reinforcement Learning Implementations*, *Journal of Machine Learning Research* (2021) (cit. on p. 91).
- [279] B. Dhingra et al., *Handling Divergent Reference Texts when Evaluating Table-to-text Generation*, arXiv preprint arXiv:1906.01081 (2019) (cit. on pp. 91, 92).
- [280] M. Post, *A Call for Clarity in Reporting BLEU Scores*, arXiv preprint arXiv:1804.08771 (2018) (cit. on p. 92).
- [281] R. Vedantam, C. Lawrence Zitnick and D. Parikh, “Cider: Consensus-based Image Description Evaluation”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015 (cit. on pp. 92, 126).
- [282] P. Anderson, B. Fernando, M. Johnson and S. Gould, “Spice: Semantic Propositional Image Caption Evaluation”, *Proceedings of European Conference on Computer Vision*, 2016 (cit. on pp. 92, 126).
- [283] P. Laban, T. Schnabel, P. N. Bennett and M. A. Hearst, *SummaC: Re-Visiting NLI-based Models for Inconsistency Detection in Summarization*, *Transactions of the Association for Computational Linguistics* **10** (2022) (cit. on p. 92).
- [284] W. Johnson, *Studies in Language Behavior: A Program of Research*, *Psychological Monographs* **56** (1944) (cit. on p. 92).
- [285] C. E. Shannon, *A Mathematical Theory of Communication*, *The Bell System Technical Journal* **27** (1948) (cit. on p. 92).
- [286] J. Li, M. Galley, C. Brockett, J. Gao and B. Dolan, *A Diversity-Promoting Objective Function for Neural Conversation Models*, arXiv preprint arXiv:1510.03055 (2015) (cit. on p. 92).
- [287] J. Schulman, P. Moritz, S. Levine, M. Jordan and P. Abbeel, *High-dimensional Continuous Control using Generalized Advantage Estimation*, arXiv preprint arXiv:1506.02438 (2015) (cit. on p. 92).
- [288] P. Ammanabrolu, *Language Learning in Interactive Environments*, PhD thesis: Georgia Institute of Technology, 2021 (cit. on p. 93).
- [289] S. Huang and S. Ontañón, *A Closer Look at Invalid Action Masking in Policy Gradient Algorithms*, arXiv preprint arXiv:2006.14171 (2020) (cit. on p. 93).
- [290] A. Holtzman et al., *Learning to Write with Cooperative Discriminators*, arXiv preprint arXiv:1805.06087 (2018) (cit. on p. 93).
- [291] M. Andrychowicz et al., “Hindsight Experience Replay”, *Proceedings in Advances in Neural Information Processing Systems*, ed. by I. Guyon et al., vol. 30, 2017 (cit. on p. 93).

-
- [292] A. Maas et al., “Learning Word Vectors for Sentiment Analysis”, *Proceedings of Annual Meeting of ACL: Human language technologies*, 2011 142 (cit. on p. 95).
- [293] B. Y. Lin et al., “CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning”, *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020 (cit. on pp. 95, 126).
- [294] A. Parikh et al., “ToTTo: A Controlled Table-To-Text Generation Dataset”, *Proceedings of Conference on Empirical Methods in Natural Language Processing*, 2020 (cit. on pp. 95, 143).
- [295] O. Bojar et al., “Findings of the 2016 Conference on Machine Translation”, *Proceedings of the First Conference on Machine Translation*, 2016 (cit. on p. 95).
- [296] T. Kočiský et al., *The NarrativeQA Reading Comprehension Challenge*, *Transactions of the Association for Computational Linguistics* **6** (2018) (cit. on pp. 95, 150).
- [297] Y. Li et al., “DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset”, *Proceedings of International Joint Conference on Natural Language Processing*, 2017 (cit. on pp. 95, 158).
- [298] C. Raffel et al., *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer*, *Journal of Machine Learning Research* **21** (2020) (cit. on pp. 99, 132).
- [299] V. Sanh, L. Debut, J. Chaumond and T. Wolf, *DistilBERT, a distilled version of BERT: Smaller, Faster, Cheaper and Lighter*, arXiv preprint arXiv:1910.01108 (2019) (cit. on pp. 99, 118).
- [300] M. Hausknecht and N. Wagener, *Consistent Dropout for Policy Gradient Reinforcement Learning*, arXiv preprint arXiv:2202.11818 (2022) (cit. on p. 100).
- [301] B. Jacob et al., “Quantization and Training of Neural Networks for Efficient Integer-arithmetic-only Inference”, *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2018 (cit. on p. 102).
- [302] Z. Li et al., *Guiding Large Language Models via Directional Stimulus Prompting*, arXiv preprint arXiv:2302.11520 (2023) (cit. on p. 103).
- [303] W. Xiao, Y. Xie, G. Carenini and P. He, *ChatGPT-steered Editing Instructor for Customization of Abstractive Summarization*, arXiv preprint arXiv:2305.02483 (2023) (cit. on p. 103).
- [304] A. F. Akyürek et al., *RL4F: Generating Natural Language Feedback with Reinforcement Learning for Repairing Model Outputs*, arXiv preprint arXiv:2305.08844 (2023) (cit. on p. 103).
- [305] P. Zhou et al., “I Cast Detect Thoughts: Learning to Converse and Guide with Intents and Theory-of-Mind in Dungeons and Dragons”, 2022 (cit. on p. 103).

Bibliography

- [306] Z. Wu et al.,
Fine-Grained Human Feedback Gives Better Rewards for Language Model Training, 2023,
arXiv: 2306.01693 [cs.CL] (cit. on p. 103).
- [307] I. Loshchilov and F. Hutter, *Decoupled Weight Decay Regularization*,
arXiv preprint arXiv:1711.05101 (2017) (cit. on p. 117).
- [308] S. Liu, Z. Zhu, N. Ye, S. Guadarrama and K. Murphy,
“Improved Image Captioning via Policy Gradient Optimization of Spider”,
Proceedings of the International Conference on Computer Vision, 2017 (cit. on p. 126).
- [309] D. Khashabi et al., “UNIFIEDQA: Crossing Format Boundaries with a Single QA System”,
Findings of Association for Computational Linguistics, 2020 (cit. on pp. 150, 152).
- [310] M. Cettolo et al., “Overview of the IWSLT 2017 Evaluation Campaign”,
Proceedings of International Conference on Spoken Language Translation, 2017
(cit. on p. 154).
- [311] Y. Liu et al., *Roberta: A Robustly Optimized Bert Pretraining Approach*,
arXiv preprint arXiv:1907.11692 (2019) (cit. on p. 159).

List of Figures

2.1	ESN: A sketch of an echo state network consisting of three layers: input layer, reservoir and output layer. The weights of connections from input to reservoir and inter-connections of the reservoir are generated randomly and kept fixed. Only the weights of connections from the reservoir to the output nodes are trained for the given task.	14
3.1	Key sensitivity: (a),(e) original images; (b),(f) decrypted images using the same key (echo state network) as used for encryption; both decrypted images are identical to the originals; (c),(d),(g),(h) decrypted images using slightly modified keys, i.e. slightly modified echo state networks; here, all decrypted images differ considerably from the original images.	22
3.2	Plaintext sensitivity: (a),(f) original images; (b),(g) encrypted images; (c),(h) original images with 1% of their pixels randomly distorted; (d),(i) encryptions of the modified images; (e),(j) difference between encrypted original and encrypted modified images (33.22% and 37.78%, respectively).	23
3.3	Ciphertext sensitivity: (a),(b) plaintext distribution of the Lena image and corresponding ciphertext distribution; (c),(d) plaintext distribution of the cat image and corresponding ciphertext distribution. Since the ciphertext distributions are almost identical, frequency analysis is difficult and the system is robust against ciphertext-only attacks.	23
3.4	Encryption and Decryption run times: Run times for encryption and decryption for different message sizes. It is observed that run times grow linearly with data sizes indicating the scheme is both practical and scalable	24
4.1	Ablation: (a) Influence of hyperparameters such as spectral radius, leaking rate and input scaling. (b) Influence of reservoir size on performance; the larger the reservoir, the better the performance. The fact that the bidirectional variant performs better than the uni-directional suggests that the task depends on context from both sides	30
5.1	Learning curves: (a),(c) Evolution of total episodic reward in learning deterministic policies for discrete versions of acrobot and mountain car. (b),(d) Evolution of total episodic reward in the learning of a softmax and Gaussian policy for discrete acrobot and continuous mountain car problems tasks, respectively. It is evident that SPSA variants perform better than RL methods	39

6.1	Motivating Experiments: Figure shows learning curves of classic ES and ES with novelty rewards such as Guided KNN and Guided AE (our approach) on different levels of MiniGrid environment. All of the methods solve the simplest variant with two rooms. However, With increased difficulty, we see that guided methods accelerate learning. With the hardest variant, ES fails to solve the task whereas guided methods can solve this task due to additional rewards	51
6.2	Novelty only learning curves: Average learning curves of agents that are trained using only novelty gradients	58
6.3	Novelty-guided learning curves: Average learning curves of agents that are trained using adaptive novelty-guided methods	59
6.4	Sparsity in NS methods: Effects of sparsity levels in pure novelty search methods. It is evident that sparse representations perform better than dense encodings in most settings.	60
6.5	Sparsity in Novelty-guided methods: Effects of sparsity levels in novelty guided methods. It is evident that sparse representations perform better than dense encodings in most settings.	61
6.6	Ablation analysis: The plots (a),(b) show the effect of sequence length on the performance for KNN and AE methods. Similarly, plots (c),(d) show the effect of archive size on the performance of KNN and AE methods. It is observed that AE scales well to longer sequences and it requires only a small archive set	62
7.1	Reacher environment: A simple three-segmented robot arm with a randomized target. The goal is to plan a sequence of actions using feedback and rewards so that the tip of the arm touches the target.	66
7.2	Robotic arm: A robotic manipulator consisting of three segments denoted as \vec{L}_1 , \vec{L}_2 and \vec{L}_3 . The relative angles at the joints are denoted as α , β , and γ . The vector pointing to the end-effector (in green) is described as \vec{R}	67
7.3	MMC architecture: A sketch of MMC architecture described in this paper; it consists of components for MMC Net sandwiched by forward and inverse transformation components. The computed joint angles are fed back into the system until the end effector relaxes to the desired target	68
7.4	Learning curves: Evolution of total episodic reward in the learning of Reacher task with two variants; (a) Variant I: in which target position as given in the observation and (b) Variant II: in which target position is not directly available, but as a feedback signal in the observation. As observed, MMC networks outperform end-end approaches in both variants.	70
7.5	Agent behaviors: Initial and final arm positions are shown in orange and red respectively; (a), (d) trajectories toward the target with MLP agent; (b), (e) trajectories toward the target with RNN agent; and (c) and (f) with MMC agent. The MMC agent quickly approaches the target and is able to maintain its position around the target throughout the episode	71

7.6	Sample Complexity: (a) Evolution of reward at each step averaged over 100 runs; showing MMC networks quickly relax to high reward gaining trajectories as soon as an episode starts. (b) a summary of total interactions with the environment to learn an optimal policy with different networks and variants of the Reacher task; MMC networks are 10 times sample-efficient than end-end approaches	72
8.1	NLPGym Environments: An overview of sample episodic interactions of agent-environments for three different tasks. (a) tagging a sequence with NER tags; (b) answering multiple-choice questions; c) generating label sequence for a given sentence	81
8.2	Learning Curves: Averaged learning curves of PPO and DQN agents with variations. (a),(d) learning curves for sequence tagging with NER and POS tags. (b),(e) MLC with Reuters and AAPD datasets; (c), (f) QA tasks with QASC and AIRC datasets . .	84
9.1	Natural Language Policy Optimization (NLPO) in the case of sentiment-guided continuation. Here, the LM (i.e., the policy) needs to produce a positive sentiment continuation given a review prompt (we cover other models of human preference in Sec. 9.2.2). Two objectives are balanced: 1) an automated proxy of human preference that serves as a reward (here: a sentiment classifier); and 2) “naturalness” as measured by a KL divergence from an LM not trained with explicit human feedback. The plots show validation learning curves comparing our NLPO to the popular policy gradient method PPO. (Top plot:) RL methods can easily achieve high reward if the KL penalty is removed, (Bottom:) but at the cost of higher perplexity. NLPO+KL, our proposed approach, succeeds in balancing reward and naturalness more effectively than prior work.	89
9.2	GRUE results: Summarized results via automated metrics across all 7 GRUE tasks for each of the 5 algorithms we consider, and human participant studies for the 5 tasks suitable for human studies. Test results are averaged over all the respective metrics seen in Table 9.1.	96
A.1	Visualization of MC policy: Visualization of a Gaussian policy learned for the mountain car task.	106
B.1	Class diagram of NLPGym:	110
C.1	Results summary: Summarized results via automated metrics across all 7 GRUE tasks for each of the 5 algorithms we consider, and human participant studies for the 5 tasks suitable for human studies. We break up the metrics into task-specific, e.g. average positive sentiment for IMDB task, and naturalness metrics, such as perplexity and human perceived coherence for the human rated metrics. This plot differs from Figure 9.2 as this one averages over multiple reward functions per each task.	118

C.2 **Learning Curves:** Averaged learning curves over 5 different runs by varying target KL, shaded regions indicate one standard deviation. (a) shows the rollout episodic total reward during training (b) shows evolution of sentiment scores on the validation split (c) shows evolution of perplexity on the validation split. From (a) and (b), it is seen that higher target KL (0.2) is desired to achieve higher rewards. However, this setting drifts away from the original LM too much and loses fluency. Therefore a lower target KL (0.02 or 0.05) is required to keep the model closer to original LM. Similar trends hold for NLPO but when compared to PPO, it retains lower perplexities and is more stable even with higher KL targets, enabling higher sentiment scores. . . . 120

C.3 **IMDB Human Study Setup:** Instructions, example, and interface for the IMDB sentiment completion task. 123

C.4 **CommonGen Human Study Setup:** Instructions, examples, and interface for the Commongen task. 131

C.5 **CommonGen Pairwise Study Setup:** Instructions and interface for the pairwise Commongen HIT. 133

C.6 **CNN/DM Human Study Setup:** Instructions and interface for the summarization task. 138

C.7 **ToTTo Human Study Setup:** Instructions, two examples, and interface for the ToTTo table description task. 149

C.8 **DailyDialog Human Study Setup:** Instructions and interface for the Daily Dialogue task. 164

List of Tables

3.1	ESN configuration: An overview of hyperparameters that are used for the security analysis of the proposed cryptography system. The input sequences are split into chunks of $m = 200$ bytes and reservoir size is set to $0.95 \times m$	21
4.1	NER on Word Embeddings: Comparison of performance of different models using non-contextualized word embeddings as inputs. Even with simple word embeddings as inputs, reservoir representations obtained from Bi-ESNs are very competitive to fully trained Bi-LSTMs.	31
4.2	NER on Flair + Word Embeddings: Comparison of performance of different models using stacked flair and word embeddings. Although the resulting stacked embedding is contextualized, reservoir representations obtained using Bi-ESNs still provide a considerable performance gain over original stacked embeddings	32
5.1	Performance summary: Evaluation results containing average episodic total reward in the last 100 iterations of policy learning on classic problems for different variants and their baselines (the higher the value, the better the performance)	40
7.1	Performance Comparison: Final Performance in terms of total episodic reward (higher is better) of best policies with different networks averaged over 100 episodes; MMC nets outperform other end-end architectures	72
8.1	Test Performance: Summary of performance of selected DQN and PPO agents on the test set. For ST and MLC, micro F1-scores are reported, and for QA, accuracy is reported	85
9.1	GRUE Benchmark using RL4LMs showing the various tasks, input and output types, and the metrics used. We note that we test RL algorithms on these tasks for a wider range of possible rewards than just the task specific ones shown here. Unless specified, datasets are in English.	95
9.2	Key questions answered using GRUE + RL4LMs: This table summarizes the results found in the ablations and Fig. and provides an overview of the questions we ask in Section 9.4: which tasks require warm starts or are easily reward hackable; when to use RL over Supervised, when to use both; and when to use NLPO over PPO. All conclusions drawn are the result of statistical analysis as discussed in the experimental setup.	97
9.3	IMDB Ablation Results	97

A.1 **Summary of Hyperparameters:** Hyperparameters and their values for different experiments. 105

C.1 **IMDB Hyperparams:** Table shows a list of all hyper-parameters and their settings . 119

C.2 **Target KL Ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. It is seen from perplexity scores that a lower target KL constraint is desired to keep the model closer to the original model. On the otherhand, a higher target KL yields higher sentiment scores at the cost of fluency. inf KL penalty (target KL of inf), model simply learns to generate positive phrases (eg: "I highly recommend this movie to all!", "worth watching") regardless of the context 121

C.3 **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, data budget ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table measures performance differences as a function of the fraction of the dataset that has been used. In the case of the RL approaches, this measures how much data is used to train the reward classifier, and for the supervised method it directly measures fraction of positive reviews used for training. We note that using even a small fraction of data to train a reward classifier proves to be effective in terms of downstream task performance while this is not true for supervised approaches. This lends evidence to the hypothesis that adding expending data budget on a reward classifier is more effective than adding more gold label expert demonstrations. 121

C.4 **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, discount factor ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table measures performance differences for the discount factor. We note that most NLP approaches using RL follow the style of [240, 259] and use a discount factor of 1. This is equivalent to reducing the generation MDP to a bandit feedback environment and causes performance loss (in the case of NLPO) and reward hacking and training instability (in the case of PPO). 122

C.5 **Evaluation of GPT2 with different algorithms on IMDB sentiment text continuation task, NLPO hyperparameter ablations:** Mean and standard deviations over 5 random seeds is reported for sentiment scores along with fluency and diversity metrics. This table shows results of NLPO’s stability to the unique hyperparameters introduced in the algorithm - all other parameters held constant from the best PPO model. The number of iterations after which the masking model syncs with the policy and the top-p nucleus percentage for the mask model itself. We see that in general, the higher the top-p mask percentage, the better the performance. For target update iterations, performance is low if the mask model is not updated often enough or if it updated too often. 122

C.6 **IMDB Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorf’s alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 300 data points per algorithm. 122

C.7	IMDB Human Study Tukey Results: Results of a post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding p -values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall p -values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.	124
C.8	CommonGen Hyperparams: Table shows a list of all hyper-parameters and their settings	127
C.9	CommonGen test evaluation Table shows official scores obtained from CommonGen hold-out evaluation. The most important result is that RL fine-tuning on a supervised model yields better performance across most metrics especially Coverage which indicates the ratio of concepts covered in generated texts	128
C.10	CommonGen - Lexical and Semantic Metrics on dev set: Table shows lexical and semantic for best performing models found in each algorithm-reward function combinations along with best performing supervised baseline models. Generated text from these models are submitted to official CommonGen test evaluation to obtain test scores presented in Table C.9	128
C.11	CommonGen - Diversity Metrics on dev set: Table shows diversity metrics for best performing models found in each algorithm-reward function combinations along with best performing supervised baseline models. Generated text from these models are submitted to official CommonGen test evaluation to obtain test scores presented in Table C.9	129
C.12	CommonGen Human Study Results: Results of the human subject study showing the number of participants N , average Likert scale value for coherence and sentiment, Krippendorf’s alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 300 data points per algorithm.	129
C.13	CommonGen Human Study Tukey Results: Results of a post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding p -values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall p -values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.	130
C.14	CNN/DM Hyperparams: Table shows a list of all hyper-parameters and their settings	134
C.15	CNN/Daily Mail test evaluation (Lexical and Semantic metrics): Table presents a wide range of metrics: lexical, semantic and factual correctness. As baselines, we report lead-3 which selects first three sentences as the summary, Zero-Shot and a supervised model. PPO and NLPO models are on par with supervised performance on several metrics including Rouge-2, Rouge-L, and Bleu. On fine-tuning on top of supervised model, performance improves consistently on all metrics indicating that RL fine-tuning is beneficial. Another interesting finding is that, RL fine-tuned models are factually consistent as measured by SummaCZS metric.	135
C.16	CNN/Daily Mail test evaluation (Diversity Metrics): Table presents a wide range of diversity metrics on test set	135

C.17 **PPO Ablation/Model Selection:** Evaluation of PPO models on validation set with different reward functions and top k values for rollouts. For each alg-reward combo, best model (top k) is chosen. 136

C.18 **NLPO Ablation/Model Selection:** Evaluation of NLPO models on validation set with different reward functions, top k values for rollouts and target update iterations. For each alg-reward combo, best model is chosen 136

C.19 **CNN/DM Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorf’s alpha showing inter-annotator agreement, and Skew. For each model a total of 50 samples were drawn randomly from the test set and rated by 3 annotators each, each resulting in 150 data points per algorithm. 137

C.20 **CNN/DM Human Study Tukey Results:** Results of an post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding *p*-values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall *p*-values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment. 137

C.21 **ToTTo Hyperparams:** Table shows a list of all hyper-parameters and their settings . 143

C.22 **ToTTo dev evaluation (Lexical, Semantic and Factual Consistency Metrics):** Table shows lexical, semantic and factual correctness metric scores of algorithms with different reward functions on dev set. Without supervised pre-training, both PPO and NLPO results in sub-optimal solutions, with NLPO better than PPO. With supervised pre-training, PPO and NLPO achieve better scores across all metrics showing RL fine-tuning is beneficial. Most importantly, RL fine-tuned models produce more factually correct text as seen in higher PARENT scores. Another observation, fine-tuning with a task-specific metric PARENT is better than training just on task-agnostic lexical metrics 144

C.23 **ToTTo dev evaluation (Diversity Metrics) :** Table shows diversity metrics of algorithms with different reward functions on dev set. 145

C.24 **ToTTo test evaluation:** Table shows lexical, semantic and factual correctness metric scores of algorithms with different reward functions on hold-out test set. Without supervised pre-training, both PPO and NLPO results in sub-optimal solutions, with NLPO better than PPO. With supervised pre-training, PPO and NLPO achieve better scores across all metrics showing RL fine-tuning is beneficial. Most importantly, RL fine-tuned models produce more factually consistent text as seen in higher PARENT scores. Another observation, fine-tuning with a task-specific metric PARENT is better than training on task-agnostic lexical rewards 146

C.25 **ToTTo Human Study Results:** Results of the human subject study showing the number of participants N, average Likert scale value for coherence and sentiment, Krippendorf’s alpha showing inter-annotator agreement, and Skew. For each model a total of 50 samples were drawn randomly from the test set and rated by 3 annotators each, resulting in 150 data points per algorithm. 146

C.26	ToTTo Human Study Tukey Results: Results of a post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding p -values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall p -values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.	147
C.27	NarQA Hyperparams: Table shows a list of all hyper-parameters and their settings .	151
C.28	Evaluation of NarrativeQA (Lexical and Semantic Metrics): Reference Metrics, supervised is based on UnifiedQA [309].	152
C.29	Evaluation of NarrativeQA (Diversity Metrics): Reference Metrics, supervised is based on UnifiedQA [309].	152
C.30	NMT Hyperparams: Table shows a list of all hyper-parameters and their settings . .	155
C.31	WMT-16 and IWSLT test evaluation - lexical and semantic: Table shows lexical, semantic metrics for RL algorithms with different reward functions bench-marked against supervised baseline models	156
C.32	WMT-16 and IWSLT test evaluation - diversity metrics Table shows diversity metrics for RL algorithms with different reward functions bench-marked against supervised baseline models	157
C.33	DailyDialog Hyperparams: Table shows a list of all hyper-parameters and their settings	160
C.34	Evaluation of Daily Dialog: Table shows lexical, semantic metrics for RL algorithms bench-marked against supervised baseline models	161
C.35	Evaluation of Daily Dialog: Table shows diversity metrics for RL algorithms bench-marked against supervised baseline models	161
C.36	DailyDialog Human Study Results: Results of the human subject study showing the number of participants N , average Likert scale value for coherence and sentiment, Krippendorf's alpha showing inter-annotator agreement, and Skew. For each model a total of 100 samples were drawn randomly from the test set and rated by 3 annotators each, each resulting in 300 data points per algorithm.	161
C.37	DailyDialog Human Study Tukey Results: Results of a post-hoc Tukey HSD Test for difference in means between pairs of algorithms (Group 2 - Group 1) and corresponding p -values. Individually statistically significant results are bolded and are used to discuss results in the analysis. Overall p -values showing that there is a significant difference in means between the models via a one-way ANOVA test are significant with $p \ll 0.05$ for both coherence and sentiment.	162

Publications

This thesis is based on the following publications (in chronological order) :

1. R. Ramamurthy, C. Bauckhage, K. Buza and S. Wrobel, “Using Echo State Networks for Cryptography”, *Proceedings of International Conference on Artificial Neural Networks*, 2017, URL: https://doi.org/10.1007/978-3-319-68612-7_75
2. R. Ramamurthy, C. Bauckhage, R. Sifa and S. Wrobel, “Policy Learning Using SPSA”, *Proceedings of International Conference on Artificial Neural Networks*, 2018, URL: https://doi.org/10.1007/978-3-030-01424-7_1
3. R. Ramamurthy, C. Bauckhage, R. Sifa, J. Schücker and S. Wrobel, “Leveraging Domain Knowledge for Reinforcement Learning Using MMC Architectures”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30484-3_48
4. R. Ramamurthy, R. Stenzel, R. Sifa, A. Ladi and C. Bauckhage, “Echo State Networks for Named Entity Recognition”, *Proceedings of International Conference on Artificial Neural Networks*, 2019, URL: https://doi.org/10.1007/978-3-030-30493-5_11
5. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Novelty-Guided Reinforcement Learning via Encoded Behaviors”, *Proceedings of International Joint Conference on Neural Networks*, 2020, URL: <https://doi.org/10.1109/IJCNN48605.2020.9206982>
6. R. Ramamurthy, R. Sifa, M. Lübbering and C. Bauckhage, “Guided Reinforcement Learning via Sequence Learning”, *Proceedings of International Conference on Artificial Neural Networks*, 2020, URL: https://doi.org/10.1007/978-3-030-61616-8_27
7. R. Ramamurthy, R. Sifa and C. Bauckhage, “NLP Gym – A Toolkit for Evaluating RL agents on Natural Language Processing Tasks”, *Proceedings of Wordplay Workshop in NeurIPS 2020*, URL: <https://doi.org/10.48550/arXiv.2011.08272>
8. R. Ramamurthy, P. Ammanabrolu, K. Brantley, J. Hessel, R. Sifa, C. Bauckhage, H. Hajishirzi and Y. Choi, “Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization”, *Proceedings of International Conference on Learning Representations*, 2023, URL: <https://openreview.net/forum?id=8aHzds2uUyB>