Fear the REAPER: A System for Automatic Multi-Document Summarization with Reinforcement Learning

Cody Rioux
University of Lethbridge
Lethbridge, AB, Canada
cody.rioux@uleth.ca

Sadid A. Hasan
Philips Research North America
Briarcliff Manor, NY, USA
sadid.hasan@philips.com

Yllias Chali
University of Lethbridge
Lethbridge, AB, Canada
chali@cs.uleth.ca

Abstract

This paper explores alternate algorithms, reward functions and feature sets for performing multi-document summarization using reinforcement learning with a high focus on reproducibility. We show that ROUGE results can be improved using a unigram and bigram similarity metric when training a learner to select sentences for summarization. Learners are trained to summarize document clusters based on various algorithms and reward functions and then evaluated using ROUGE. Our experiments show a statistically significant improvement of 1.33%, 1.58%, and 2.25% for ROUGE-1, ROUGE-2 and ROUGE-L scores, respectively, when compared with the performance of the state of the art in automatic summarization with reinforcement learning on the DUC2004 dataset. Furthermore query focused extensions of our approach show an improvement of 1.37% and 2.31% for ROUGE-2 and ROUGE-SU4 respectively over query focused extensions of the state of the art with reinforcement learning on the DUC2006 dataset.

1 Introduction

The multi-document summarization problem has received much attention recently (Lyngbaek, 2013; Sood, 2013; Qian and Liu, 2013) due to its ability to reduce large quantities of text to a human processable amount as well as its application in other fields such as question answering (Liu et al., 2008; Chali et al., 2009a; Chali et al., 2009b; Chali et al., 2011b). We expect this trend to further increase as the amount of linguistic data on the web from sources such as social media, wikipedia, and online newswire increases. This paper focuses specifically on utilizing reinforcement learning (Sutton and Barto, 1998; Szepesv, 2009) to create a policy for summarizing clusters of multiple documents related to the same topic.

The task of extractive automated multi-document summarization (Mani, 2001) is to select a subset of textual units, in this case sentences, from the source document cluster to form a summary of the cluster in question. This extractive approach allows the learner to construct a summary without concern for the linguistic quality of the sentences generated, as the source documents are assumed to be of a certain linguistic quality. This paper aims to expand on the techniques used in Ryang and Abekawa (2012) which uses a reinforcement learner, specifically $TD(\lambda)$, to create summaries of document clusters. We achieve this through introducing a new algorithm, varying the feature space and utilizing alternate reward functions.

The $TD(\lambda)$ learner used in Ryang and Abekawa (2012) is a very early reinforcement learning implementation. We explore the option of leveraging more recent research in reinforcement learning algorithms to improve results. To this end we explore the use of $SARSA$ which is a derivative of $TD(\lambda)$ that models the action space in addition to the state space modelled by $TD(\lambda)$. Furthermore we explore the use of an algorithm not based on temporal difference methods, but instead on policy iteration techniques. Approximate Policy Iteration (Lagoudakis and Parr, 2003) generates a policy, then evaluates and iterates until convergence.

The reward function in Ryang and Abekawa (2012) is a delayed reward based on $tf{*}idf$ values. We further explore the reward space by introducing similarity metric calculations used in ROUGE (Lin, 2004) and base our ideas on Saggion et al. (2010). The difference between immediate rewards and delayed rewards is that the learner re-

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 681–690,
October 25-29, 2014, Doha, Qatar. ©2014 Association for Computational Linguistics

ceives immediate feedback at every action in the former and feedback only at the end of the episode in the latter. We explore the performance difference of both reward types. Finally we develop query focused extensions to both reward functions and present their results on more recent Document Understanding Conference (DUC) datasets which ran a query focused task.

We first evaluate our systems using the DUC2004 dataset for comparison with the results in Ryang and Abekawa (2012). We then present the results of query focused reward functions against the DUC2006 dataset to provide reference with a more recent dataset and a more recent task, specifically a query-focused summarization task. Evaluations are performed using ROUGE for ROUGE-1, ROUGE-2 and ROUGE-L values for general summarization, while ROUGE-2 and ROUGE-SU4 is used for query-focused summarization. Furthermore we selected a small subset of query focused summaries to be subjected to human evaluations and present the results.

Our implementation is named REAPER (**R**elatedness-focused **E**xtractive **A**utomatic summary **P**reparation **E**xploiting **R**einfocement learning) thusly for its ability to harvest a document cluster for ideal sentences for performing the automatic summarization task. REAPER is not just a reward function and feature set, it is a full framework for implementing summarization tasks using reinforcement learning and is available online for experimentation.[1] The primary contributions of our experiments are as follows:

- Exploration of $TD(\lambda)$, $SARSA$ and Approximate Policy Iteration.
- Alternate REAPER reward function.
- Alternate REAPER feature set.
- Query focused extensions of automatic summarization using reinforcement learning.

2 Previous Work and Motivation

Previous work using reinforcement learning for natural language processing tasks (Branavan et al., 2009; Wan, 2007; Ryang and Abekawa, 2012; Chali et al., 2011a; Chali et al., 2012) inspired us to use a similar approach in our experiments. Ryang and Abekawa (2012) implemented a reinforcement learning approach to multi-document summarization which they named Automatic Summarization using Reinforcement Learning (ASRL). ASRL uses TD(λ) to learn and then execute a policy for summarizing a cluster of documents. The algorithm performs N summarizations from a blank state to termination, updating a set of state-value predictions as it does so. From these N episodes a policy is created using the estimated state-value pairs, this policy greedily selects the best action until the summary enters its terminal state. This summary produced is the output of ASRL and is evaluated using ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004). The results segment of the paper indicates that ASRL outperforms greedy and integer linear programming (ILP) techniques for the same task.

There are two notable details that provide the motivation for our experiments; TD(λ) is relatively old as far as reinforcement learning (RL) algorithms are concerned, and the optimal ILP did not outperform ASRL using the same reward function. The intuition gathered from this is that if the optimal ILP algorithm did not outperform the suboptimal ASRL on the ROUGE evaluation, using the same reward function, then there is clearly room for improvement in the reward function's ability to accurately model values in the state space. Furthermore one may expect to achieve a performance boost exploiting more recent research by utilizing an algorithm that intends to improve upon the concepts on which $TD(\lambda)$ is based. These provide the motivation for the remainder of the research preformed.

Query focused multi-document summarization (Li and Li, 2013; Chali and Hasan, 2012b; Yin et al., 2012; Wang et al., 2013) has recently gained much attention due to increasing amounts of textual data, as well as increasingly specific user demands for extracting information from said data. This is reflected in the query focused tasks run in the Document Understanding Conference (DUC) and Text Analysis Conference (TAC) over the past decade. This has motivated us to design and implement query focused extensions to these reinforcement learning approaches to summarization.

There has been some research into the effects of sentence compression on the output of automatic summarization systems (Chali and Hasan, 2012a), specifically the evaluation results garnered from compressing sentences before evaluation (Qian and Liu, 2013; Lin and Rey, 2003; Ryang and

[1] https://github.com/codyrioux/REAPER

Abekawa, 2012). However Ryang and Abekawa (2012) found this technique to be ineffective in improving ROUGE metrics using a similar reinforcement learning approach to this paper, as a result we will not perform any further exploration into the effects of sentence compression.

3 Problem Definition

We use an identical problem definition to Ryang and Abekawa (2012). Assume the given cluster of documents is represented as a set of textual units $D = \{x_1, x_2, \cdots, x_n\}$ where $|D| = n$ and x_i represents a single textual unit. Textual units for the purposes of this experiment are the individual sentences in the document cluster, that is $D = D_1 \cup D_2 \cup \cdots \cup D_m$ where m is the number of documents in the cluster and each D_i represents a document.

The next necessary component is the score function, which is to be used as the reward for the learner. The function $score(s)$ can be applied to any $s \subset D$. s is a summary of the given document or cluster of documents.

Given these parameters, and a length limitation k we can define an optimal summary s^* as:

$$\begin{aligned} s^* = argmax\ score(s) \\ where\ s \subset D\ and\ length(s) \leq k \end{aligned} \tag{1}$$

It is the objective of our learner to create a policy that produces the optimal summary for its provided document cluster D. Henceforth the length limitations used for general summarization will be 665 bytes, and query focused summarization will use 250 words. These limitations on summary length match those set by the Document Understanding Conferences associated with the dataset utilized in the respective experiments.

4 Algorithms

$TD(\lambda)$ and $SARSA$ (Sutton and Barto, 1998) are temporal difference methods in which the primary difference is that $TD(\lambda)$ models state value predictions, and $SARSA$ models state-action value predictions. Approximate Policy Iteration (API) follows a different paradigm by iteratively improving a policy for a markov decision process until the policy converges.

4.1 TD(λ)

In the ASRL implementation of $TD(\lambda)$ the learning rate α_k and temperature τ_k decay as learning progresses with the following equations with k set to the number of learning episodes that had taken place.

$$\alpha_k = 0.001 \cdot 101/(100 + k^{1.1}) \tag{2}$$

$$\tau_k = 1.0 * 0.987^{k-1} \tag{3}$$

One can infer from the decreasing values of α_k that as the number of elapsed episodes increases the learner adjusts itself at a smaller rate. Similarly as the temperature τ_k decreases the action selection policy becomes greedier and thus performs less exploration, this is evident in (5) below.

Note that unlike traditional $TD(\lambda)$ implementations the eligibility trace e resets on every episode. The reasons for this will become evident in the experiments section of the paper in which $\lambda = 1, \gamma = 1$ and thus there is no decay during an episode and complete decay after an episode. The same holds true for $SARSA$ below.

The action-value estimation $Q(s, a)$ is approximated as:

$$Q(s, a) = r + \gamma V(s') \tag{4}$$

The policy is implemented as such:

$$policy(a|s; \theta; \tau) = \frac{e^{Q(s,a)/\tau}}{\sum_{a \in A} e^{Q(s,a)/\tau}} \tag{5}$$

Actions are selected probabilistically using softmax selection (Sutton and Barto, 1998) from a Boltzmann distribution. As the value of τ approaches 0 the distribution becomes greedier.

4.2 $SARSA$

$SARSA$ is implemented in a very similar manner and shares α_k, τ_k, $\phi(s)$, m, and $policy(s)$ with the $TD(\lambda)$ implementation above. $SARSA$ is also a temporal difference algorithm and thus behaves similarly to $TD(\lambda)$ with the exception that values are estimated not only on the state s but a state-action pair $[s, a]$.

4.3 Approximate Policy Iteration

The third algorithm in our experiment uses Approximate Policy Iteration (Lagoudakis and Parr, 2003) to implement a reinforcement learner. The novelty introduced by (Lagoudakis and Parr, 2003) is that they eschew standard representations for a policy and instead use a classifier to represent the current policy π. Further details on the algorithm can be obtained from Lagoudakis and Parr (2003).

5 Experiments

Our state space S is represented simply as a three-tuple $[s\ a\ f]$ in which s is the set of textual units (sentences) that have been added to the summary, a is a sequence of actions that have been performed on the summary and f is a boolean with value 0 representing non-terminal states and 1 representing a summary that has been terminated.

The individual units in our action space are defined as [:insert x_i] where x_i is a textual unit as described earlier, let us define D_i as the set [:insert x_i] for all $x_i \in D$ where D is the document set. We also have one additional action [:finish] and thus we can define our action space.

$$A = D_i \cup \{[: finish]\} \tag{6}$$

The actions eligible to be executed on any given state s is defined by a function $actions(A, s)$:

$$actions(A, s) = \begin{cases} [: finish] & \text{if } length(s) > k \\ A - a_t & otherwise \end{cases} \tag{7}$$

The state-action transitions are defined below:

$$[s_t\ , a_t\ , 0] \xrightarrow{a=insertx_i} [s_t \cup x_i\ , a_t \cup a\ , 0] \tag{8}$$

$$[s_t\ , a_t\ , 0] \xrightarrow{:finish} [s_t\ , a_t\ , 1] \tag{9}$$

$$[s_t\ , a_{t+1}\ , 1] \xrightarrow{any} [s_t\ , a_t\ , 1] \tag{10}$$

Insertion adds both the content of the textual unit x_i to the set s as well as the action itself to set a. Conversely finishing does not alter s or a but it flips the f bit to on. Notice from (10) that once a state is terminal any further actions have no effect.

5.1 Feature Space

We present an alternate feature set called REAPER feature set based on the ideas presented in Ryang and Abekawa (2012). Our proposed feature set follows a similar format to the previous one but depends on the presence of top **bigrams** instead of $tf * idf$ words.

- One bit $b \in 0, 1$ for each of the top n **bigrams** (Manning and Schütze, 1999) present in the summary.
- Coverage ratio calculated as the sum of the bits in the previous feature divided by n.
- Redundancy Ratio calculated as the number of redundant times a bit in the first feature is flipped on, divided by n.
- Length Ratio calculated as $length(s)/k$ where k is the length limit.
- Longest common subsequence length.
- Length violation bit. Set to 1 if $length(s) > k$

Summaries which exceed the length limitation k are subject to the same reduction as the ASRL feature set (Ryang and Abekawa, 2012) to an all zero vector with the final bit set to one.

5.2 Reward Function

Our reward function (termed as REAPER reward) is based on the n-gram concurrence score metric, and the longest-common-subsequence recall metric contained within ROUGE (Lin, 2004).

$$reward(s) = \begin{cases} -1, & \text{if } length(s) > k \\ score(s) & \text{if } s \text{ is terminal} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

Where $score$ is defined identically to ASRL, with the exception that Sim is a new equation based on ROUGE metrics.

$$score(s) = \sum_{x_i \in S} \lambda_s Rel(x_i) - \sum_{x_i, x_j \in S, i<j} (1 - \lambda_s) Red(x_i, x_j) \tag{12}$$

$$Rel(x_i) = Sim(x_i, D) + Pos(x_i)^{-1} \tag{13}$$

$$Red(x_i, x_j) = Sim(x_i, x_j) \tag{14}$$

Ryang and Abekawa (2012) experimentally determined a value of 0.9 for the λ_s parameter. That value is used herein unless otherwise specified. Sim has been redefined as:

$$\begin{aligned} Sim(s) =& 1 * ngco(1, D, s) + \\ & 4 * ngco(2, D, s) + \\ & 1 * ngco(3, D, s) + \\ & 1 * ngco(4, D, s) + \\ & 1 * rlcs(D, s) \end{aligned} \tag{15}$$

and $ngco$ is the ngram co-occurence score metric as defined by Lin (2004).

$$ngco(n, D, s) = \frac{\sum_{r \in D} \sum_{ngram \in r} Count_{match}(ngram)}{\sum_{S_r \in D} \sum_{ngram \in r} Count(ngram)} \tag{16}$$

Where n is the n-gram count for example 2 for bigrams, D is the set of documents, and s is the summary in question. $Count_{match}$ is the maximum number of times the ngram occurred in either D or s.

The $rlcs(R, S)$ is also a recall oriented measure based on longest common subsequence (Hirschberg, 1977). Recall was selected as DUC2004 tasks favoured a β value for F-Measure (Lin, 2004) high enough that only recall would be considered. lcs is the longest common subsequence, and $length(D)$ is the total number of tokens in the reference set D.

$$rlcs(D, s) = \frac{lcs(D, s)}{length(D)} \tag{17}$$

We are measuring similarity between sentences and our entire reference set, and thusly our D is the set of documents defined in section 3. This is also a delayed reward as the provided reward is zero until the summary is terminal.

5.2.1 Query Focused Rewards

We have proposed an extension to both reward functions to allow for query focused (QF) summarization. We define a function $score'$ which aims to balance the summarization abilities of the reward with a preference for selecting textual units related to the provided query q. Both ASRL and REAPER $score$ functions have been extended in the following manner where Sim is the same similarity functions used in equation (13) and (15).

$$score'(q, s) = \beta Sim(q, s) + (1 - \beta) score(s) \tag{18}$$

The parameter β is a balancing factor between query similarity and overall summary score in which $0 <= \beta <= 1$, we used an arbitrarily chosen value of 0.9 in these experiments. In the case of ASRL the parameter q is the vectorized version of the query function with $tf * idf$ values, and for Sim q is a sequence of tokens which make up the query, stemmed and stop-words removed.

5.2.2 Immediate Rewards

Finally we also employ immediate versions of the reward functions which behave similarly to their delayed counterparts with the exception that the score is always provided to the caller regardless of the terminal status of state s.

$$reward(s) = \begin{cases} -1, & \text{if } length(s) > k \\ score(s) & \text{otherwise} \end{cases} \tag{19}$$

6 Results

We first present results[2] of our experiments, specifying parameters, and withholding discussion until the following section. We establish a benchmark using ASRL and other top-scoring summarization systems compared with REAPER using ROUGE. For generic multi-document summarization we run experiments on all 50 document clusters, each containing 10 documents, of DUC2004 task 2 with parameters for REAPER and ASRL fixed at $\lambda = 1$, $\gamma = 1$, and $k = 665$. Sentences were stemmed using a Porter Stemmer (Porter, 1980) and had the ROUGE stop word set removed. All summaries were processed in this manner and then projected back into their original (unstemmed, with stop-words) state and output to disk.

Config	R-1	R-2	R-L
REAPER	0.40339	0.11397	0.36574
ASRL	0.39013	0.09479	0.33769
MCKP	0.39033	0.09613	0.34225
PEER65	0.38279	0.09217	0.33099
ILP	0.34712	0.07528	0.31241
GREEDY	0.30618	0.06400	0.27507

Table 1: Experimental results with ROUGE-1, ROUGE-2 and ROUGE-L scores on DUC2004.

[2] ROUGE-1.5.5 run with -m -s -p 0

Table 1 presents results for REAPER, ASRL (Ryang and Abekawa, 2012), MCKP (Takamura and Okumura, 2009), PEER65 (Conroy et al., 2004) , and GREEDY (Ryang and Abekawa, 2012) algorithms on the same task. This allows us to make a direct comparison with the results of Ryang and Abekawa (2012).

REAPER results are shown using the $TD(\lambda)$ algorithm, REAPER reward function, and ASRL feature set. This is to establish the validity of the reward function holding all other factors constant. REAPER results for ROUGE-1, ROUGE-2 and ROUGE-L are statistically significant compared to the result set presented in Table 2 of Ryang and Abekawa (2012) using $p < 0.01$.

Run	**R-1**	**R-2**	**R-L**
REAPER 0	0.39536	0.10679	0.35654
REAPER 1	0.40176	0.11048	0.36450
REAPER 2	0.39272	0.11171	0.35766
REAPER 3	0.39505	0.11021	0.35972
REAPER 4	0.40259	0.11396	0.36539
REAPER 5	0.40184	0.11306	0.36391
REAPER 6	0.39311	0.10873	0.35481
REAPER 7	0.39814	0.11001	0.35786
REAPER 8	0.39443	0.10740	0.35586
REAPER 9	0.40233	0.11397	0.36483
Average	0.39773	0.11063	0.36018

Table 2: REAPER run 10 times on the DUC2004.

We present the results of 10 runs of REAPER, with REAPER feature set.. As with ASRL, REAPER does not converge on a stable solution which is attributable to the random elements of $TD(\lambda)$. Results in all three metrics are again statistically significant compared to ASRL results presented in the Ryang and Abekawa (2012) paper. All further REAPER experiments use the bigram oriented feature space.

Reward	**R-1**	**R-2**	**R-L**
$Delayed$	0.39773	0.11397	0.36018
$Immediate$	0.32981	0.07709	0.30003

Table 3: REAPER with delayed and immediate rewards on DUC2004.

Table 3 shows the performance difference of REAPER when using a delayed and immediate reward. The immediate version of REAPER provides feedback on every learning step, unlike the delayed version which only provides score at the end of the episode.

Features	**R-1**	**R-2**	**R-L**
$ASRL$	0.40339	0.11397	0.36574
$REAPER$	0.40259	0.11396	0.36539

Table 4: REAPER with alternate feature spaces on DUC2004.

We can observe the results of using REAPER with various feature sets in Table 4. Experiments were run using REAPER reward, $TD(\lambda)$, and the specified feature set.

Algorithm	**R-1**	**R-2**	**R-L**
$TD(\lambda)$	0.39773	0.11063	0.36018
$SARSA$	0.28287	0.04858	0.26172
API	0.29163	0.06570	0.26542

Table 5: REAPER with alternate algorithms on DUC2004.

Table 5 displays the performance of REAPER with alternate algorithms. $TD(\lambda)$ and $SARSA$ are run using the delayed reward feature, while API requires an immediate reward and was thus run with the immediate reward.

System	**R-2**	**R-SU4**
$REAPER$	0.07008	0.11689
$ASRL$	0.05639	0.09379
$S24$	0.09505	0.15464
Baseline	0.04947	0.09788

Table 6: QF-REAPER on DUC2006.

For query-focused multi-document summarization we experimented with the DUC2006 system task, which contained 50 document clusters consisting of 25 documents each. Parameters were fixed to $\lambda = 1$, $\gamma = 1$ and $k = 250$ **words**. In Table 6 we can observe the results[3] of our query focused systems against DUC2006's top scorer (S24) for ROUGE-2, and a baseline. The baseline was generated by taking the most recent document in the cluster and outputting the first 250 words.

Human Evaluations: We had three native English-speaking human annotators evaluate a set of four randomly chosen summaries produced by REAPER on the DUC2004 dataset.

[3] ROUGE-1.5.5 run with -n 2 -x -m -2 4 -u -c 95 -r 1000 -f A -p 0.5 -t 0 -l 250

Metric	A1	A2	A3	AVG
Grammaticality	3.00	4.00	4.00	**3.67**
Redundancy	4.75	4.25	2.75	**3.92**
Referential Clarity	4.00	4.50	3.50	**4.00**
Focus	4.50	3.50	2.25	**3.42**
Structure	3.50	4.00	3.00	**3.50**
Responsiveness	4.25	3.75	3.00	**3.67**

Table 7: Human evaluation scores on DUC2004.

Table 7 shows the evaluation results according to the DUC2006 human evaluation guidelines. The first five metrics are related entirely to the linguistic quality of the summary in question and the final metric, Responsiveness, rates the summary on its relevance to the source documents. Columns represent the average provided by a given annotator over the four summaries, and the AVG column represents the average score for all three annotators over all four summaries. Score values are an integer between 1 and 5 inclusive.

7 Discussion

First we present a sample of a summary generated from a randomly selected cluster. The following summary was generated from cluster $D30017$ of the DUC 2004 dataset using REAPER reward with $TD(\lambda)$ and REAPER feature space.

A congressman who visited remote parts of North Korea last week said Saturday that the food and health situation there was desperate and deteriorating, and that millions of North Koreans might have starved to death in the last few years. North Korea is entering its fourth winter of chronic food shortages with its people malnourished and at risk of dying from normally curable illnesses, senior Red Cross officials said Tuesday. More than five years of severe food shortages and a near-total breakdown in the public health system have led to devastating malnutrition in North Korea and probably left an entire generation of children physically and mentally impaired, a new study by international aid groups has found. Years of food shortages have stunted the growth of millions of North Korean children, with two-thirds of children under age seven suffering malnourishment, U.N. experts said Wednesday. The founder of South Korea's largest conglomerate plans to visit his native North Korea again next week with a gift of 501 cattle, company officials said Thursday. "There is enough rice.

We can observe that the summary is both syntactically sound, and elegantly summarizes the source documents.

Our baseline results table (Table 1) shows REAPER outperforming ASRL in a statistically significant manner on all three ROUGE metrics in question. However we can see from the absolute differences in score that very few additional important words were extracted (ROUGE-1) however REAPER showed a significant improvement in the structuring and ordering of those words (ROUGE-2, and ROUGE-L).

The balancing factors used in the REAPER reward function are responsible for the behaviour of the reward function, and thus largely responsible for the behaviour of the reinforcement learner. In equation 15 we can see balance numbers of 1, 4, 1, 1, 1 for 1-grams, 2-grams, 3-grams, 4-grams, and LCS respectively. In adjusting these values a user can express a preference for a single metric or a specific mixture of these metrics. Given that the magnitude of scores for n-grams decrease as n increases and given that the magnitude of scores for 1-grams is generally 3 to 4 times larger, in our experience, we can see that this specific reward function favours bigram similarity over unigram similarity. These balance values can be adjusted to suit the specific needs of a given situation, however we leave exploration of this concept for future work.

We can observe in Figure 1 that ASRL does not converge on a stable value, and dips towards the 300^{th} episode while in Figure 2 REAPER does not take nearly such a dramatic dip. These figures display average normalized reward for all 50 document clusters on a single run. Furthermore we can observe that ASRL reaches it's peak reward around episode 225 while REAPER does so around episode 175 suggesting that REAPER converges faster.

7.1 Delayed vs. Immediate Rewards

The delayed vs. immediate rewards results in Table 3 clearly show that delaying the reward provides a significant improvement in globally optimizing the summary for ROUGE score. This can be attributed to the $\lambda = 1$ and $\gamma = 1$ parameter values being suboptimal for the immediate reward situation. This has the added benefit of being much more performant computationally as far fewer reward calculations need be done.

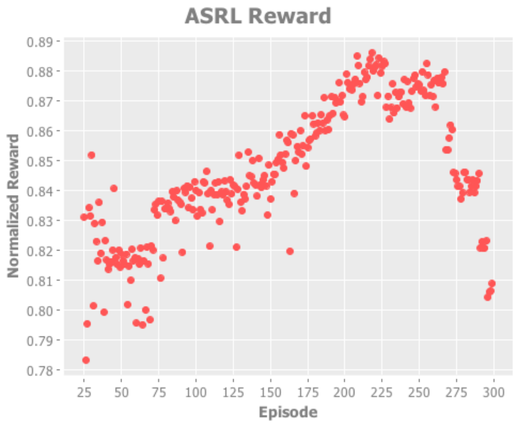

Figure 1: ASRL normalized reward.

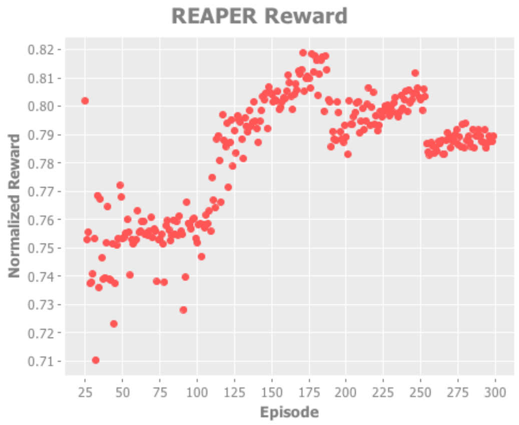

Figure 2: REAPER normalized reward.

7.2 Feature Space

The feature space experiments in Table 4 seem to imply that REAPER performs similarly with both feature sets. We are confident that an improvement could be made through further experimentation. Feature engineering, however, is a very broad field and we plan to pursue this topic in depth in the future.

7.3 Algorithms

$TD(\lambda)$ significantly outperformed both $SARSA$ and API in the algorithm comparison. Ryang and Abekawa (2012) conclude that the feature space is largely responsible for the algorithm performance. This is due to the fact that poor states such as those that are too long, or those that contain few important words will reduce to the same feature set and receive negative rewards collectively. $SARSA$ loses this benefit as a result of its modelling of state-action pairs.

API on the other hand may have suffered a performance loss due to its requirements of an immediate reward, this is because when using a delayed reward if the trajectory of a rollout does not reach a terminal state the algorithm will not be able to make any estimations about the value of the state in question. We propose altering the policy iteration algorithm to use a trajectory length of one episode instead of a fixed number of actions in order to counter the need for an immediate reward function.

7.4 Query Focused Rewards

From the ROUGE results in Table 6 we can infer that while REAPER outperformed ASRL on the query focused task, however it is notable that both systems under performed when compared to the top system from the DUC2006 conference.

We can gather from these results that it is not enough to simply naively calculate similarity with the provided query in order to produce a query-focused result. Given that the results produced by the generic summarization task is rather acceptable according to our human evaluations we suggest that further research be focused on a proper similarity metric between the query and summary to improve the reward function's overall ability to score summaries in a query-focused setting.

8 Conclusion and Future Work

We have explored alternate reward functions, feature sets, and algorithms for the task of automatic summarization using reinforcement learning. We have shown that REAPER outperforms ASRL on both generic summarization and the query focused tasks. This suggests the effectiveness of our reward function and feature space. Our results also confirm that $TD(\lambda)$ performs best for this task compared to $SARSA$ and API.

Due to the acceptable human evaluation scores on the general summarization task it is clear that the algorithm produces acceptable summaries of newswire data. Given that we have a framework for generating general summaries, and the current popularity of the query-focused summarization task, we propose that the bulk of future work in this area be focused on the query-focused task specifically in assessing the relevance of a summary to a provided query. Therefore we intend to pursue future research in utilizing word-sense disambiguation and synonyms, as well as other techniques for furthering REAPER's query similarity metrics in order to improve its ROUGE and human evaluation scores on query-focused tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their useful comments. The research reported in this paper was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada - discovery grant and the University of Lethbridge. This work was done when the second author was at the University of Lethbridge.

References

S . Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement Learning for Mapping Instructions to Actions. *In Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 82–90.

Y. Chali and S. A. Hasan. 2012a. On the Effectiveness of Using Sentence Compression Models for Query-Focused Multi-Document Summarization. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 457–474. Mumbai, India.

Y. Chali and S. A. Hasan. 2012b. Query-focused Multi-document Summarization: Automatic Data Annotations and Supervised Learning Approaches. *Journal of Natural Language Engineering*, 18(1):109–145.

Y. Chali, S. A. Hasan, and S. R. Joty. 2009a. Do Automatic Annotation Techniques Have Any Impact on Supervised Complex Question Answering? *Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP 2009)*, pages 329–332.

Y. Chali, S. R. Joty, and S. A. Hasan. 2009b. Complex Question Answering: Unsupervised Learning Approaches and Experiments. *Journal of Artificial Intelligence Research*, 35:1–47.

Y. Chali, S. A. Hasan, and K. Imam. 2011a. A Reinforcement Learning Framework for Answering Complex Questions. In *Proceedings of the 16th International Conference on Intelligent User Interfaces*, pages 307–310. ACM, Palo Alto, CA, USA.

Y. Chali, S. A. Hasan, and S. R. Joty. 2011b. Improving Graph-based Random Walks for Complex Question Answering Using Syntactic, Shallow Semantic and Extended String Subsequence Kernels. *Information Processing and Management (IPM), Special Issue on Question Answering*, 47(6):843–855.

Y. Chali, S. A. Hasan, and K. Imam. 2012. Improving the Performance of the Reinforcement Learning Model for Answering Complex Questions. In *Proceedings of the 21st ACM Conference on Information and Knowledge Management (CIKM 2012)*, pages 2499–2502. ACM, Maui, Hawaii, USA.

J. Conroy, J. Goldstein, and D. Leary. 2004. Left-Brain / Right-Brain Multi-Document Summarization. *In Proceedings of the Document Un- derstanding Conference (DUC 2004).*

D. S. Hirschberg. 1977. Algorithms for the Longest Common Subsequence Problem. *In Journal of the ACM*, 24(4):664–675, October.

M. Lagoudakis and R. Parr. 2003. Reinforcement learning as classification: Leveraging modern classifiers. *In Proceedings of the Twentieth International Conference on Machine Learning*, 20(1):424.

J. Li and S. Li. 2013. A Novel Feature-based Bayesian Model for Query Focused Multi-document Summarization. *In Transactions of the Association for Computational Linguistics*, 1:89–98.

C. Lin and M. Rey. 2003. Improving Summarization Performance by Sentence Compression A Pilot Study. *In Proceedings of the Sixth International Workshop on Information Retrieval with Asian Languages.*

C. Lin. 2004. ROUGE : A Package for Automatic Evaluation of Summaries. *In Information Sciences*, 16(1):25–26.

Y. Liu, S. Li, Y. Cao, C. Lin, D. Han, and Y. Yu. 2008. Understanding and Summarizing Answers in Community-Based Question Answering Services. *In COLING '08 Proceedings of the 22nd International Conference on Computational Linguistics*, 1(August):497–504.

S. Lyngbaek. 2013. *SPORK: A Summarization Pipeline for Online Repositories of Knowledge*. M.sc. thesis, California Polytechnic State University.

I. Mani. 2001. *Automatic Summarization*. John Benjamins Publishing.

C. D. Manning and H. Schütze. 1999. *Foundations of Statistical Natural Language Processing*, volume 26 of . MIT Press.

M. F. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

X. Qian and Y. Liu. 2013. Fast Joint Compression and Summarization via Graph Cuts. *In Conference on Empirical Methods in Natural Language Processing.*

S. Ryang and T. Abekawa. 2012. 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*, 1:256–265.

H. Saggion, C. Iria, T. M. Juan-Manuel, and E. San-Juan. 2010. Multilingual Summarization Evaluation without Human Models. *In COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 1:1059–1067.

A. Sood. 2013. *Towards Summarization of Written Text Conversations*. M.sc. thesis, International Institute of Information Technology, Hyderabad, India.

R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

C. A. Szepesv. 2009. *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers.

H. Takamura and M. Okumura. 2009. Text Summarization Model based on Maximum Coverage Problem and its Variant. *In EACL '09 Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, (April):781–789.

X Wan. 2007. Towards an Iterative Reinforcement Approach for Simultaneous Document Summarization and Keyword Extraction. *In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 552–559.

L. Wang, H. Raghavan, V. Castelli, R. Florian, and C. Cardie. 2013. A Sentence Compression Based Framework to Query-Focused. *In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 1:1384–1394.

W. Yin, Y. Pei, F. Zhang, and L. Huang. 2012. Query-focused multi-document summarization based on query-sensitive feature space. *In Proceedings of the 21st ACM international conference on Information and knowledge management - CIKM '12*, page 1652.