Reinforcement Learning For Query-based Multi-document Extractive Summarisation

By

Christopher Rhys Jones

A thesis submitted to Macquarie University
for the degree of
Masters of Research
Department of Compuuting
October 2019

MACQUARIE
University
SYDNEY · AUSTRALIA

Examiner's Copy

Typeset in LaTeX 2_ε.

Statement of Originality

This work has not previously been submitted for a degree or diploma in any university. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

(Signed) Date: 23/10/2019

Christopher Rhys Jones

Acknowledgements

I would like to thank my supervisor Dr. Diego Mollá Aliod for his constant support throughout my project, and for always making the time to give advice and provide comments for this thesis. This thesis would not have been possible without his guidance and support.

I would also like to thank my family and close friends for their support and prayers throughout my project, and for offering to help in so many ways.

List of Publications

Parts of this thesis extend on some preliminary work reported in the publication below.

- Diego Mollá, Christopher Jones. *Classification Betters Regression in Query-based Multi-document Summarisation Techniques for Question Answering: Macquarie University at BioASQ7b*. (Proceedings of BioASQ Workshop 2019), 2019.

Abstract

Text summarisation helps to manage the growth of digitally stored textual information, by allowing users to learn key information from reading short summaries. This research project focuses on query-based multi-document extractive summarisation, which constructs a summary made of sentences extracted directly from multiple source documents and based on a user query. Much of the past research in extractive summarisation is based on supervised machine learning approaches, which requires converting target human summaries into explicit annotations of the input sentences. In contrast, our research focuses on reinforcement learning, which can incorporate the target human summaries directly into the learning process. We explore the impact of various key aspects of reinforcement learning. First, we compare several variants of the Proximal Policy Optimization (PPO) approach with baseline reinforcement learning approaches. Second, we investigate pre-training our policy using supervised approaches. We report our results on data provided by the BioASQ Challenge. We observe that PPO penalises changes to the policy as mentioned in literature. However, there is no significant improvement to our summarisation quality when using PPO or pre-training.

Contents

List of Figures

List of Tables

1 Introduction

Text is a valuable means of communicating information, whether it is conveyed using paper, computers, mobile devices, or any other means. More and more textual information is being stored digitally, and new information is uploaded to the internet all the time. However, too much information can cause difficulties for users, and can make key information harder to find. One of many examples is a doctor who wants to find references related to their diagnosis, in order to conduct Evidence Based Medicine. Another example is a reader who wants to find a particular news story. In both of these cases, it is inefficient for the user to read every piece of information relevant to their topic to find what they are looking for.

Text summarisation helps to manage this growth of digitally stored text [38]. Text summarisation provides a short summary of the source texts, which allows the user to quickly identify key information about the texts. Text summarisation approaches can be divided into extractive summarisation and abstractive summarisation. This thesis focuses on extractive summarisation — where sentences from the source document are re-used directly in the summary, as opposed to abstractive summarisation where the summary text is different from the original sentences.

Past research in text summarisation includes Pre-Machine Learning, Machine Learning, and Deep Learning approaches (See Section 2.2). Instead, our research project uses reinforcement learning approaches with the goal of generating human-like summaries. One well known application of reinforcement learning is AlphaGo, which was developed by the Google DeepMind team to achieve superhuman performance in the board game Go [70]. Another application is the OpenAI reinforcement learning library which we use in this thesis, where robots are trained to walk using the OpenAI Gyms [3, 68]. These applications of reinforcement learning aimed to either mimic or outperform human behaviour.

Reinforcement learning has also been applied to extractive summarisation in past research [22, 30, 47, 60, 82]. Reinforcement learning is used instead of supervised machine learning approaches because it can incorporate the target human summaries directly into the learning process. In contrast, supervised approaches require explicit annotations of which sentences are the correct ones to extract and use in the summary. Reinforcement learning is also an approach which can predict the long-term outcome of choosing a sentence. These benefits make reinforcement learning an attractive research topic for summarisation tasks. However, many past reinforcement learning summarisation systems use only the vanilla reinforcement learning approaches with minimal modifications (See Section 2.3).

We apply a new reinforcement learning approach known as Proximal Policy Optimization (PPO) [68], which has not yet been applied to query-based multi-document extractive summarisation to our knowledge. We use PPO because it performs well for Atari games, and can be applied to vanilla policy gradients with only a few lines of code change [68]. We address the research questions described in Section 1.1, and make the contributions described in Section 1.2.

1.1 Aims

Our long-term aim is to improve the summary quality of our reinforcement learning summarisation system by changing the policy it uses. However, the short-term aim of this thesis is to firstly understand the policy used in Proximal Policy Optimization (PPO) reinforcement learning, and then to discover the impact of applying PPO and pre-training approaches to our summarisation task. In doing so, we discover if PPO can improve our summarisation quality, and what modifications have an impact on the performance of the PPO approach.

We consider the following 2 research questions in this thesis.

1. Can PPO be used to improve the summarisation quality of our query-based multi-document extractive summarisation task?

2. Does pre-training reinforcement learning to avoid learning random sequences also improve the summarisation quality?

We divide our first research question into three necessary parts. The necessary parts to answer this question are a **PPO implementation**, a **measure of summarisation quality** and a **summarisation task**. After obtaining the necessary parts (described in Chapter 3), we perform experiments to answer our first research question (described in Chapter 4). We compare our PPO implementation with several baseline systems including a vanilla REINFORCE policy-based reinforcement learning approach [78]. We observe that PPO penalises changes to the policy as mentioned in literature [68]. However, we do not observe a significant improvement in summarisation quality when using PPO.

For our second research question we use the same three necessary parts, and add a pre-training stage to our existing systems. Chapter 5 describes the results of our experiments using pre-training for PPO, and our baseline REINFORCE approach. Pre-training does not significantly improve our results either, and we can investigate ways to improve our pre-training approach in future research.

1.2 Contributions

This thesis applies the Proximal Policy Optimization (PPO) Actor-Critic reinforcement learning approach to our text summarisation task. Past research in summarisation applies vanilla reinforcement learning approaches to text summarisation, but few of the Actor-Critic approaches have been applied to text summarisation. PPO was published in 2017 [68], and has not yet been applied to query-based multi-document extractive summarisation to our knowledge. Our contribution includes applying PPO to this summarisation task, and applying modifications and pre-training to our PPO implementation.

In addition, I contributed to a paper submitted to the BioASQ Workshop 2019 [48]. In the paper, I contributed to the training of Neural Regression and Neural Classification systems for Batch 1-3 only of the BioASQ competition 2019. I also contributed to adding word embeddings to the REINFORCE approach for Batch 2 only. I also contributed to the scraping of human evaluations from the BioASQ results page, and calculating the Pearson correlation only for evaluation of the ROUGE-SU4 F1 metric. We explain our contribution to the BioASQ competition in 2019 in Section 3.2.

1.3 Outline

The rest of this thesis is organised as follows. Chapter 2 performs a literature review of reinforcement learning approaches used for summarisation. Chapter 3 describes our baseline reinforcement learning system which we submitted to the BioASQ shared task. Chapter 4 describes the PPO system which we compare with our baseline system. Chapter 5 describes the pre-training stage which we add to PPO and REINFORCE to begin training with a high-scoring model. Chapter 6 reports our results in all the systems that we explored. Chapter 7 concludes this thesis and suggests future research directions.

2 Literature Review

The communication of information using text has changed and developed throughout history, and originated from spoken languages [8]. Today, text is commonly used for electronic communication, and contributes to the rapid growth of information on the internet. Text Summarisation aims to deal with the wealth of textual information on the internet [38], by reducing large texts into smaller more readable summaries.

This chapter performs a literature review of text summarisation for query-based multi-document datasets. This is not a systematic review, and only covers some aspects of text summarisation which are relevant to this thesis. Chapter 32 of The Oxford handbook of computational linguistics (page 584) describes summarisation as follows [43]:

> Definition: a summary is a text that is produced from one or more texts, that contains a significant portion of the information in the original text(s), and that is no longer than half of the original text(s).

This definition includes both summarisation by machines, and by humans. However, the scope of this thesis is limited to machine learning and reinforcement learning approaches, and the summarisation tasks which they could be applied to.

The rest of this chapter is as follows. Section 2.1 describes the different types of summarisation, with specific focus on query-based multi-document summarisation. Section 2.2 describes the history of summarisation methods. Section 2.3 describes reinforcement learning, which is the focus of this thesis. Finally, Section 2.4 discusses the areas of reinforcement learning which can be applied to future research in query-based multi-document summarisation.

2.1 Text Summarisation

In this section, we discuss the different types of text summarisation, according to their inputs and outputs [38]. The inputs of a summarisation system can be categorised as either single documents or multiple documents. This thesis also includes query-based multi-document summarisation so that we can discuss this area in detail. The outputs of a summarisation system can be categorised as extractive, abstractive, or a hybrid of both [55, 82]. We define each of these categories below, and discuss the different research for each.

2.1.1 Single-document Input

Figure 2.1 shows what single-document extractive summarisation does. This summarisation approach takes a single document containing multiple sentences as input, and outputs a "summary" derived from the important information in the document. There are many single-document datasets which are listed on Ruder's GitHub Page[1], and in Dernoncourt et al. [9]. This subsection reviews only 3 examples of single-document datasets for Gigaword, CNN / Daily Mail, and arXiv / PubMed.

[1]https://github.com/sebastianruder/NLP-progress/blob/master/english/summarization.md

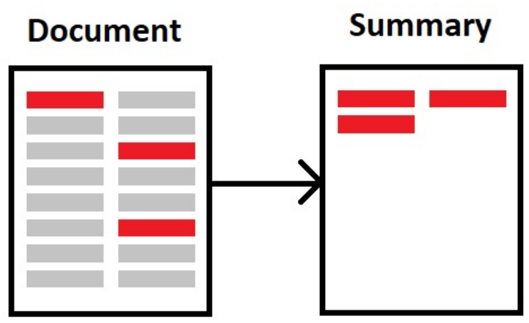


FIGURE 2.1: A diagram of single-document extractive summarisation. Important sentences in the document (shown in red) are extracted to create a shorter summary.

The **Gigaword** dataset contains 4,111,240 newswire text documents collected in English by the Linguistic Data Consortium [18]. At the time of writing, there are 18 papers using the Gigaword dataset on Ruder's GitHub Page[2]. Li et al. [31] uses the Gigaword dataset for their reinforcement learning model.

The **CNN / Daily Mail** dataset contains 312,084 news reports as single documents [5]. Originally the CNN dataset and Daily Mail dataset were two separate datasets [23], but Nallapati et al. [50] combines these two datasets and publishes anonymized and non-anonymized versions. At the time of writing, there are 12 papers using the anonymized dataset and 25 using the non-anonymized dataset listed on GitHub[2]. Zhang et al. [82] uses the CNN / Daily Mail dataset for their reinforcement learning model.

Cohan et al. [7] introduces a dataset which takes scientific papers from the arXiv[3] and PubMed[4] scientific repositories to be summarised. The arXiv dataset contains 215,000 documents, and the PubMed dataset contains 133,000 documents. This dataset is another example which can be used for single-document summarisation models.

[2]https://github.com/sebastianruder/NLP-progress/blob/master/english/summarization.md
[3]https://arxiv.org/
[4]https://www.ncbi.nlm.nih.gov/pubmed/

2.1.2 Multi-document Input

Multi-document summarisation takes input sentences from multiple documents, and produces one summary for all of the documents. Summarisation for multiple documents comes with its own set of challenges. For example, having multiple documents means that there are more sentences that can be chosen for extraction, and there is a greater chance of disfluency between unrelated sentences [54]. Some approaches to multi-document summarisation include clustering, maximal marginal relevance, and graph-based approaches [65]. One of the earliest systems for multi-document summarisation was SUMMONS (1998) [58]. We review several multi-document datasets below, with specific focus on the TAC / DUC dataset.

The **Document Understanding Conference** (DUC) datasets were released as a shared task from 2001 to 2007 [54]. The datasets published by DUC are widely used for multi-document summarisation [4]. In 2008, DUC was replaced by the Text Analysis Conference (TAC), which is still ongoing. The content of DUC and TAC datasets changes every year [9], however most include a multi-document summarisation challenge. Nallapati et al. [51] uses the DUC 2002 dataset to compare their results with other summarisation models using this dataset. The more recent datasets are only available to track participants.

The **Text Summarisation Challenge** (TSC) is another shared task which includes both single and multi-document datasets. [16] This dataset was part of a workshop at NTCIR (NII Test Collection for Information Retrieval Project), and contains newspaper articles in Japanese.

Multi-document datasets can also be generated from publicly available comments. The TGSum dataset is one example which collects tweets about a topic to gather more data than the DUC dataset [4]. The Opinosis dataset is another example which contains multiple reviews from users about certain topics [17].

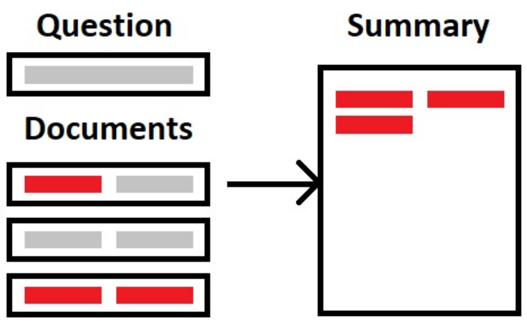


FIGURE 2.2: A diagram of query-based multi-document extractive summarisation. Summary sentences (shown in red) are extracted from multiple documents, and can be based on a question.

2.1.3 Query-based Multi-document Input

Figure 2.2 shows an example of what query-based multi-document summarisation does, where an input question is provided with multiple documents that may contain an answer. The summarisation system is expected to answer the provided question using key information from the documents. Below we describe the BioASQ dataset, which is used for both Question Answering and query-based multi-document summarisation.

BioASQ is a shared task which publishes query-based multi-document datasets [74]. The BioASQ challenge started in 2013, and includes biomedical questions and documents [46]. The BioASQ training data which we use is publicly available, but the human annotated answers for the test data are not released until after the competition is completed. Participating systems receive results on the test data by uploading the machine generated summaries to the BioASQ website.

Summarisation techniques perform well in the BioASQ competitions [46]. This is because the gold standard human answers released by BioASQ contain both an **ideal answer** and an **exact answer**. The exact answer is generally one word, and can be generated using Question Answering techniques [84]. The ideal answer contains several sentences, and often repeats the same sentences that are in the source documents. Summarisation techniques are more geared to generating ideal answers, and perform well for this task [46].

2.1.4 Extractive and Abstractive Outputs

This subsection briefly describes the outputs of a summarisation system. The outputs of a summarisation system can be categorised as extractive, abstractive, or a hybrid of both [55, 82]. Extractive summarisation re-uses sentences that already exist in the source document again in the summary paragraph. Abstractive summarisation creates a new paragraph that doesn't exist in the document but is like the correct summary which was annotated by a human. Generally, extractive summarisation has been more successful than abstractive summarisation — which is not guaranteed to generate legible sentences [51]. There is also a risk that abstractive summarisation will introduce false or irrelevant information which was not included in the original text.

Extractive summarisation is used by some papers to produce results [22, 30, 52, 60, 82]. Narayan et al. [52] report their results using a purely extractive summariser with reinforcement learning. This thesis only performs extractive summarisation, and we consider abstractive summarisation a possible direction for future research.

Abstractive models are also becoming increasingly popular, and are sometimes compared with similar extractive approaches [82]. Rush et al. [62] uses an attention model to generate abstractive summaries from a Neural Network. Zhang et al. [82] uses a Sequence-2-Sequence model to generate abstractive summaries. Hybrid models can also be used to combine extractive and abstractive approaches into a single model [12]. Future research into these approaches could be used to further improve text summarisation models.

2.2 Past Approaches

There are many methods used to perform extractive and abstractive summarisation. We divide these approaches into Pre-Machine Learning, Machine Learning, Deep Learning, and then Reinforcement Learning (Section 2.3). More detail on early approaches to summarisation can be found in the literature review by Lloret and Palomar [38]. This thesis focuses on the reinforcement learning approach to extractive summarisation (Section 2.3).

2.2.1 Pre-Machine Learning

Before Machine summarisation, Luhn [40] proposed using word frequency in a document to find sentences to summarise it. Edmundson [14] also proposed an early method of summarising before machine learning techniques. This was termed as the Edmundsonian Paradigm, in which sentences are ranked for extraction based on a linear function.

Edmundson proposed that using the cue method, key method, location method and title method aid the identification of good summary sentences. Newer summarisation approaches build on this idea that sentence content, location and/or headings are important features for the summarisation of a document. Machine Learning approaches are one such example which has gained inspiration from these early summarisation approaches.

2.2.2 Machine Learning

This subsection describes some machine learning approaches used for summarisation. The machine learning approaches can be divided into supervised and unsupervised machine learning (however we only discuss supervised machine learning here). Supervised machine learning can be further divided into Regression and Classification approaches. Some Regression approaches include Linear Regression, Logistic Regression and Regression Trees. Some Classification approaches include Naive Bayes, K-Nearest Neighbours and Support Vector Machines. There are many other examples of machine learning approaches which have been used for summarisation [38], which we will not list exhaustively. Below we describe some early examples of machine learning used in summarisation.

The Naive Bayes Classifier was used by early papers to perform summarisation [29, 65]. Ouyang et al. [53] applies Regression techniques to a query-based multi-document summarisation dataset (DUC 2005) to gain an improvement over Classification approaches. Graph-based approaches have also been used for query-based multi-document summarisation [83]. We do not provide an exhaustive list of Regression and Classification approaches, but provide these only as examples of machine learning summarisation approaches.

It is also possible to rank each word in a sentence and find the best one. This approach has been applied to query-based multi-document summarisation [49, 53, 66]. Words can be ranked using Term Frequency-Inverse Document Frequency (TFIDF), which ranks each word based on how often each word occurs in one document compared to other documents [64]. Alternatively, word features can be trained using a word embedding vector based on the meaning of each word [36, 42].

2.2.3 Deep Learning

This subsection describes the use of Deep Learning for summarisation tasks. Deep Learning has been applied to the field of summarisation, and involves training a neural network. Early neural network approaches were inspired by the network of neurons in the brain [41]. Neural Network models can be fit to complex non-linear functions, and the input features do not need to be specified in advance [37]. This means that deep learning can be used for a wide range of datasets, and can also be applied to unsupervised machine learning which does not require labels [27, 59]. Dong [13] performs a recent survey of deep learning models used for summarisation. This subsection briefly lists some Deep Learning models used for extractive summarisation.

Deep learning performs well for extractive summarisation as reported by SummaRuNNer [51] and Latent [82]. SummaRuNNer [51] performs extractive summarisation using GRU-RNNs (Gated Recurrent Unit — Recurrent Neural Networks) at the word and sentence layer. Latent [82] uses LSTM-RNNs (Long-Short Term Memory — Recurrent Neural Networks) for extractive summarisation, and performs reinforcement learning with a pre-trained neural network to avoid random sequences in their latent model. RBMs (Restricted Boltzmann Machines) are also shown to work for query-based multi-document extractive summarisation [35, 85]. Convolutional layers can be added to Neural Networks as well, and Narayan et al. [52] uses a Convolutional Encoder in their neural network to perform extractive summarisation.

2.3 Reinforcement Learning

This section describes the different types of reinforcement learning, and their applications to summarisation. The standard reinforcement learning framework follows a Markov decision process [72, 73], in which an agent explores the environment it is in. This process involves knowing the current **state**, performing an **action**, and then repeating actions to observe a delayed (long-term) **reward**.

State (s)	**Action (a)**	**Reward (r)**
Value Function		
V(s) =	-	Expected Future Reward
Q(s,a) =	-	Future Reward After Action a
Policy Function		
π(s) =	Predicted Action (Deterministic)	-
π(a|s) =	Probability of Action a (Stochastic)	-

TABLE 2.1: The value function and policy function notations, highlighting that a value function outputs a reward, and a policy function outputs an action (or probability).

Reinforcement learning approaches can be divided into Value-based, Policy-based and Actor-Critic. Table 2.1 briefly compares the common value and policy functions. Value based approaches use only a value function which predicts the future reward for a state, whereas policy-based approaches use only a policy function which predicts the action to take in a state. Actor-Critic approaches combine both value and policy functions, so that the policy function is trained by incorporating the value function in its loss.

Reinforcement learning is known to work well for Atari Games [1, 44], the board game GO [70, 75], and OpenAI Robot Gyms [32, 67, 68]. Atari Games are simulated video game environments, and OpenAI Gyms [3] are simulated real world environments which a robot learns to move in. Reinforcement learning has also been applied to both single-document summarisation [52, 55, 63, 82] and multi-document summarisation [47, 60].

Our project uses reinforcement learning for extractive summarisation, because it can optimize a summarisation system based on the target human summary. When using supervised machine learning approaches, an extractive summarisation system can only be trained to extract a sentence which has been annotated as a gold (correct) sentence.

However, by using reinforcement learning the system can be trained to extract different sentences — which will still have been the best choice after comparing all other extracted sentences with the human annotated summary.

In this section we describe the different types of Reinforcement Learning, and whether they have been applied to summarisation or not. Table 2.2 provides an overview of the individual approaches we discuss.

Type	**Approach**	**Used For**
Value-Based	**Q-Learning** (1989) [77]	Robots [34] **Summarisation** [22]
Value-Based	**DQN**: Deep Q Network	Atari Games [44] **Summarisation** [30]
Value-Based	**CDQN**: Continuous DQN	OpenAI Gym [19]
Policy-Based	**REINFORCE** (1992) [78]	Atari Games [6] **Summarisation** [82]
Actor-Critic	**SARSA**: State-Action-Reward-State-Action	OpenAI, Atari [71] **Summarisation** [60]
Actor-Critic	**DDPG**: Deterministic Policy Gradient	Add to Policy [69]
Actor-Critic	**TRPO**: Trust Region Policy Optimization	OpenAI, Atari [67]
Actor-Critic	**PPO**: Proximal Policy Optimization	OpenAI, Atari [68]
Actor-Critic	**A3C**: Asynchronous Advantage Actor-Critic	Atari Games [45]
Actor-Critic	**DDPG**: Deep Deterministic Policy Gradient	OpenAI Gym [32]
Actor-Critic	**ACER**: Actor-Critic with Experience Replay	Atari Games [76]
Actor-Critic	**SAC**: Soft Actor-Critic	OpenAI Gym [21]
Actor-Critic	**ACKTR**: Actor-Critic using Kronecker-factored Trust Region	OpenAI Gym [79]

Table 2.2: List of reinforcement learning approaches we reviewed, and citations to papers which apply them to Atari Games, OpenAI Gyms or Summarisation. Papers without Summarisation listed in the Used For column have not been used for Summarisation tasks yet to our knowledge.

2.3.1 Value-Based

$$V(s) = \text{Expected Future Reward} \tag{2.1}$$

A Value function (Equation 2.1) predicts the expected future **reward** of being in the current state. Every time the agent receives a reward, it trains the value function (which for DQN is a neural network) to output the reward that it got. In a value-based approach, the next action is chosen by considering the future reward for each possible action the agent can take, and choosing the action which has the highest predicted reward. Table 2.1 shows some common notations for value and policy functions.

Value-Based methods usually choose the action which results in the highest value at each step. As more actions are taken, the predictions of the value function (Equation 2.1) become more accurate, and the best action to take becomes clearer. We describe some value-based approaches to reinforcement learning below.

Q-Learning (1989) [77] is a model-free value-based approach, which usually uses an array to store values. Q-learning adds a Q value function to the value-based approach (See Table 2.1), which returns the value (future reward) based on the current state, and an action that can be taken in the current state. At each step, Q-learning considers all possible actions in the current state, and chooses the action with the highest value of $Q(s,a)$. Q-Learning has been applied to single-document and multi-document extractive summarisation [22]. Q-Learning has been applied to other tasks as well, such as programming robots [34].

Deep Q Networks (DQNs) improve Q-learning by using a Neural Network instead of an array, which allows more states to be approximated with less memory. Q-Learning and Deep Q Networks are most commonly used for video game environments [44]. DQN has also been applied to single-document extractive summarisation [30]. However, Q-Learning and DQN do not perform well in OpenAI Gyms, and are poorly understood [68].

Continuous DQN (called CDQN or NAF) is another approach which improves Deep Q Networks. This approach adds a Normalized Advantage Function (NAF) to Q-Learning so that it has continuous state and action spaces. Gu et al. [19] uses the Normalized Advantage Function in a Q-Learning approach to learn OpenAI Robot Gyms. Continuous DQN has not yet been applied to summarisation to our knowledge.

2.3.2 Policy-Based

$$\pi(s) = \text{Action} \tag{2.2}$$

A policy function predicts the **action** to take in the current state. A policy can be either deterministic — meaning that it outputs only a single action, or stochastic — meaning that it outputs a probability distribution which the next action is sampled from. Equation 2.2 is a simplistic interpretation of a deterministic policy, which is trained to directly output a single action to take. Stochastic policies have a different notation, and are trained to output the probability of the action instead. Table 2.1 shows the common notations for deterministic and stochastic policies.

Policy-Based approaches train the policy function (Equation 2.2) to map a state to an action that should be taken [1]. Approaches which use a policy function can be either on-policy or off-policy, depending on whether the action that is chosen comes directly from the policy function's predicted action or not [72] (DDPG and SAC are example off-policy Actor-Critic approaches). A loss function is used when training the policy function, which specifies how good the predictions made by the model are. Policy-Based approaches use the discounted future reward in the loss function to determine how good an action is which leads to that reward. In Mollá [47] the un-discounted final reward is back propagated to the loss function of each action, meaning that all actions are equally impacted by the final reward. Actor-Critic approaches also include the output of a value function in the loss function. We describe the vanilla REINFORCE policy-based algorithm below.

REINFORCE is a simple on-policy policy-based approach to reinforcement learning [78]. It is a vanilla policy-based approach with poor data efficiency and robustness [68]. For stochastic policies REINFORCE usually uses the Monte Carlo method to choose random actions from the probability distribution. REINFORCE is sometimes used as an Actor-Critic approach as well, simply by incorporating a value function baseline into the loss of the policy function. REINFORCE has been used for playing Atari Games [6], and for extractive summarisation [47, 82]. An Actor-Critic version of REINFORCE has also been used for single-document abstractive summarisation [31].

2.3.3 Actor-Critic

Actor-Critic approaches combine the policy gradient used in policy-based approaches with a value function similar to a value-based approach. As the name suggests, they include an "actor" part which updates the policy with the best actions, and a "critic" part which critiques how good the policy is using a value function. The value function is incorporated into the loss of the policy function, which allows it to penalise bad policies when it is trained well. This subsection describes some of the Actor-Critic approaches from Table 2.2.

Trust Region Policy Optimization (TRPO) [67] is an Actor-Critic approach published by John Schulman to learn OpenAI Robot Gyms. TRPO uses trust regions to prevent the model from learning policies which are too different from a trusted policy. Generative Adversarial Imitation Learning (GAIL) [25] is a method that improves the TRPO algorithm.

Proximal Policy Optimization (PPO) [68] improves the TRPO algorithm by adding clipping to the probability ratios. PPO has not been applied to summarisation tasks yet to our knowledge, but could be applied because it simply adds limits to the probability ratios of the policy function. We apply PPO to query-based multi-document extractive summarisation in this thesis.

State-Action-Reward-State-Action (SARSA) is an on-policy actor-critic version of Q-Learning, which updates the policy after taking each action. Rummery and Niranjan [61] first introduced SARSA as a footnote. Sutton [71] applies SARSA to several tasks from Puddle Games to OpenAI Gyms. SARSA has been applied to multi-document extractive summarisation as well, using the Temporal Difference (TD) method [60].

Asynchronous Advantage Actor-Critic (A3C) is another Actor-Critic approach [45]. A3C is used to solve Atari Games, Car Simulators and Physics Simulators. The Synchronous Advantage Actor-Critic (A2C) is an update on A3C to make it synchronous, which is faster and requires less runs. This has not been applied to summarisation yet to our knowledge.

Deep Deterministic Policy Gradient (DDPG) is an off-policy Actor-Critic approach that is based on Q-Learning, and is used in OpenAI Robot Gyms [32]. Distributed Distributional DDPG (D4PG) further improves DDPG to make it run in a distributed fashion [2]. Multi-agent DDPG (MADDPG) also improves DDPG [39]. Twin Delayed Deep Deterministic (TD3) applies the same techniques as DDPG, based on Q-Learning [15].

Actor-Critic with Experience Replay (ACER) is an Actor-Critic approach, used in Atari Games in the original paper [76]. So is Actor-Critic using Kronecker-factored Trust Region (ACKTR) [79], and Soft Actor-Critic (SAC) [21], which are both used in OpenAI Robot Gyms in the original papers. These methods have not been applied to summarisation tasks yet to our knowledge.

2.3.4 PPO

This subsection describes the Actor-Critic approach of PPO in more detail, as it is a major part of this thesis. Proximal Policy Optimization (PPO) was published in 2017 [68] by the author of TRPO [67]. We choose the PPO Actor-Critic approach for this thesis because it can be applied to a vanilla policy gradient implementation, with only a few lines of code change [68]. We know from past research that the REINFORCE policy-based approach can be applied to our summarisation task [47], so PPO could be applied to the same task. We therefore apply PPO to our query-based multi-document summarisation task, which has not been done before to our knowledge. We also applied the ACER approach to our task, but we were not able to adapt other Actor-Critic approaches (such as TRPO). In future research, we would like to apply the other Actor-Critic approaches in Table 2.2 to our summarisation task as well.

PPO performs clipping on the probability ratios similar to TRPO in order to prevent large changes to the actions predicted by the policy. Unlike TRPO, PPO imposes a CLIP function within the neural network, which is learned and adjusted to suit the model. The CLIP function simply adjusts the probability ratios output by a policy function, and can be applied to a vanilla policy gradient implementation [68]. We use the stable-baselines implementation of PPO [24], which uses a TensorFlow neural network to add the CLIP function.

We use the default settings for PPO as much as possible, and treat the original system as a black box for most of our experiments. We also make some small modifications to the PPO algorithm which we describe in Section 4.2.2. All our other results follow the original PPO algorithm, as presented in Schulman et al. [68].

2.4 Discussion

Table 2.2 lists the reinforcement learning approaches we discussed. Many Actor-Critic approaches listed in Table 2.2 have not yet been applied to summarisation tasks (with the exception of SARSA). The Deep Q Network value-based approach has only been applied to a single document dataset in Lee and Lee [30], and could be applied to query-based multi-document datasets. The PPO Actor-Critic approach could be applied to summarisation, because the CLIP function used in PPO can be applied to the a vanilla policy gradient implementation, with only a few lines of code change [68]. The vanilla REINFORCE approach has already been applied to extractive summarisation [47, 82], so PPO could be applied to the same task. In addition, many other Actor-Critic approaches could be applied to summarisation, which have not yet been attempted to our knowledge.

Several new Actor-Critic approaches have emerged since 2015 [21, 32, 45, 67, 68, 76, 79], showing that there is no single Actor-Critic approach which guarantees success. Instead, different approaches work for different problems, and applying these approaches to summarisation could improve our results. However, the Actor-Critic approach of combining Policy and Value functions seems to improve model performance over earlier approaches. We focus on the PPO Actor-Critic approach in this thesis, and could investigate other Actor-Critic approaches in future research.

3 Approach

This thesis aims to address two main research questions. The first research question we address is whether Proximal Policy Optimization (PPO) can improve the summarisation quality of reinforcement learning for a query-based multi-document extractive summarisation task. PPO has outperformed other policy gradient methods for Atari games [68], but has not yet been applied to summarisation to our knowledge. The second research question we address is whether pre-training reinforcement learning to avoid learning random sequences (similar to Zhang et al. [82]) also improves the summarisation quality.

In this chapter we describe our approach to answering both of our research questions. We address our first research question pertaining to PPO by implementing three necessary parts which we list below. The three parts below are also needed to answer our second research question, which includes a pre-training stage which we will describe in Chapter 5.

1. A **query-based multi-document extractive summarisation task** (Section 3.1)

2. A measure of **summarisation quality** (Section 3.3)

3. A **Proximal Policy Optimization (PPO)** reinforcement learning implementation (Section 3.4)

The remainder of this chapter describes our approach to prepare each of the three parts needed to address our research questions. Section 3.1 describes the BioASQ shared task, which is our extractive summarisation task. Section 3.2 describes our participation in the BioASQ shared task using our baseline REINFORCE approach. Section 3.3 describes the ROUGE-SU4 F1 metric which we use to evaluate summarisation quality. Section 3.4 describes the stable baselines implementation of PPO which we use.

3.1 BioASQ Dataset

The dataset we use has been provided to us by the organisers of the BioASQ competition [74]. The BioASQ competition is a shared task, which started in 2013 and organises annual challenges. The 2019 BioASQ challenge consists of 3 individual tasks. Task 7a requires participants to classify new PubMed documents. Task 7b-PhaseA requires participants to submit snippets or RDF triples from PubMed documents which relate to a query. Task 7b-PhaseB requires participants to submit answers (**ideal** or exact) to a query, which can be based on the snippets provided.

Our system generates **ideal answers** to the queries in the BioASQ Task 7b-PhaseB dataset. Some example data from the BioASQ dataset is shown in Table 3.1. The dataset contains queries, snippets from documents, ideal answers and exact answers. Our system generates answers similar to the ideal answers in this dataset by extracting sentences from the snippets provided. We choose extractive summarisation for this task because we observe that the gold (correct) human summaries tend to contain the same text used in the source snippets [46]. Also, the results of this BioASQ task are reported using the ROUGE metrics [33], which is a common metric for summarisation approaches. We therefore treat this BioASQ Task 7b-PhaseB as an extractive summarisation task.

The dataset that we use consists of 2,747 questions, and 36,006 snippets shared between the questions. We divide the data into a training set (2,289 questions) and a testing set (458 questions) in order to train and evaluate our systems. We also submit our baseline systems to the BioASQ competition, which consists of further test data where the human annotated answers are only available to the BioASQ organisers. The competition organisers evaluate the results of the systems we submit, and provide the ROUGE scores and human evaluations of our results.

question	Are long non coding RNAs spliced?
type	yesno
ideal answer	Long non coding RNAs appear to be spliced through the same pathway as the mRNAs
exact answer	yes
snippet 1	Our analyses indicate that lncRNAs are generated ...
snippet 2	For alternative exons and long noncoding RNAs, ...
snippet 3	bosome-mapping data to identify lncRNAs of Caenorhabditis ...
snippet 4	We introduce an approach to predict spliced lncRNAs ...
snippet 5	Owing to similar alternative splicing pattern to mRNAs ...
snippet 6	Our synthesis of recent studies suggests that neither size, ...

TABLE 3.1: Example Data from BioASQ Task 7b-PhaseB Training Dataset. Ideal answers contain sentences which are sometimes similar to the snippets, and exact answers give a more direct answer based on the question type.

3.2 REINFORCE BioASQ Submission

Our submission to the BioASQ task in 2019 [48] consists of 5 baseline systems. Our systems are First-N (MQ-1), Support Vector Classifier (MQ-2), Neural Regression (MQ-3), Neural Classification (MQ-4), and REINFORCE (MQ-5). I contributed to creating and training the Neural Regression model (For MQ-3, Batch 1-3), training the Neural Classification model (For MQ-4, Batch 1-3), and training the REINFORCE algorithm using word embeddings (For MQ-5, Batch 2 only). Further details of this submission are published in Mollá and Jones [48].

Table 3.2 shows the human evaluations for these systems in the BioASQ competition in 2019. **First-N (MQ-1)** is a strong baseline which simply takes the first N sentences to be used to answer the question. The value of N changes depending on the question type, which is shown in Table 3.3. **Neural Regression (MQ-3)** and **Neural Classification (MQ-4)** apply the deep learning approach using the supervised regression and classification techniques described in Mollá [47]. **REINFORCE (MQ-5)** is a simple implementation of the REINFORCE algorithm [78] which we use as a baseline in this thesis. We include our results from the BioASQ shared task because this thesis compares our REINFORCE (MQ-5) implementation with the PPO reinforcement learning approach. We also include ROUGE-SU4 F1 scores of the First 3 batches in Appendix A.

Table 3.2 shows that the REINFORCE (MQ-5) approach produces the best results in 3 batches, and is very close to the best results in the other 2 batches. This shows that reinforcement learning approaches can outperform supervised machine learning approaches, as observed in past research [55]. This also supports our decision to focus on reinforcement learning in this thesis.

Batch	**Batch1**	**Batch2**	**Batch3**	**Batch4**	**Batch5**
First-N (MQ-1)	4.195	**4.318**	4.213	4.260	4.405
Support Vector Classifier (MQ-2)	**4.223**	4.285	4.213	4.258	4.393
Neural Regression (MQ-3)	4.195	**4.318**	4.213	4.260	4.405
Neural Classification (MQ-4)	4.160	4.300	4.208	4.285	4.395
REINFORCE (MQ-5)	4.208	4.300	**4.223**	**4.375**	**4.428**

TABLE 3.2: This table shows the mean human evaluation score for each of our submissions to the BioASQ Competition. The results in bold are the highest average human evaluation for that batch. The runs that I contributed to were MQ-3 (Batch 1-3), MQ-4 (Batch 1-3), and MQ-5 (Batch 2 only).

Type	summary	factoid	yesno	list
First-N	6	2	2	3

Table 3.3: The number of sentences chosen by the First-N (MQ-1) system, based on Question Type.

Our submission to the BioASQ competition in 2019 [48] uses a slightly modified version of the REINFORCE algorithm described in Mollá [47]. The REINFORCE (MQ-5) implementation is based on the original REINFORCE algorithm [78], and trains a policy that predicts actions using the final reward directly in the loss function. This REINFORCE approach does not subtract any baseline (value function) from the policy gradient, and does not discount the future reward. Instead, the reward from the final action is back propagated and used in the gradient for all previous actions that were taken. We use our own version of REINFORCE, because REINFORCE is not implemented in the stable-baselines library that we use for PPO. Section 3.4 describes the unmodified PPO stable-baselines implementation, which discounts the future reward and subtracts a value function baseline in its loss function. We also report our results using a modified version of PPO (PPO-mod) which directly incorporates the future reward in the loss function, because this is what our REINFORCE baseline does (See Subsection 4.2.2).

The policy function that we use for REINFORCE is stochastic, and uses a Keras neural network [20] to predict the probability of choosing a sentence. The neural network design is shown in Figure 3.1. The inputs to the neural network are taken from the sentences in the current environment which are to be summarised by the system. The first 300 words from each sentence are mapped to an embedding representation for that word which comes from the word2vec embedding matrix. The mean of the word-level embeddings is taken as the representation of each sentence, and concatenated into a single input layer. We then use 2 hidden layers, each of size 200 and using the ReLU activation function. Two output nodes are then applied for the value function and policy function, each with different weights. The value function is only used for PPO, whereas the policy function is used for both PPO (using linear activation) and REINFORCE (using sigmoid activation). The final summary comes directly from the sentences chosen based on the policy function

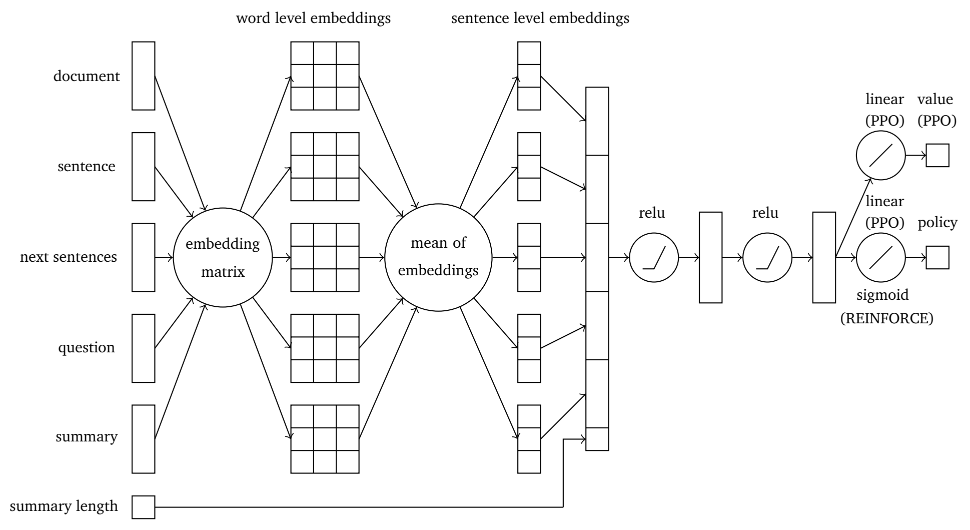


FIGURE 3.1: A diagram of our neural network for PPO and REINFORCE. The REINFORCE neural network only outputs a policy function, whereas the PPO implementation outputs both a policy function and a value function after the second hidden layer (where each output layer has different weights).

outputs — meaning that if the policy doesn't choose any sentences then the final summary will be empty.

The state observations fed into the neural network are made up of the 6 feature types listed below. Our environment returns these same observations for PPO, REINFORCE, and all other approaches we use in this thesis. These feature types are the same for the REINFORCE (MQ-5) algorithm we submitted to the BioASQ competition as well [48].

1. **Document** — word2vec embeddings for the whole document
2. **Sentence** — word2vec embeddings for the current sentence
3. **Next Sentences** — word2vec embeddings for the remaining sentences that are yet to be considered
4. **Question** — word2vec embeddings for the query text
5. **Summary** — word2vec embeddings for the summary generated so far
6. **Summary Length** — The length of the summary generated so far

3.3 ROUGE Metrics

We evaluate our summaries using the ROUGE algorithm [33], which compares the system outputs with the gold standard answers. We choose ROUGE to evaluate our system's performance because it is commonly used to evaluate other Summarisation methods [52]. It is also used by the organisers of the BioASQ shared task. However, human evaluations are a more accurate measure of the usefulness and human-readability of the summary that was generated, and are useful in evaluating summarisation methods. In future research we would like to perform human evaluations of our system as much as possible, however it was not feasible to obtain human evaluations for all of our systems within the timeline of this Masters Project.

We also performed an investigation of the ROUGE metrics [33] which are used in scoring the BioASQ summarisation task. Our results were published in Mollá and Jones [48]. I contributed to scraping the human evaluation results from the BioASQ Results page, and calculating the Pearson correlation only. Mollá and Jones [48] compares ROUGE Precision, Recall and F1 with the human annotated results, and shows that Precision has the highest correlation with human evaluations, followed by F1. Given that Precision may favour short summaries, we choose F1 to train our models, and report all our results using the Perl ROUGE-SU4 F1 metric. This thesis provides human annotations where possible, and uses the mean ROUGE-SU4 F1 score for all our other results.

3.4 PPO Implementation

This section introduces the Proximal Policy Optimization (PPO) algorithm library that we use, which comes from the OpenAI open source libraries.

Our PPO algorithm is based on the OpenAI implementation of PPO. OpenAI baselines [11] is a GitHub repository containing a number of reinforcement learning algorithms which are designed to work for OpenAI Gyms [3], and for Atari Games. The OpenAI Gym environments simulate real world environments in which a robot can be trained to move and perform tasks, and the Atari Game environments simulate Atari video games.

Applying the OpenAI version of PPO to another environment is difficult, and requires complex changes[1].

Stable-Baselines [24] is a GitHub repository forked from the OpenAI baselines repository. Stable-Baselines simplifies the OpenAI reinforcement learning code so that it is easier to create new environments. The reinforcement learning algorithms implemented by Stable-Baselines are listed in Table 3.4. We use the stable-baselines implementation of the Proximal Policy Optimization (PPO) algorithm, and apply it to a new environment that performs query-based multi-document extractive summarisation.

We create a new environment similar to an OpenAI Gym environment [3], but which interacts with our BioASQ dataset shown in Table 3.1. The environment operates using states, actions and rewards as described by the Markov decision process. Each **state** observation includes the word2vec embeddings of the feature types described in Section 3.2. The **actions** possible in the environment are '0' to add the current sentence to the summary, or '1' to not use the sentence in the summary. The **reward** is based directly on the Perl ROUGE-SU4 F1 evaluation which compares the machine generated summary to a gold (correct) human annotated summary.

We then use the stable-baselines PPO implementation to train an actor for our new environment. We use the default stable-baselines actor-critic MLP model for our policy and value functions. Figure 3.1 shows the design of our PPO and REINFORCE models. We note that the stable-baselines PPO model outputs both a policy function and value function using a linear activation layer, both with an output size of 2 for the 2 possible actions. This is because the PPO stable-baselines implementation requires the model to have a specific output layer (which can handle environments with more than 2 actions as well).

In future research, we would like to implement the other stable-baselines reinforcement learning approaches listed in Table 3.4. We did not have time to implement all of these during the course of this Masters Project, or to adapt our system to use the TRPO approach. We compare our results with the DQN and ACER approaches in this thesis, but we did not

[1]https://towardsdatascience.com/stable-baselines- a-fork-of-openai-baselines-reinforcement-learning-made-easy-df87c4b2fc82

parameter tune or optimize these approaches for our summarisation task either.

A2C	Actor-Critic (Improved A3C)
ACER	Actor-Critic
ACKTR	Actor-Critic
DDPG	Actor-Critic (Off-Policy)
DQN	Value-Based
GAIL	Actor-Critic (Improved TRPO)
HER	Apply to: DQN, DDPG or SAC
PPO	Actor-Critic
SAC	Actor-Critic (Off-Policy)
TRPO	Actor-Critic

TABLE 3.4: List of Reinforcement Learning algorithms implemented in the stable-baselines repository [24]. Proximal Policy Optimization (PPO) is in bold because we use the stable-baselines implementation of this algorithm in our experiments.

4 Proximal Policy Optimization

In Chapter 3, we described the approach we use to answer both of our research questions, including the PPO library we use. In this chapter we describe the experiments which we run in order to answer our first research question of whether PPO can be used to improve the summarisation quality of a query-based multi-document extractive summarisation task. We also describe the baseline systems which we use to compare our PPO approach performance with. Our results indicate that although PPO does not outperform our REINFORCE baseline in terms of ROUGE-SU4 F1 score, the PPO CLIP function penalises changes to the policy [68], resulting in a more stable learning curve.

The layout of this chapter is as follows. Section 4.1 describes the baseline systems which we use to compare our PPO results. Section 4.2 describes the PPO algorithm that we test, and the parameters we choose for it. Section 4.3 describes our results using several batch sizes for PPO.

4.1 Baselines

This section describes the baseline systems we use to compare with PPO. Subsection 4.1.1 describes our results using the First-N baseline, and other basic baselines. Subsection 4.1.2 describes our results using the REINFORCE algorithm from Mollá [47] as a baseline. Subsection 4.1.3 describes our results using the DQN and ACER approaches implemented in the stable-baselines repository [24].

4.1.1 First-N

This subsection describes the First-N baseline which we compare with PPO. First-N was included in our submission to the BioASQ task in 2019 [48], and was described in Section 3.2. In this subsection, we compare First-N with some other basic baseline approaches, and report our results on our test data set.

Table 4.1 lists our results using 4 basic baseline approaches. The baselines we compare are First-N, Last-N, Random and All. All of our results are reported using the same test data set, and the results reported here are comparable to our REINFORCE and PPO results. We briefly describe each baseline below.

First-N is a strong baseline which simply takes the first N sentences to be used to answer the question. The number of sentences it takes is dependent on the question type, as we showed in Section 3.2 Table 3.3 [48].

Last-N takes the same number of sentences as First-N, but takes the last sentences instead of the first ones. We observe that this baseline is very similar to randomly choosing, which suggests that the position of the sentences is important, and the first sentences are more likely to correspond to the human annotated summary than the last sentences (despite choosing the same number of sentences from a different position).

Random takes every sentence at random with a 50 percent probability. This baseline represents the score that our model should be able to achieve without any learning. Table 4.1 shows the mean result running Random 10 times.

All takes every sentence, and includes them all in the summary. This baseline also performs poorly, which shows that simply choosing all of the sentences does not produce a good ROUGE score.

Baseline (On Test Data)	**ROUGE-SU4 F1**
First-N	**0.247**
Last-N	0.196
Random	0.195
All	0.189

TABLE 4.1: Mean ROUGE-SU4 F1 scores for our basic baseline systems, using consistent test data for all experiments. The Random result is the mean of running 10 separate runs with a standard deviation of 0.007.

4.1.2 REINFORCE

This subsection describes the REINFORCE baseline which we implement based on Mollá [47], and was submitted to a shared task [48]. We include REINFORCE as a baseline so that we can compare PPO with a purely policy-based reinforcement learning approach. We use the same neural network architecture as described in Section 3.2, with only a policy function output. We run this REINFORCE baseline 3 times over 500,000 timesteps, and record the **maximum training data result** for each run (which represents the maximum point on the REINFORCE learning curve, taken from the average ROUGE-SU4 F1 score received for each question in the training dataset). Table 4.2 reports the average of the 3 **maximum training data results** which were received on the 3 different runs. We observe that REINFORCE outperforms FIRST-N, and is a strong baseline. We also include the baseline learning curve graph of our REINFORCE approach in Figure 4.1. We observe that the REINFORCE learning curve goes up and down on the test data, and that the training data results are generally low because of the model exploring alternative actions.

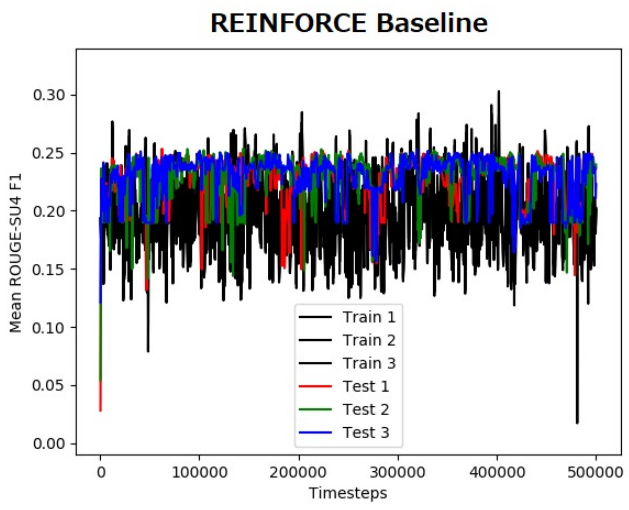


FIGURE 4.1: Graph of the REINFORCE learning curve for 3 runs over 500,000 timesteps, using ROUGE-SU4 F1 score as the reward. The training results (black) tend to score either the same or worse than the test data (red, green, and blue) because of REINFORCE exploring different actions stochastically.

4.1.3 DQN and ACER

We also include results from the stable-baseline [24] implementations of the Deep Q Network (DQN) and Actor-Critic with Experience Replay (ACER) approaches. We include DQN as a baseline because it is a purely value-based reinforcement learning approach which we can compare our results with. Actor-Critic approaches such as PPO generally perform better than purely value-based approaches, and this is reflected in our results in Table 4.2. We also include the results using the ACER stable-baselines approach, which performs similarly to PPO in Atari Games but is more complex [68]. We could only make minor adjustments to the parameters for these algorithms, and can continue tuning the parameters of these algorithms in future research. For ACER we use a horizon of 20 (the default horizon) and 1000 (the best horizon for PPO) and observe that a horizon of 20 works better for ACER. Subsection 4.2.1 describes the horizon (n-steps) parameter in more detail for the PPO approach. Our results for these baselines are included in Table 4.2. The learning curve for both approaches is included in Appendix B.

Model	**Horizon**	**ROUGE-SU4 F1**	**S.D.**
FIRST-N	-	0.247	-
REINFORCE	-	**0.253**	0.001
DQN	-	0.214	0.003
ACER	20	0.247	0.003
ACER	1000	0.211	0.001
PPO-mod	1000	0.252	0.002
PPO	100	0.250	0.002
PPO	1000	0.251	0.001
PPO	10000	0.248	0.001

TABLE 4.2: List of PPO experiments using different horizon values, and a modified PPO version (PPO-mod), compared to the baseline reinforcement learning approaches. The scores shown are the average of the highest ROUGE-SU4 F1 scores across 3 runs for each approach. S.D. is the standard deviation of the 3 runs.

4.2 PPO

This section describes our experiments using the PPO stable-baselines implementation. Proximal Policy Optimization (PPO) was published in 2017 [68], and has not been applied to the task of summarisation previously to our knowledge. We apply PPO to query-based multi-document summarisation, and compare our results to the baseline algorithms described in Section 4.1.

We use the default parameters for PPO wherever possible, and make only minor changes to the PPO algorithm. The changes we make to the PPO algorithm are described in the subsections below. Subsection 4.2.1 describes our parameter tuning, using the horizon parameter of PPO. Subsection 4.2.2 describes our modified version of PPO (PPO-mod), which uses a different policy gradient. Table 4.2 lists the results of our experiments.

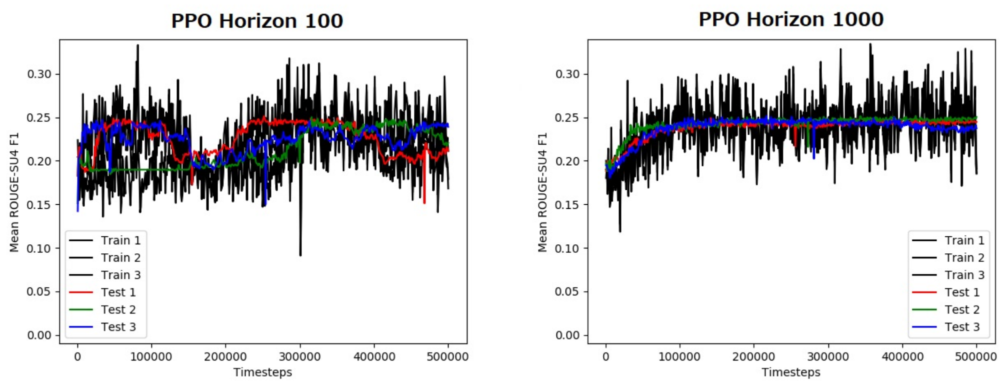


FIGURE 4.2: Comparison of the PPO learning curve with Horizon at 100 and Horizon at 1000 (each 3 runs using ROUGE-SU4 F1 as reward). The Horizon 1000 model is more consistent and does not unlearn good models.

4.2.1 Parameter Tuning

This subsection describes the **horizon** parameter that we tune to improve the performance of our PPO algorithm. The horizon (n-steps) parameter controls the number of steps taken before the policy gradient is updated. The reason we tune the horizon parameter is because parameters which relate to step size (like horizon) are known to impact the performance of reinforcement learning approaches, and the horizon parameter had the biggest impact out of the parameters we tested. Increasing the horizon reduces the impact of a single good summary, and incorporates a range of summaries in different environments into a single update to the neural network. The default horizon for Atari games is 128 (As stated in Schulman et al. [68], and implemented in stable-baselines), but we experiment with 100, 1000, and 10000 horizon sizes.

Figure 4.2 provides a comparison between the PPO learning curve at 100 and 1000 horizon sizes. The results of 3 runs over 500,000 timesteps are overlayed in different colours for both graphs. We observe that the learning curve for a horizon size of 1000 is more stable, and changes to the policy are penalised as intended [68]. We also report the best results from each learning curve averaged across the 3 runs in Table 4.2. We observe that there is very little difference between the best results of each horizon size. The learning curve for 10000 horizon size is included in Appendix B.

4.2.2 Policy-Based PPO Modification

This subsection describes our modification to the PPO policy gradient in **PPO-mod**. We implement a modified version of PPO for two reasons. The first reason is that our REINFORCE implementation uses the reward directly in the loss function (instead of a discounted reward) for its policy gradient, so we modify our PPO implementation to do the same thing so that we can better compare PPO with our REINFORCE baseline. The second reason we modify our PPO implementation is to better understand how PPO works, and to find out whether modifying the algorithm will improve our results. The remainder of this subsection describes the changes we made to the PPO algorithm, and the results that we received with our modified version.

The PPO paper [68] describes the advantage function used in the original PPO algorithm using Equation 4.1.

$$\begin{aligned} &A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \\ &\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{4.1}$$

At each timestep t the reward r_t is offset by subtracting a baseline which utilizes the state value function $V(s)$. The resulting offset reward δ_t is then discounted at each timestep using the discount factors gamma (γ) and lambda (λ). This advantage function (A) is based on Advantage Actor-Critic approaches [45], which incorporate the value function $V(s)$ as the critic.

We create a modified version of PPO (PPO-mod) using the advantage function in Equation 4.2, which is the same as the policy gradient for our REINFORCE algorithm described in Section 3.2.

$$A_t = r_t + r_{t+1} + \dots + \dots + r_{T-1} \tag{4.2}$$

Simply, PPO-mod removes the value function baseline which offset the reward, and removes the discount factors applied to future rewards. The resulting algorithm is a policy-based approach (as it doesn't use a value function), similar to our REINFORCE implementation, which applies clipping to the probability ratios based on the PPO CLIP algorithm. We can remove the discount factors from our rewards because the reward

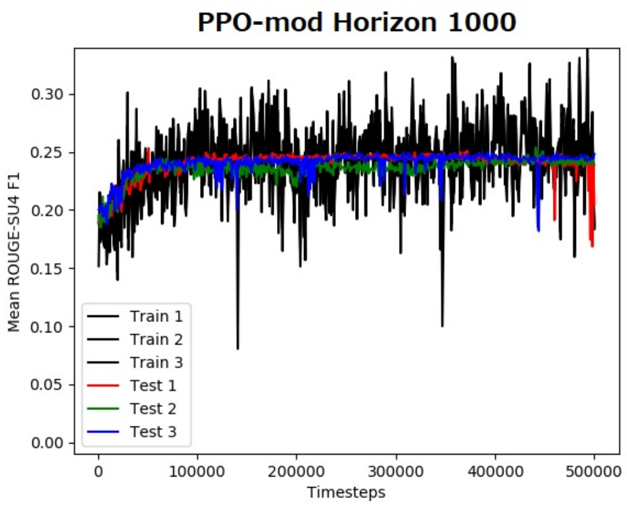


FIGURE 4.3: The learning curve for PPO-mod run 3 times over 500,000 timesteps. Reward uses the ROUGE-SU4 F1 metric.

of all actions except for the final action is 0. The results of PPO-mod represent the impact that the PPO CLIP function alone has when compared to our baseline REINFORCE approach. We report the results of PPO-mod in Table 4.2, and observe that this approach performs similarly to the unmodified PPO implementation, and does not outperform the REINFORCE baseline approach. We also report the learning curve of PPO-mod in Figure 4.3.

4.3 Results and Discussion

Our results in this chapter showed that PPO does not improve the ROUGE score for our query-based multi-document extractive summarisation task. There is very little difference between our results for PPO and REINFORCE in Table 4.2, and we do not receive any human evaluations which we can report. Our First-N baseline is outperformed by the PPO and REINFORCE approaches, by a ROUGE difference of 0.004 and 0.006 respectively.

In Figure 4.2 we observe that the PPO learning curve is consistent, and our training results (black) are evenly spread on either side of our testing results (red, green and blue). Our REINFORCE Baseline (Figure 4.1) has an uneven learning curve and updates the policy gradient every time the environment is done. However, the PPO CLIP function penalises changes to the policy, resulting in a more consistent learning curve with little variation in the test data results [68]. We also observe that using a larger horizon size ensures that a good policy is not unlearned, because more steps are considered before each update to the policy gradient. As a result, the PPO learning curve appears to be more consistent than our REINFORCE baseline implementation.

We also observed that our modification to PPO (PPO-mod) appears slightly more competitive with REINFORCE than the unmodified PPO algorithm. However the difference between the PPO and PPO-mod ROUGE-SU4 F1 scores is 0.001, which is not a significant difference for the ROUGE metric. The learning curve for PPO-mod is also very similar to the original PPO implementation, which confirms that the PPO clipping approach still works when it is applied to a vanilla policy gradient approach [68]. This also suggests that the value function we are using for PPO does not improve our model, as removing it does not make much difference.

Overall, the PPO algorithm does not provide any improvement to the baseline REINFORCE algorithm results. However, we observe that the learning curve for the PPO algorithm is more steady, and it does not unlearn good models when using a large horizon.

5 Pre Training

In Chapter 4 we described the experiments we ran to answer our first research question pertaining to PPO. In this chapter we describe the experiments we run in order to answer our second research question of whether pre-training reinforcement learning to avoid learning random sequences improves the summarisation quality. The reason we load a pre-trained model is to start with a strong baseline and avoid learning random sequences when training our model, similar to Zhang et al. [82]. We then test whether this pre-training step results in an improvement to the summarisation quality. The rest of this chapter describes the deep learning approach which we use for our pre-trained model, and the reinforcement learning systems which we apply the pre-trained model to.

The remainder of this chapter is as follows. Section 5.1 describes how we pre-train our neural network. Section 5.2 describes how we load the pre-trained model into PPO and REINFORCE. Section 5.3 discusses our results, and the probability of choosing the first sentences after pre-training.

5.1 Pre Training

This section introduces our approach which pre-trains the neural network using supervised deep learning approaches on our BioASQ dataset. Pre-training can be applied to word vectors, language models, or to the neural network. Pre-trained word vectors such as GLOVE are commonly used for summarisation tasks [56, 80]. Pre-trained language models such as BERT and ELMo have also been used in recent works [10, 57]. Our approach involves pre-training the weights of our **neural network** to avoid random label sequences while training, similar to Zhang et al. [82].

We perform pre-training on the same dataset, using the same split of training and testing data each time. We do not perform any transfer learning on our neural network. First, we label the top N sentences using the approach in sub-section 5.1.1. Next, we train the neural network to predict the labels using the deep learning classification approach with binary cross entropy loss and the Adam Optimizer [28]. We adjust the weights for the classes using the approach in sub-section 5.1.2. Finally, we report the ROUGE score for the sentences predicted by the model with a threshold of 0.5 (ie. the sentences with a probability over 50 percent are chosen to be in the summary). Table 5.1 shows these ROUGE results after pre-training alone. We do not report the classification precision or accuracy for the classifiers.

The highest ROUGE value we get for our classification model is 0.247. We use this best pre-trained model for all of our experiments that load a pre-trained model in Section 5.2. We also include the results of this model as our pre-training baseline in Table 5.2. We note that this classification model uses the same architecture as our REINFORCE neural network from Section 3.2. This is so that the pre-trained classification model can be loaded into our REINFORCE model to continue training. As a result, the classifier architecture we use is different to the classifier reported in Mollá and Jones [48] (MQ-4, using LSTMs). Because the classifier reported in Mollá and Jones [48] gives a better ROUGE score in the final summaries, we consider incorporating this classifier's neural network into our reinforcement learning approach in future work.

Weight	**N = 2**	**N = 3**	**N = 4**	**N = 5**	**N = 6**
1	0.118	0.128	0.152	0.183	0.181
5	0.229	0.237	0.244	0.242	0.234
10	0.230	0.241	0.245	0.241	0.234
25	0.238	0.242	0.246	0.241	0.233
50	0.237	0.244	**0.247**	0.241	0.233
100	0.236	0.244	0.247	0.238	0.228
500	0.236	0.244	0.245	0.220	0.189

TABLE 5.1: Results from pre-training our neural network to classify sentences. Results are shown using the average ROUGE evaluation of the summary generated for each query in the test dataset when choosing sentences using the classification model. The result in bold is the result that we choose to pre-train our reinforcement learning models with, where the weight is 50, and the best Top N labels is 4.

5.1.1 Top N Labels

Before Pre-Training the neural network model, we label the gold (correct) sentences which the classifier will be trained to predict. To do this, we choose the first N sentences which have the highest ROUGE score compared to the human annotated summary, and class those sentences to be selected for the machine summary. Once we have labels for the gold sentences, we can then train the neural network to identify those gold sentences without prior knowledge of the human annotated summary. We try a range of values for N when choosing our top N labels, and report our results in Table 5.1. We identify 4 as the best value for Top N based on Table 5.1.

5.1.2 Class Weights

This subsection describes the class weights that we apply when training our deep learning classifier using the Adam Optimizer [28]. We train our classification model using the binary cross entropy loss function as provided by keras. We also apply class weights, which are used for weighting the loss function during training time only. Please see the keras documentation[1] for more information about the class weight parameter for the model fit function. We apply a weight of 1 to the non-summary sentence class, and vary the weight of the summary sentence class according to the weights in Table 5.1. We assign weights because our summary and non-summary sentences are not evenly distributed, and because our classification model performs better when it is more likely to include sentences in the summary.

Table 5.1 shows our results using different weights for the training of the summary sentences. Intuitively, when choosing more sentences for the Top N, there will be more instances of that class, and the optimal weight will be smaller. Despite this, the optimal weight values are much larger than we expected. For example, when N = 4 there are 2.75 non-summary sentence instances for every 1 summary sentence instance, but we observe that the best weight at N = 4 is closer to 50 or 100 (in favour of predicting summary sentences).

5.2 Loading Pre Trained Model

In Section 5.1 we explained how we choose the best pre-trained model for our neural network. This section describes how we use our pre-trained model to improve our reinforcement learning results. As previously mentioned, we use a pre-trained model in order to avoid learning random sequences similar to Zhang et al. [82], and so that we can determine the impact that pre-training has on our summarisation quality. We train the REINFORCE and PPO algorithms starting with our best pre-trained model (N = 4, Weight = 50), and report the results in Table 5.2. The rest of this section describes our individual experiments for REINFORCE and PPO.

[1] `https://keras.io/models/sequential`

Model	ROUGE-SU4 F1	S.D.
Pre-Training	0.247	-
REINFORCE	**0.253**	0.001
REINFORCE + Pre-Training	0.251	0.001
PPO	0.251	0.001
PPO + Pre-Training	0.250	0.000

TABLE 5.2: Results of Pre-Training reinforcement learning approaches compared to the original runs. The scores shown are the average of the highest ROUGE-SU4 F1 scores across 3 runs for each reinforcement learning approach. S.D. is the standard deviation of the 3 runs. The baseline REINFORCE result is shown in bold because it has the highest score.

5.2.1 REINFORCE + Pre-Training

This subsection describes our experiments using the **REINFORCE + Pre-Training** system. The pre-trained network has the same design as our REINFORCE neural network model, so we load the pre-trained model directly and perform reinforcement learning with it. Figure 5.1 shows the learning curve when running the REINFORCE algorithm for 200,000 timesteps after pre-training with N = 4 and Weight = 50. Our results show that the pre-trained model does not significantly improve on the baseline REINFORCE model. We also observe a similar learning curve to the normal REINFORCE model despite starting with a pre-trained model. We discuss these results further in Section 5.3.

5.2.2 PPO + Pre-Training

This subsection describes our experiments using the **PPO + Pre-Training** system. Our PPO neural network is different to the pre-trained neural network as it contains linear policy and value function outputs, both of size 2. Section 3.2 Figure 3.1 shows the differences between the REINFORCE output layer, and the PPO output layers. Because the output layers for the policy functions are not compatible, we insert the sigmoid output layer of size 1 before the linear output layer of size 2, so that we can return the probability for both actions as required by the stable-baselines PPO implementation. We then initialize

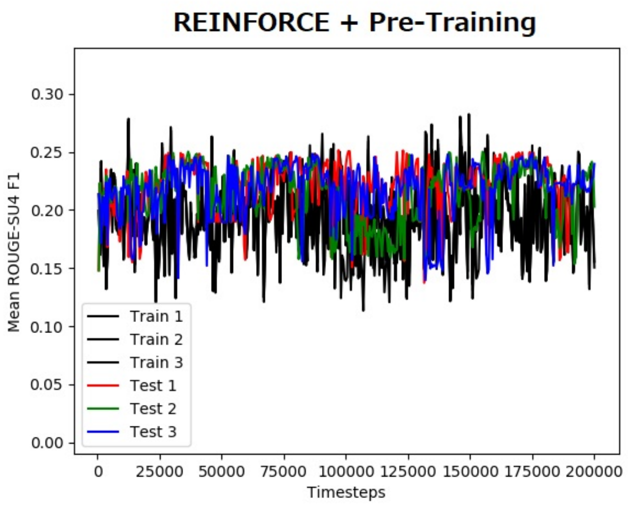


FIGURE 5.1: The learning curve for REINFORCE run on a pre-trained model with N = 4 and Weight = 50 for 3 runs over 200,000 timesteps.

the weights of the output layer to -6 and 6 (for actions 0 and 1 respectively), so that the linear output layer returns an action based on the sigmoid output layer. The resulting model directly returns the predictions from the pre-trained model to start with, and does not appear to explore different models.

Figure 5.2 shows our results running the PPO algorithm after pre-training. We again observe that our PPO implementation train results (black) are distributed evenly on both sides of the test results (red, green and blue), which is not the case for REINFORCE. We believe that CLIP function for PPO adjusts the probability ratios when choosing each sentence, as it is intended to do [68], which gives it some variance in the sentences that are chosen. We also observe that the learning curve starts much higher than our previous experiments, but does not appear to increase above its initial results. Section 5.3 discusses our results in more detail.

5.3 Results and Discussion

Table 5.2 shows that there is no significant improvement in ROUGE score when Pre-Training our PPO or REINFORCE models. The rest of this section analyses the results we

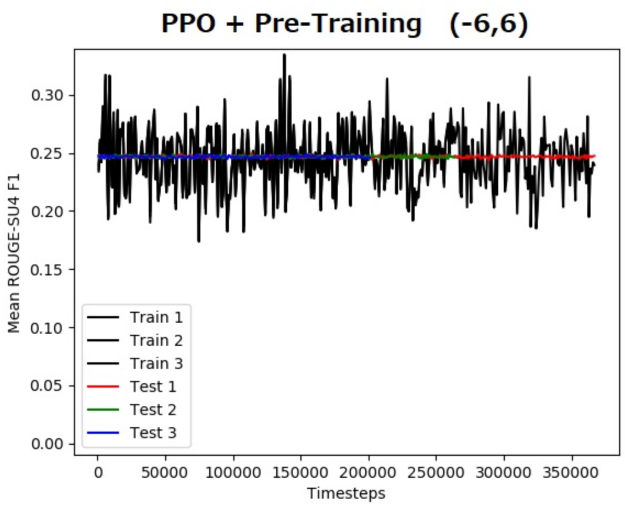


FIGURE 5.2: The learning curve for the pre-trained PPO model for 3 runs. The training data score (black) is distributed on either side of the testing data (red, green and blue). For this experiment the pre-trained sigmoid layer is fed into a linear output layer node with weights -6 and 6.

get during pre-training.

Pre-Training: We observe that our best pre-training model (before applying reinforcement learning) gets a ROUGE score of 0.247, which is the same as out First-N baseline from Subsection 4.1.1. On further investigation, we observe that our pre-trained model is choosing the first 4 sentences with a high probability, and not choosing the remaining sentences. Figure 5.3 shows that the probability of choosing each sentence in our pre-trained model is extremely biased, but becomes closer to 0.5 after running REINFORCE on the same model. These strong probabilities suggest that our pre-trained model acts very similarly to our First-N baseline, in that it is more likely to choose the first 4 sentences.

REINFORCE: In contrast, the probability of choosing sentences after running REINFORCE is closer to 0.5, which means that there will be a 50 percent chance of selecting those sentences when training (Figure 5.3). There are 2 changes that we could make to our post-training REINFORCE model to prevent the probability distribution from becoming too random. The first change we could make is to reduce the impact of a single policy gradient update on the model. PPO already does this by performing only one gradient update for each horizon, but we could modify REINFORCE to make smaller changes

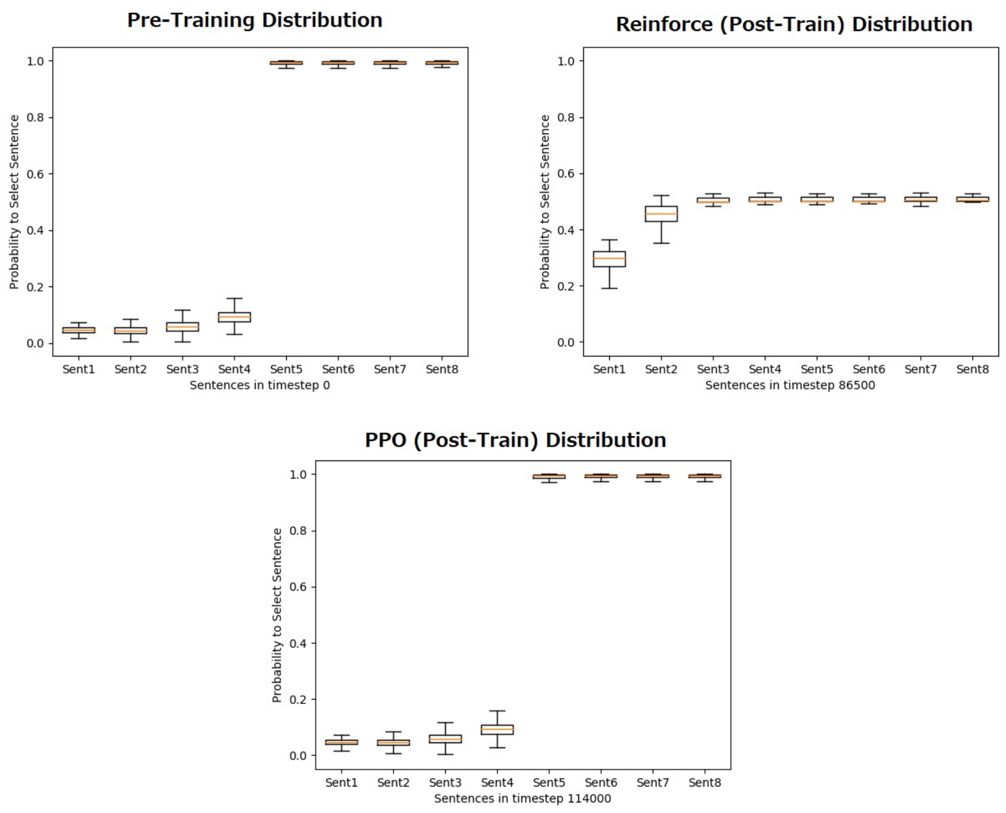


FIGURE 5.3: These graphs compare the average probability of choosing each sentence across all 458 test questions, for Pre-Training, REINFORCE + Pre-Training and PPO + Pre-Training. The pre-training probabilities of the first 4 sentences are strongly skewed towards 0, but become closer to 0.5 after applying reinforcement learning.

as well by reducing the learning rate or adding a horizon to the model. We report the learning curve after changing the Pre-Trained REINFORCE Horizon and learning rate (LR) in Appendix B, and there is no significant improvement in the results. The other change we could make is to reduce the amount of noise applied to the probability ratios. Our REINFORCE implementation uses Equation 5.1 to vary the noise N over multiple timesteps, where t is the current timestep and T is the total number of timesteps.

$$N = 0.2\frac{T}{(T+t)} \tag{5.1}$$

Currently we reduce the noise from 0.2 to 0.1 over time. Making the noise smaller may help our model to exploit good actions that it finds. Finding a good balance between exploring (choosing randomly) and exploiting (following the policy) is a central issue in reinforcement learning [26], and is a possible topic for future research. Table 5.2 reports our REINFORCE + Pre-Training results using our unmodified REINFORCE implementation which includes noise. Our results indicate that REINFORCE + Pre-Training does not improve on our original REINFORCE experiment without pre-training.

PPO: Our PPO model only updates the policy gradient every 1,000 steps (based on horizon size), and has a more consistent learning curve (Figure 5.2) than our REINFORCE implementation. As in Chapter 4, we observe that the training data results (black) are evenly distributed on either side of the testing data (red, green and blue). PPO does not add any noise to the probability ratios, but instead applies the CLIP function to the probability ratios, which we do not adjust. For this reason, we observe that PPO still chooses sentences based on the probability distribution.

We observe that the PPO learning curve does not improve much when using a pre-trained model (Figure 5.2). The PPO CLIP function prevents any large changes to the pre-trained model, which means that running PPO on a strongly pre-trained model does not make any difference compared to just performing the pre-training stage. We also observe that the probability of choosing each sentence after running PPO is similar to the sentence probabilities in the original pre-training model (Figure 5.3). This suggests that applying PPO to a strongly pre-trained model does not allow for any improvement, as PPO will rarely choose a different action for a sentence which is already biased towards taking a certain action. However, Zhang and Wang [81] reports an improvement when pre-training PPO for a mobile robot, and in future research we could investigate ways of making pre-training effective for our PPO summarisation approach (Such as using smaller weights for the output node, which gave a lower ROUGE score as shown in Appendix B).

In this chapter we pre-trained the weights of our neural network in order to avoid random label sequences while training, similar to Zhang et al. [82]. We observe that REINFORCE does not perform any better, and the PPO learning curve starts from a good position but does not learn any better policies. We do not observe any significant improvement in our results from pre-training the neural network.

6 Results and Discussion

This section discusses the results of our experiments reported in this thesis, which were based on two research questions. Our first research question is whether Proximal Policy Optimization (PPO) can improve the summarisation quality of our query-based multi-document extractive summarisation task. Our second research question is whether pre-training reinforcement learning to avoid learning random sequences also improves the summarisation quality.

Table 6.1 lists the results from each of our reinforcement learning experiments. We observe that PPO does not significantly improve the ROUGE-SU4 F1 metric compared to our REINFORCE baseline. However, we do observe that the PPO learning curve is more consistent than REINFORCE when we use a high horizon value (See Section 4.2), as PPO penalises changes to the policy [68]. We do not receive human evaluations of our summarisation quality. Overall, we report no significant improvement to the summarisation quality for our first research question.

We also observe that pre-training does not improve our results for PPO or for REINFORCE. PPO struggles to learn any better policies because it penalises changes to the policy. In contrast, REINFORCE diverges from the pre-trained policy quickly, and performs the same as it does without pre-training. Overall, we observe no benefit in using pre-training for our second research question, and could investigate ways of making pre-training more effective in future research.

All of the approaches listed in Table 6.1 use the word2vec word embeddings, except for where TFIDF is specified. In Mollá and Jones [48], I contributed to adding word embeddings to REINFORCE, and observed that there is no major improvement when using word embeddings instead of TFIDF. In Table 6.1, we show that the difference between the maximum ROUGE-SU4 F1 score for REINFORCE and REINFORCE + TFIDF is 0.005. Our word embeddings approach does not appear to majorly improve on the REINFORCE results using TFIDF reported in Mollá and Jones [48].

Type	**Model**	**ROUGE-SU4 F1**	**S.D.**
Baseline	FIRST-N	0.247	-
Baseline	Pre-Training	0.247	-
Policy-Based	REINFORCE	**0.253**	0.001
Policy-Based	REINFORCE + Pre-Training	0.251	0.001
Policy-Based	REINFORCE + TFIDF	0.248	0.001
Policy-Based	PPO-mod	0.252	0.002
Actor-Critic	PPO	0.251	0.001
Actor-Critic	PPO + Pre-Training	0.250	0.000
Actor-Critic	PPO + TFIDF	0.247	0.006
Actor-Critic	ACER	0.247	0.003
Value-Based	DQN	0.214	0.003

TABLE 6.1: Comparison of all Reinforcement Learning experiments run on the BioASQ text summarisation dataset. The PPO Actor-Critic approach does not outperform our original REINFORCE algorithm.

7 Conclusion

This thesis investigated two research questions. First, we investigated whether PPO can improve the summarisation quality of query-based multi-document extractive summarisation tasks. Second, we investigated whether pre-training reinforcement learning to avoid random sequences also improves the summarisation quality.

We observe that there is no significant difference between the summarisation quality of the PPO and REINFORCE approaches, based on the maximum ROUGE-SU4 F1 score for each. We also observe that there is no significant improvement in results when pre-training our reinforcement learning approaches. We only report our results using the BioASQ dataset, and could use additional datasets and human evaluations in future research.

We performed several experiments using PPO with different horizon sizes, and using a modified version which we called PPO-mod. Our experiments confirm that PPO penalises changes to the policy [68], but show that it reaches the same global maximum as our

REINFORCE implementation. We did not observe any benefit from using PPO-mod, but report that it performs similarly to the original PPO algorithm.

We also reported the distribution of sentence probabilities after pre-training, and after running REINFORCE and PPO with the pre-trained model. We observed that REINFORCE quickly diverges from the pre-trained model and chooses most sentences with a probability close to 0.5, and that PPO chooses sentences with the same probabilities as in the pre-trained model because it penalises changes to the model and struggles to learn an alternative model. In future research, we could adjust these models further so that the pre-trained model better assists our reinforcement learning approaches (such as using a different noise or learning rate).

In future research, we would like to apply more Actor-Critic approaches, as there are still several approaches which have not yet been applied to summarisation in Section 2.3 Table 2.2. In addition, we could add an abstractive stage to the summarisation process, which has been explored in past research [51, 82]. We would also like to investigate using LSTMs in our neural network in future research, as our classifier in Section 5.1 was outperformed by a past classification approach which used LSTMs [48]. Finally, future research could investigate different features to use as inputs to the policy function, as our current features (and word embeddings) may not be the best representation of the sentences we extract.

A

Appendix A: BioASQ ROUGE F1 Results

This appendix includes our ROUGE-SU4 F1 scores in the 2019 BioASQ competition. Only the F1 scores for the first 3 batches were provided by the BioASQ organisers.

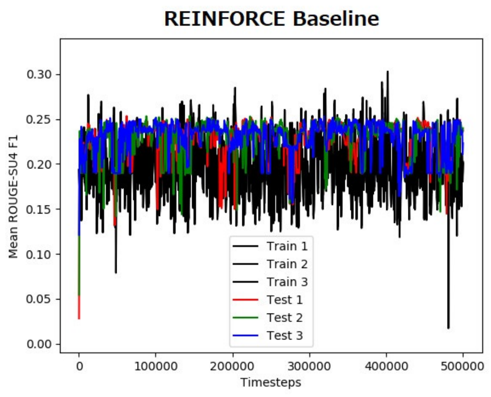
Batch	**Batch 1**	**Batch 2**	**Batch 3**
MQ-1	0.312	0.336	0.248
MQ-2	0.344	0.336	0.262
MQ-3	0.312	0.336	0.248
MQ-4	0.343	0.346	0.276
MQ-5	0.329	0.326	0.264

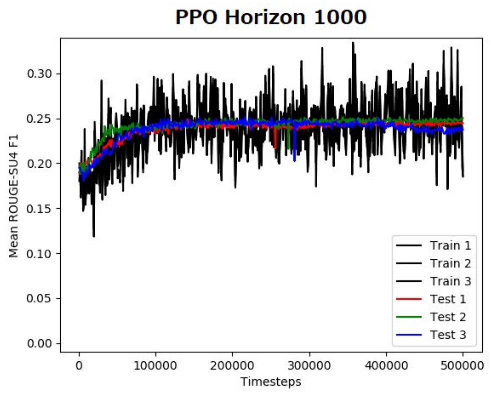
TABLE A.1: ROUGE-SU4 F1 scores for the first 3 batches of the 2019 BioASQ competition. Only the F1 scores for the first 3 batches were sent to us by the BioASQ organisers. Human evaluations are reported in Section 3.2. Mollá and Jones [48] reports the ROUGE Recall scores we received.

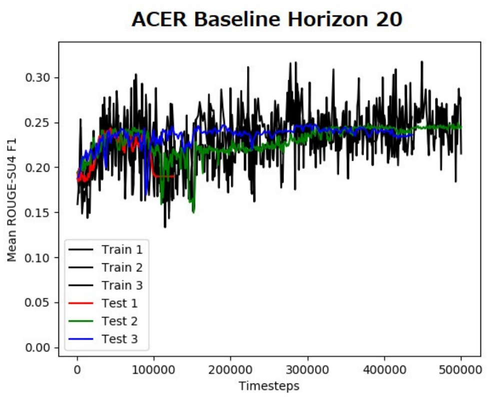
B

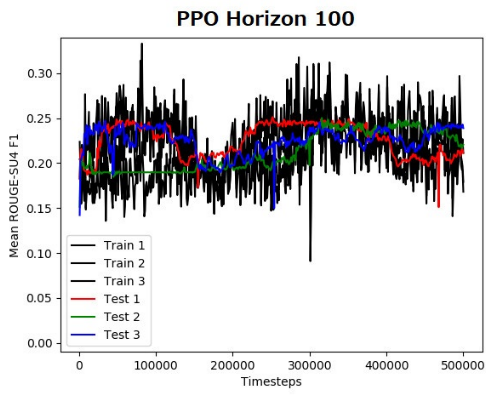
Appendix B: Learning Curves

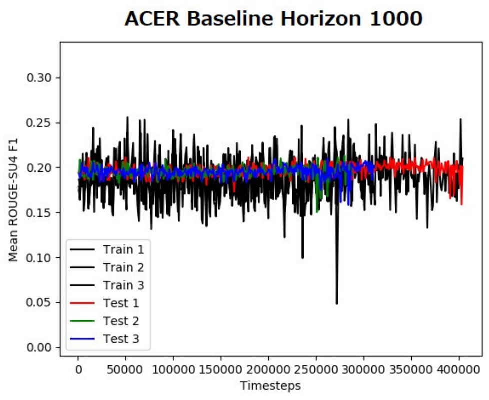
This appendix includes the Learning Curve graphs of 16 experiments reported in this thesis. The Learning Curve Graphs start on the next page, and continue for 2 pages.

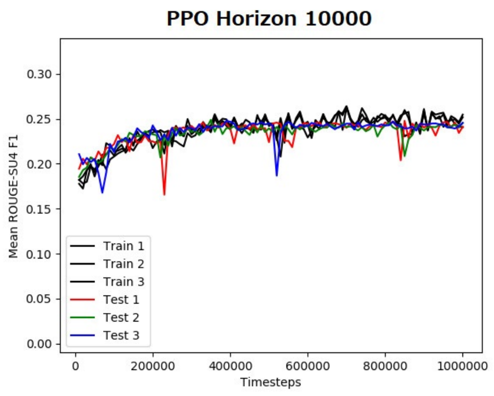

REINFORCE Baseline
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3

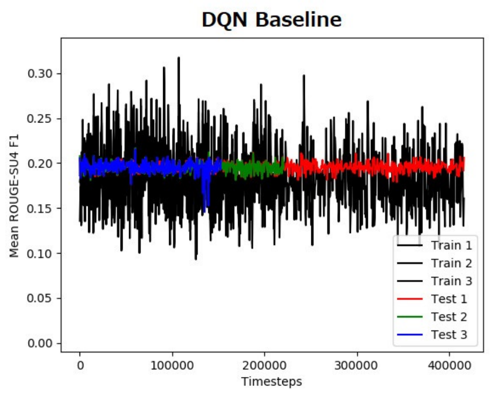

PPO Horizon 1000
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3

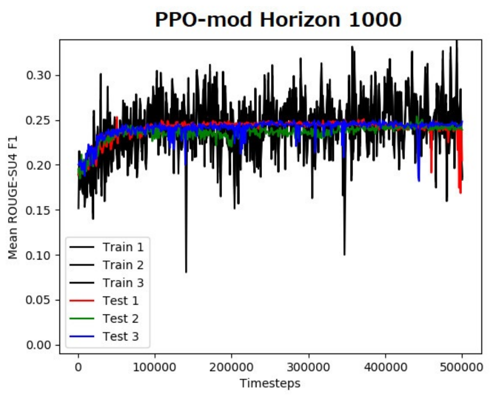

ACER Baseline Horizon 20
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3


PPO Horizon 100
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3


ACER Baseline Horizon 1000
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3


PPO Horizon 10000
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3


DQN Baseline
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3


PPO-mod Horizon 1000
Mean ROUGE-SU4 F1
Timesteps
Train 1
Train 2
Train 3
Test 1
Test 2
Test 3

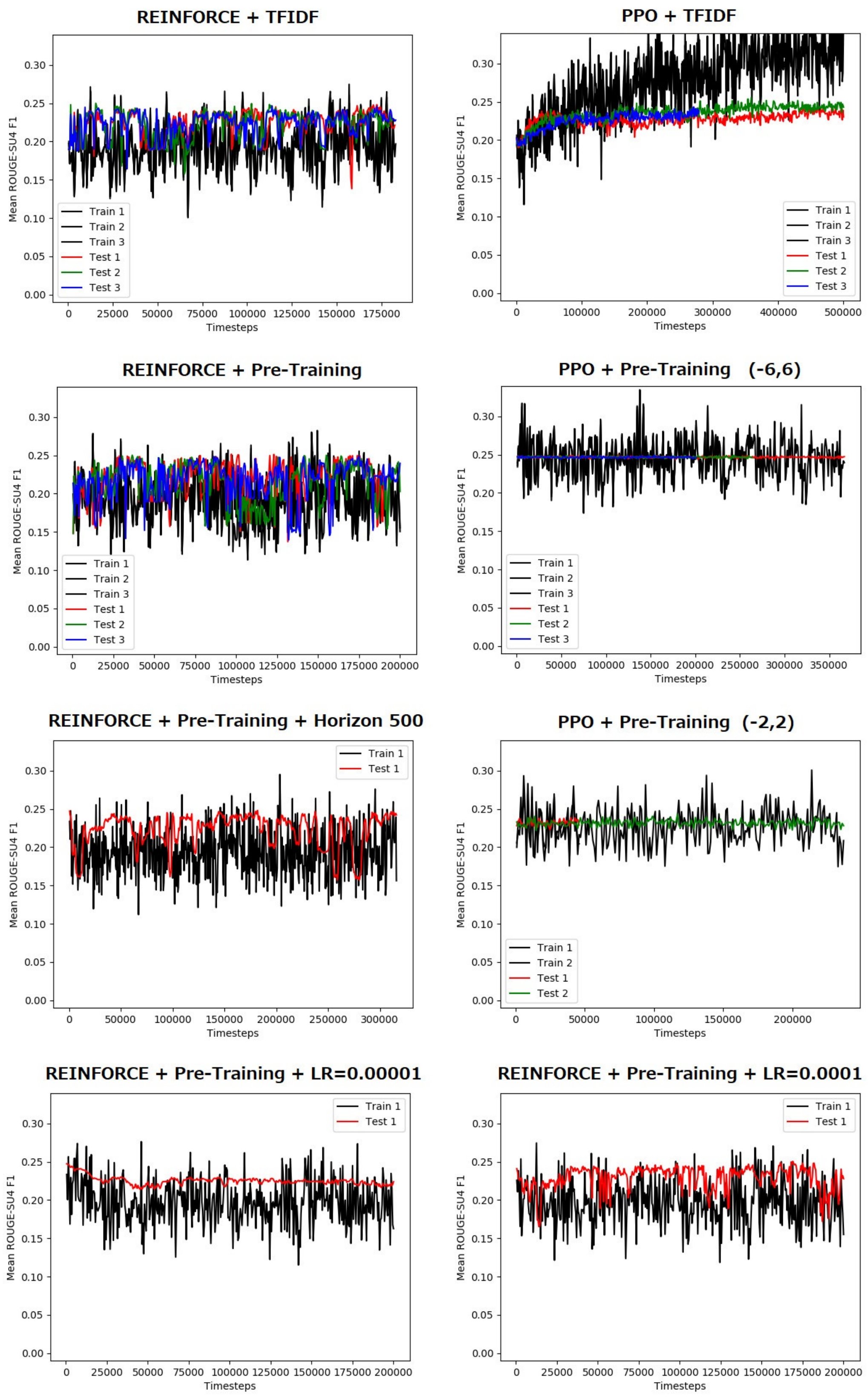


FIGURE B.1: Learning curves for the reinforcement learning algorithms using different settings. Runs on the left side use baseline systems (eg REINFORCE), and runs on the right side use PPO (except for the final REINFORCE one). TFIDF and Pre-Training graphs are on the second page.

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*, 2017.

[2] Gabriel Barth-Maron, Matthew W Hoffman, David Budden, Will Dabney, Dan Horgan, Alistair Muldal, Nicolas Heess, and Timothy Lillicrap. Distributed distributional deterministic policy gradients. *arXiv preprint arXiv:1804.08617*, 2018.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Ziqiang Cao, Chengyao Chen, Wenjie Li, Sujian Li, Furu Wei, and Ming Zhou. Tgsum: Build tweet guided multi-document summarization dataset. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[5] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[6] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. Back to basics: Benchmarking canonical evolution strategies for playing atari. *arXiv preprint arXiv:1802.08842*, 2018.

[7] Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. A discourse-aware attention model for abstractive summarization of long documents. *arXiv preprint arXiv:1804.05685*, 2018.

[8] Alex de Voogt and Joachim Quack. *The Idea of Writing: writing across borders*. 01 2012. ISBN 978-9004215450. doi: 10.1163/9789004217003.

[9] Franck Dernoncourt, Mohammad Ghassemi, and Walter Chang. A repository of corpora for summarization. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, 2018.

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

[11] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. *GitHub, GitHub repository*, 2017.

[12] Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. A hybrid approach to multi-document summarization of opinions in reviews. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 54–63, 2014.

[13] Yue Dong. A survey on neural network-based summarization methods. *arXiv preprint arXiv:1804.04589*, 2018.

[14] Harold P Edmundson. New methods in automatic extracting. *Journal of the ACM (JACM)*, 16(2):264–285, 1969.

[15] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[16] Takahiro Fukusima, Manabu Okumura, and Hidetsugu Nanba. Text summarization challenge: Text summarization evaluation at ntcir workshop2. In *NTCIR*. Citeseer, 2001.

[17] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348, 2010.

[18] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34, 2003.

[19] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[20] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[22] Stefan Henß, Margot Mieskes, and Iryna Gurevych. A reinforcement learning approach for adaptive single-and multi-document summarization. In *GSCL*, pages 3–12, 2015.

[23] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pages 1693–1701, 2015.

[24] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[25] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[26] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[27] Memoona Khanum, Tahira Mahboob, Warda Imtiaz, Humaraia Abdul Ghafoor, and Rabeea Sehar. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *International Journal of Computer Applications*, 119 (13), 2015.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Julian Kupiec, Jan Pedersen, and Francine Chen. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 68–73, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6. doi: 10.1145/215206.215333. URL http://doi.acm.org/10.1145/215206.215333.

[30] Gyoung Ho Lee and Kong Joo Lee. Automatic text summarization using reinforcement learning with embedding features. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 193–197, 2017.

[31] Piji Li, Lidong Bing, and Wai Lam. Actor-critic based training framework for abstractive summarization. *arXiv preprint arXiv:1803.11070*, 2018.

[32] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[33] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

[34] Long Ji Lin. Programming robots using reinforcement learning and teaching. In *AAAI*, pages 781–786, 1991.

[35] Yan Liu, Sheng-hua Zhong, and Wenjie Li. Query-oriented multi-document summarization via unsupervised deep learning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[36] Yang Liu. Fine-tune BERT for Extractive Summarization. mar 2019. URL http://arxiv.org/abs/1903.10318.

[37] David J Livingstone, David T Manallack, and Igor V Tetko. Data modelling with neural networks: advantages and limitations. *Journal of computer-aided molecular design*, 11(2):135–142, 1997.

[38] Elena Lloret and Manuel Palomar. Text summarisation in progress: a literature review. *Artificial Intelligence Review*, 37(1):1–41, 2012.

[39] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[40] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[41] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

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

[43] Ruslan Mitkov. *The Oxford handbook of computational linguistics*. Oxford University Press, 2004.

[44] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[45] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[46] Diego Mollá. Macquarie university at bioasq 5b–query-based summarisation techniques for selecting the ideal answers. In *BioNLP 2017*, pages 67–75, 2017.

[47] Diego Mollá. Macquarie university at bioasq 6b: Deep learning and deep reinforcement learning for query-based multi-document summarisation. *arXiv preprint arXiv:1809.05283*, 2018.

[48] Diego Mollá and Christopher Jones. Classification betters regression in query-based multi-document summarisation techniques for question answering: Macquarie university at bioasq7b. *arXiv preprint arXiv:1909.00542*, 2019.

[49] Diego Mollá, Christopher Jones, and Abeed Sarker. Impact of citing papers for summarisation of clinical documents. In *Proceedings of the Australasian Language Technology Association Workshop 2014*, pages 79–87, 2014.

[50] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

[51] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[52] Shashi Narayan, Shay B Cohen, and Mirella Lapata. Ranking sentences for extractive summarization with reinforcement learning. *arXiv preprint arXiv:1802.08636*, 2018.

[53] You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. Applying regression models to query-focused multi-document summarization. *Information Processing & Management*, 47 (2):227–237, 2011.

[54] Paul Over, Hoa Dang, and Donna Harman. Duc in context. *Information Processing & Management*, 43(6):1506–1520, 2007.

[55] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*, 2017.

[56] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

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

[58] Dragomir R Radev and Kathleen R McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 24(3):470–500, 1998.

[59] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880. ACM, 2009.

[60] Cody Rioux, Sadid A Hasan, and Yllias Chali. Fear the reaper: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 681–690, 2014.

[61] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, England, 1994.

[62] Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

[63] Seonggi Ryang and Takeshi Abekawa. Framework of automatic text summarization using reinforcement learning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265. Association for Computational Linguistics, 2012.

[64] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[65] Abeed Sarker, Diego Molla, and Cecile Paris. Automated text summarisation and evidence-based medicine: A survey of two domains. *arXiv preprint arXiv:1706.08162*, 2017.

[66] Frank Schilder and Ravikumar Kondadadi. Fastsum: fast and accurate query-based multi-document summarization. In *Proceedings of the 46th annual meeting of the association for computational linguistics on human language technologies: Short papers*, pages 205–208. Association for Computational Linguistics, 2008.

[67] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

[68] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[69] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

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

[71] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044, 1996.

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, second edition, 2018.

[73] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[74] George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, et al. An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC bioinformatics*, 16(1):138, 2015.

[75] Fei-Yue Wang, Jun Jason Zhang, Xinhu Zheng, Xiao Wang, Yong Yuan, Xiaoxiao Dai, Jie Zhang, and Liuqing Yang. Where does alphago go: From church-turing thesis to alphago thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*, 3(2):113–120, 2016.

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

[77] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[78] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[79] Yuhuai Wu, Elman Mansimov, Roger B Grosse, Shun Liao, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5279–5288, 2017.

[80] Haoyu Zhang, Yeyun Gong, Yu Yan, Nan Duan, Jianjun Xu, Ji Wang, Ming Gong, and Ming Zhou. Pretraining-based natural language generation for text summarization. *arXiv preprint arXiv:1902.09243*, 2019.

[81] Shansi Zhang and Weiming Wang. Tracking control for mobile robot based on deep reinforcement learning. In *2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS)*, pages 155–160. IEEE, 2019.

[82] Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. Neural latent extractive document summarization. *arXiv preprint arXiv:1808.07187*, 2018.

[83] Lin Zhao, Lide Wu, and Xuanjing Huang. Using query expansion in graph-based approach for query-focused multi-document summarization. *Information Processing & Management*, 45(1):35–41, 2009.

[84] Zhiping Zheng. Answerbus question answering system. In *Proceedings of the second international conference on Human Language Technology Research*, pages 399–404. Morgan Kaufmann Publishers Inc., 2002.

[85] Sheng-hua Zhong, Yan Liu, Bin Li, and Jing Long. Query-oriented unsupervised multi-document summarization via deep learning model. *Expert systems with applications*, 42(21):8146–8155, 2015.