Framework of Automatic Text Summarization Using Reinforcement Learning

Seonggi Ryang
Graduate School of Information Science and Technology
University of Tokyo
sryang@is.s.u-tokyo.ac.jp

Takeshi Abekawa
National Institute of Informatics
abekawa@nii.ac.jp

Abstract

We present a new approach to the problem of automatic text summarization called Automatic Summarization using Reinforcement Learning (ASRL) in this paper, which models the process of constructing a summary within the framework of reinforcement learning and attempts to optimize the given score function with the given feature representation of a summary. We demonstrate that the method of reinforcement learning can be adapted to automatic summarization problems naturally and simply, and other summarizing techniques, such as sentence compression, can be easily adapted as actions of the framework.

The experimental results indicated ASRL was superior to the best performing method in DUC2004 and comparable to the state of the art ILP-style method, in terms of ROUGE scores. The results also revealed ASRL can search for sub-optimal solutions efficiently under conditions for effectively selecting features and the score function.

1 Introduction

Automatic text summarization aims to automatically produce a short and well-organized summary of single or multiple documents (Mani, 2001). Automatic summarization, especially multi-document summarization, has been an increasingly important task in recent years, because of the exponential explosion of available information. The brief summary that the summarization system produces allows readers to quickly and easily understand the content of original documents without having to read each individual document, and it should be helpful for dealing with enormous amounts of information.

The extractive approach to automatic summarization is a popular and well-known approach in this field, which creates a summary by directly selecting some textual units (e.g., words and sentences) from the original documents, because it is difficult to genuinely evaluate and guarantee the linguistic quality of the produced summary.

One of the most well-known extractive approaches is maximal marginal relevance (MMR), which scores each textual unit and extracts the unit that has the highest score in terms of the MMR criteria (Goldstein et al., 2000). Greedy MMR-style algorithms are widely used; however, they cannot take into account the whole quality of the summary due to their greediness, although a summary should convey all the information in given documents. Global inference algorithms for the extractive approach have been researched widely in recent years (Filatova and Hatzivassiloglou, 2004; McDonald, 2007; Takamura and Okumura, 2009) to consider whether the summary is "good" as a whole. These algorithms formulate the problem as integer linear programming (ILP) to optimize the score: however, as ILP is non-deterministic polynomial-time hard (NP-hard), the time complexity is very large. Consequently, we need some more efficient algorithm for calculations.

We present a new approach to the problem of automatic text summarization called Automatic Summarization using Reinforcement Learning (ASRL), which models the process of construction of a summary within the framework of reinforcement learn-

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 256–265, Jeju Island, Korea, 12–14 July 2012.

ing and attempts to optimize the given score function with the given feature representation of a summary. We demonstrate that the method of reinforcement learning can be adapted to problems with automatic summarization naturally and simply, and other summarizing techniques, such as sentence compression, can be easily adapted as actions of the framework, which should be helpful to enhance the quality of the summary that is produced. This is the first paper utilizing reinforcement learning for problems with automatic summarization of text.

We evaluated ASRL with the DUC2004 summarization task 2, and the experimental results revealed ASRL is superior to the best method of performance in DUC2004 and comparable with the state of the art ILP-style method, based on maximum coverage with the knapsack constraint problem, in terms of ROUGE scores with experimental settings. We also evaluated ASRL in terms of optimality and execution time. The experimental results indicated ASRL can search the state space efficiently for some suboptimal solutions under the condition of effectively selecting features and the score function, and produce a summary whose score denotes the expectation of the score of the same features' states. The evaluation of the quality of a produced summary only depends on the given score function, and therefore it is easy to adapt the new method of evaluation without having to modify the structure of the framework.

2 Formulation of Extractive Approach

We first focus on the extractive approach, which is directly used to produce a summary by extracting some textual units, by avoiding the difficulty of having to consider the genuine linguistic quality of a summary.

The given document (or documents) in extractive summarization approaches is reduced to the set of textual units: $D = \{x_1, x_2, \cdots, x_n\}$, where n is the size of the set, and x_i denotes individual textual units. Note that any textual unit is permitted, such as *character*, *word*, *sentence*, *phrase*, and *conceptual unit*. If we determine a *sentence* is a textual unit to be extracted, the formulated problem is a problem of extracting sentences from the source document, which is one of the most popular settings for summarization tasks.

Next, we define the score function, $score(S)$, for any subset of the document: $S \subset D$. Subset S is one of the summaries of the given document. The aim of this summarization problem is to find the summary that maximizes this function when the score function is given. The score function is typically defined by taking into consideration the tradeoff between relevance and redundancy.

Then, we define length function $L(S)$, which indicates the length of summary S. The length is also arbitrary, which can be based on the *character*, *word*, and *sentence*. We assume the limitation of summary length K is given in summarization tasks.

Finally, we define the extractive approach of the automatic summarization problem as:

$$\begin{aligned} S^* &= \arg\max_{S \subset D} score(S) \\ \text{s.t.} \quad & L(S) \leq K. \end{aligned} \tag{1}$$

3 Motivation

We can regard the extractive approach as a search problem. It is extremely difficult to solve this search problem because the final result of evaluation given by the given score function is not available until it finishes, and we therefore need to try all combinations of textual units. Consequently, the score function, which denotes some criterion for the quality of a summary, tends to be determined so that the function can be decomposed to components and it is solved with global inference algorithms, such as ILP. However, both decomposing the score function properly and utilizing the evaluation of half-way process of searches are generally difficult. For example, let us assume that we design the score function by using some complex semantic considerations to take into account the readability of a summary, and the score is efficiently calculated if the whole summary is given. Then, formulating the problem as a global inference problem and solving it with methods of integer linear programming might generally be difficult, because of the complex composition of the score function, despite the ease with which the whole summary is evaluated. The readability score might be based on extremely complex calculations of dependency relations, or a great deal of external knowledge the summarizer cannot know

merely from the source documents. In fact, it is ideal that we can only directly utilize the score function, in the sense that we do not have to consider the decomposed form of the given score function.

We need to consider the problem with automatic summarization to be the same as that with reinforcement learning to handle these problems. Reinforcement learning is one of the solutions to three problems.

- The learning of the agent only depends on the reward provided by the environment.
- Furthermore, the reward is delayed, in the sense that the agent cannot immediately know the actual evaluation of the executed action.
- The agent only estimates the value of the state with the information on rewards, without knowledge of the actual form of the score function, to maximize future rewards.

We suggest the formulation of the problem as we have just described will enable us to freely design the score function without limitations and expand the capabilities of automatic summarization.

4 Models of Extractive Approach for Reinforcement Learning

4.1 Reinforcement Learning

Reinforcement learning is a powerful method of solving planning problems, especially problems formulated as Markov decision processes (MDPs) (Sutton and Barto, 1998). The agent of reinforcement learning repeats three steps until terminated at each episode in the learning process.

1. The agent observes current *state* s from the *environment*, contained in state space $\mathcal{S}$.
2. Next, it determines and executes next *action* a according to current policy π. Action a is contained in the action space limited by the current state: $\mathcal{A}(s)$, which is a subset of whole action space $\mathcal{A} = \bigcup_{s \in S} \mathcal{A}(s)$. Policy π is the strategy for selecting action, represented as a conditional distribution of actions: $p(a|s)$.
3. It then observes next state s' and receives *reward* r from the environment.

The aim of reinforcement learning is to find optimal policy π^* only with information on sample trajectories and to reward the experienced agent.

We describe how to adapt the extractive approach to the problem of reinforcement learning in the sections that follow.

4.2 State

A state denotes a summary. We represent state s as a tuple of summary S (a set of textual units) and additional state variables: $s = (S, A, f)$. We assume s has the history of actions A that the agent executed to achieve this state. Additionally, s has the binary state variable, $f \in \{0, 1\}$, which denotes whether s is a terminal state or not. Initial state s_0 is $(\emptyset, \emptyset, 0)$.

We assume the d-dimensional feature representation of state s: $\phi(s) \in \mathbb{R}^d$, which only depends on the feature of summary $\phi'(S) \in \mathbb{R}^{d-1}$. Given $\phi'(S)$, we define the features as:

$$\phi(s) = \begin{cases} (\phi'(S), 0)^{\mathrm{T}} & (L(S) \leq K) \\ (\mathbf{0}, 1)^{\mathrm{T}} & (K < L(S)) \end{cases} . \qquad (2)$$

This definition denotes that summaries that violate the length limitation are shrunk to a single feature, $(\mathbf{0}, 1)^{\mathrm{T}}$, which means it is not a summary.

Note the features of the state only depend on the features of the summary, not on the executed actions to achieve the state. Unlike naive search methods, this property has the potential for different states to be represented as the same vector, which has the same features. The agent, however, should search as many possible states as it can. Therefore, the generalization function of the feature representation is of utmost importance. The accurate selection of features contributes to reducing the search space and provides efficient learning as will be discussed later.

4.3 Action

An action denotes a transition operation that produces a new state from a current state. We assumed all actions were deterministic in this study. We define $\mathbf{insert}_i (1 \leq i \leq n)$ actions, each of which inserts textual unit x_i to the current state unless the state is terminated, as described in the following di-

agram:

$$\begin{array}{ccc} s_t & a_t & s_{t+1} \\ \left(\begin{array}{c} S_t \\ A_t \\ 0 \end{array}\right) & \xrightarrow{\mathbf{insert}_i} & \left(\begin{array}{c} S_t \cup \{x_i\} \\ A_t \cup \{\mathbf{insert}_i\} \\ 0 \end{array}\right) \end{array}. \quad (3)$$

In addition to insertion actions, we define **finish** that terminates the current episode in reinforcement learning:

$$\begin{array}{ccc} s_t & a_t & s_{t+1} \\ \left(\begin{array}{c} S_t \\ A_t \\ 0 \end{array}\right) & \xrightarrow{\mathbf{finish}} & \left(\begin{array}{c} S_t \\ A_t \cup \{\mathbf{finish}\} \\ 1 \end{array}\right) \end{array} \quad (4)$$

Note that $f_t = 1$ means state s_t is a terminal state.

Then, the whole action set, $\mathcal{A}$, is defined by $\mathbf{insert}_i$ and **finish**:

$$\mathcal{A} = \{\mathbf{insert}_1, \mathbf{insert}_2, \cdots, \mathbf{insert}_n, \mathbf{finish}\}. \quad (5)$$

We can calculate the available actions limited by state s_t:

$$\mathcal{A}(s_t) = \begin{cases} \mathcal{A} \backslash A_t & (L(S_t) \leq K) \\ \{\mathbf{finish}\} & (K < L(S_t)) \end{cases}. \quad (6)$$

This definition means that the agent may execute one of the actions that have not yet been executed in this episode, and it has no choice but to finish if the summary of the current state already violates length limitations.

4.4 Reward

The agent receives a reward from the environment as some kind of criterion of how good the action the agent executed was. If the current state is s_t, the agent executes a_t, and the state makes a transition into s_{t+1}; then, the agent receives the reward, r_{t+1}:

$$r_{t+1} = \begin{cases} score(S_t) & (a_t = \mathbf{finish}, L(S_t) \leq K) \\ -R_{penalty} & (a_t = \mathbf{finish}, K < L(S_t)) \\ 0 & (\text{otherwise}) \end{cases}, \quad (7)$$

where $R_{penalty} > 0$.

The agent can receive the score awarded by the given score function if and only if the executed action is **finish** and the summary length is appropriate. If the summary length is inappropriate but the executed action is **finish**, the environment awards a penalty to the agent. The most important point of this definition is that the agent receives nothing under the condition where the next state is not terminated. In this sense, the reward is delayed. Due to this definition, maximizing the expectation of future rewards is equivalent to maximizing the given score function, and we do not need to consider the decomposed form of the score function, i.e., we only need to consider the final score of the whole summary.

4.5 Value Function Approximation

Our aim is to find the optimal policy. This is achieved by obtaining the optimal *state value function*, $V^*(s)$, because if we obtain this, the greedy policy is optimal, which determines the action so as to maximize the state value after the transition occurred. Therefore, our aim is equivalent to finding $V^*(s)$. Let us try to estimate the *state value function* with parameter $\theta \in \mathbb{R}^d$:

$$V(s) = \theta^{\mathrm{T}} \phi(s). \quad (8)$$

We can also represent and estimate the *action value function*, $Q(s, a)$, by using $V(s)$:

$$Q(s, a) = r + \gamma V(s'), \quad (9)$$

where the execution of a causes the state transition from s to s' and the agent receives reward r, and $\gamma (0 \leq \gamma \leq 1)$ is the discount rate. Note that all actions are deterministic in this study.

By using these value functions, we define the policy as the conditional distribution, $p(a|s; \theta, \tau)$, which is parameterized by θ and a temperature parameter τ:

$$p(a|s; \theta, \tau) = \frac{e^{Q(s,a)/\tau}}{\sum_{a'} e^{Q(s,a')/\tau}}. \quad (10)$$

Temperature τ decreases as learning progresses, which causes the policy to be greedier. This softmax selection strategy is called Boltzmann selection.

4.6 Learning Algorithm

The goal of learning is to estimate θ. We use the TD (λ) algorithm with function approximation (Sutton and Barto, 1998). Algorithm 1 represents the whole system of our method, called Automatic Summarization using Reinforcement Learning (ASRL)

Algorithm 1 ASRL

Input: document $D = \{x_1, x_2, \cdots, x_n\}$,
score function $score(S)$

1: initialize $\theta = \mathbf{0}$
2: **for** $k = 1$ **to** N **do**
3: $s \leftarrow (\emptyset, \emptyset, 0)$
// initial state
4: $e = \mathbf{0}$
5: **while** s is not terminated **do**
6: $a \sim p(a|s; \theta, \tau_k)$
// selects action with current policy
7: $(s', r) \leftarrow \mathrm{execute}(s, a)$
// observes next state and receive reward
8: $\delta \leftarrow r + \gamma\theta^{\mathrm{T}}\phi(s') - \theta^{\mathrm{T}}\phi(s)$
// calculates TD-error
9: $e \leftarrow \gamma\lambda e + \phi(s)$
// updates the eligibility trace
10: $\theta \leftarrow \theta + \alpha_k \delta e$
// learning with current learning rate
11: $s \leftarrow s'$
12: **end while**
13: **end for**
14: $s \leftarrow (\emptyset, \emptyset, 0)$
15: **while** s is not terminated **do**
16: $a \leftarrow \max_a Q(s, a)$
// selects action greedily
with the learned policy
17: $(s', r) \leftarrow \mathrm{execute}(s, a)$
18: $s \leftarrow s'$
19: **end while**
20: **return** the summary of s

in this paper. N is the number of learning episodes, and $e (\in \mathbb{R}^d)$ and $\lambda (0 \leq \lambda \leq 1)$ correspond to the eligibility trace and the trace decay parameter. The eligibility trace, e, conveys all information on the features of states that the agent previously experienced, with previously decaying influences of features due to decay parameter λ and discount rate γ (Line 9).

Line 1 initializes parameter θ to start up its learning. The following procedures from Lines 2 to 13 learn θ with the TD (λ) algorithm, by using information on actual interactions with the environment. Learning rate α_k and temperature parameter τ_k decay as the learning episode progresses. The best summary with the obtained policy is calculated in steps from Lines 14 to 19. If the agent can estimate θ properly, greedy output is the optimal solution.

5 Models of Combined Approach for Reinforcement Learning

We formulated the extractive approach as a problem with reinforcement learning in the previous section. In fact, we can also formulate a more general model of summarization, since evaluation only depends on the final state and it is not actually very important to regard the given documents as a set of textual units contained in the original documents.

We explain how to take into account other methods within the ASRL framework by modifying the models in this section, with an example of sentence compression. We assume that we have a method of sentence compression, $comp(x)$, and that a textual unit to be extracted is a sentence. What we have to do is to only simply modify the definitions of the state and action. Note that this is just one example of the combined method. Even other summarization systems can be similarly adapted to ASRL.

5.1 State

We do not want to execute sentence compression twice, so we have to modify the state variables to convey the information: $s = (S, A, c, f)$, where $c \in \{0, 1\}$, and S, A, and f are the same definitions as previously described.

5.2 Action

We add deterministic action $\mathbf{comp}$ to $\mathcal{A}$, which produces the new summary constructed by compressing the last inserted sentence of the current summary:

$$\begin{array}{ccc} s_t & a_t & s_{t+1} \\ \begin{pmatrix} S_t \\ A_t \\ 0 \\ 0 \end{pmatrix} & \xrightarrow{\mathbf{comp}} & \begin{pmatrix} S_t \backslash \{x_c\} \cup \{comp(x_c)\} \\ A_t \cup \{\mathbf{comp}\} \\ 1 \\ 0 \end{pmatrix} \end{array}, \quad (11)$$

where x_c is the last sentence that is inserted into S_t. Next, we modify $\mathbf{insert}_i$ and $\mathbf{finish}$:

$$\begin{array}{ccc} s_t & a_t & s_{t+1} \\ \begin{pmatrix} S_t \\ A_t \\ c_t \\ 0 \end{pmatrix} & \xrightarrow{\mathbf{insert}_i} & \begin{pmatrix} S_t \cup \{x_i\} \\ A_t \cup \{\mathbf{insert}_i\} \\ 0 \\ 0 \end{pmatrix} \end{array}, \quad (12)$$

$$\begin{array}{ccc} s_t & a_t & s_{t+1} \\ \begin{pmatrix} S_t \\ A_t \\ c_t \\ 0 \end{pmatrix} & \xrightarrow{\textbf{finish}} & \begin{pmatrix} S_t \\ A_t \cup \{\textbf{finish}\} \\ c_t \\ 1 \end{pmatrix} \end{array}. \quad (13)$$

Note $\textbf{comp} \in A(s_t)$ may be executed if and only if $c_t = 0$. $\textbf{insert}_i$ resets c to 0.

6 Experiments

We conducted three experiments in this study. First, we evaluated our method with ROUGE metrics (Lin, 2004), in terms of ROUGE-1, ROUGE-2, and ROUGE-L. Second, we conducted an experiment on measuring the optimization capabilities of ASRL, with the scores we obtained and the execution time. Third, we evaluated ASRL taking into consideration sentence compression by using a very naive method, in terms of ROUGE-1, ROUGE-2, and ROUGE-3.

6.1 Experimental Settings

We used sentences as textual units for the extractive approach in this research. Each sentence and document were represented as a bag-of-words vector with *tf*idf* values, with stopwords removed. All tokens were stemmed by using Porter's stemmer (Porter, 1980).

We experimented with our proposed method on the dataset of DUC2004 task2. This is a multi-document summarization task that contains 50 document clusters, each of which has 10 documents. We set up the length limitation to 665 bytes, used in the evaluation of DUC2004.

We set up the parameters of ASRL where the number of episodes $N = 300$, the training rate $\alpha_k = 0.001 \cdot 101/(100 + k^{1.1})$, and the temperature $\tau_k = 1.0 \cdot 0.987^{k-1}$ where k was the number of episodes that decayed as learning progressed. Both discount rate γ and trace decay parameter λ were fixed to 1 for episodic tasks. The penalty, $R_{penalty}$, was fixed to 1.

We used the following score function in this study:

$$score(S) = \sum_{x_i \in S} \lambda_s Rel(x_i) - \sum_{x_i, x_j \in S, i<j} (1 - \lambda_s) Red(x_i, x_j), \quad (14)$$

where

$$Rel(x_i) = Sim(x_i, D) + Pos(x_i)^{-1} \quad (15)$$

$$Red(x_i, x_j) = Sim(x_i, x_j). \quad (16)$$

λ_s is the parameter for the trade-off between relevance and redundancy, $Sim(x_i, D)$ and $Sim(x_i, x_j)$ correspond to the cosine similarities between sentence x_i and the sentence set of the given original documents D, and between sentence x_i and sentence x_j. $Pos(x_i)$ is the position of the occurrence of x_i when we index sentences in each document from top to bottom with one origin. This score function was determined by reference to McDonald (2007). We set $\lambda_s = 0.9$ in this experiment.

We designed $\phi'(S)$, i.e., the vector representation of a summary, to adapt it to the summarization problem as follows.

- **Coverage of important words**: The elements are the top 100 words in terms of the *tf*idf* of the given document with binary representation.
- **Coverage ratio**: This is calculated by counting up the number of top 100 elements included in the summary.
- **Redundancy ratio**: This is calculated by counting up the number of elements that excessively cover the top 100 elements.
- **Length ratio**: This is the ratio between the length of the summary and length limitation K.
- **Position**: This feature takes into consideration the position of sentence occurrences. It is calculated with $\sum_{x \in S} Pos(x)^{-1}$.

Consequently, $\phi'(S)$ is a 104-dimensional vector.

We executed ASRL 10 times with the settings previously described and used all the results for evaluation.

We used the dataset of DUC2003, which is a similar task that contains 30 document clusters and each cluster had 10 documents, to determine τ_k and λ_s. We determined the parameters so that they would converge properly and become close to the optimal solutions calculated by ILP, under the conditions that the described feature representation and the score function were given.

	ROUGE-1	ROUGE-2	ROUGE-L
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: Results of ROUGE evaluation compared with other peers in DUC2004. Scores for ILP and GREEDY have statistically significant differences from scores of ASRL.

6.2 Evaluation

We compared ASRL with four other conventional methods.

- **GREEDY**: This method is a simple greedy algorithm, which repeats the selection of the sentence with the highest score of the remaining sentences by using an MMR-like method of scoring as follows:

$$x = \arg\max_{x \in D \setminus S} [\lambda_s Rel(x) - (1-\lambda_s) \max_{x_i \in S} Red(x, x_i)], \quad (17)$$

  where S is the current summary.

- **ILP**: This indicates the method proposed by McDonald (2007) for maximizing the score function (14) with integer linear programming.

- **PEER65**: This is the best performing system in task 2 of the DUC2004 competition in terms of ROUGE-1 proposed by Conroy et al. (2004).

- **MCKP**: This method was proposed by Takamura and Okamura (2009). MCKP defines an automatic summarization problem as a maximum coverage problem with a knapsack constraint, which uses *conceptual units* (Filatova and Hatzivassiloglou, 2004), and composes the meaning of sentences, as textual units and attempts to cover as many units as possible under the knapsack constraint.

7 Results

7.1 Evaluation with ROUGE

We evaluated our method of ASRL with ROUGE, in terms of ROUGE-1, ROUGE-2, and ROUGE-L.

	ROUGE-1	ROUGE-2	ROUGE-L
ASRL.0	0.39274	0.09537	0.34010
ASRL.1	0.39243	0.09683	0.33855
ASRL.2	0.39241	0.09597	0.34070
ASRL.3	0.39190	0.09580	0.33898
ASRL.4	0.39054	0.09579	0.33663
ASRL.5	0.38911	0.09395	0.33551
ASRL.6	0.38866	0.09392	0.33701
ASRL.7	0.38854	0.09338	0.33661
ASRL.8	0.38821	0.09363	0.33833
ASRL.9	0.38532	0.09281	0.33321

Table 2: Results of ROGUE evaluation for each ASRL peer of 10 results in DUC2004. ASRL did not converge with stable solution with these experimental settings because of property of randomness.

The experimental results are summarized in Tables 1 and 2. Table 1 lists the results for the comparison and Table 2 lists all the results for ASRL peers.

The results imply ASRL is superior to PEER65, ILP, and GREEDY, and comparable to MCKP with these experimental settings in terms of ROUGE metrics. Note that ASRL is a kind of approximate method, because actions are selected probabilistically and the method of reinforcement learning occasionally converges with some sub-optimal solution. This can be expected from Table 2, which indicates the results vary although each ASRL solution converged with some solution. However, in this experiment, ASRL achieved higher ROUGE scores than ILP, which achieved optimal solutions. This seems to have been caused by the properties of the features, which we will discuss later. It seems this feature representation is useful for efficiently searching the feature space. The method of mapping a state to features is, however, approximate in the sense that some states will shrink to the same feature vector, and ASRL therefore has no tendency to converge with some stable solution.

7.2 Evaluation of Optimization Capabilities

Since we proposed our method as an approach to approximate optimization, there was the possibility of convergence with some sub-optimal solution as previously discussed. We also evaluated our approach from the point of view of the obtained scores and the execution time to confirm whether our method had

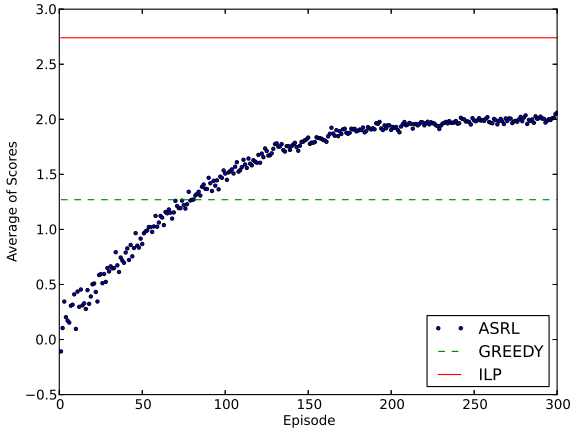

Figure 1: Average score for each episode in ASRL in DUC2004. Horizontal lines indicate scores of summaries obtained with ILP and GREEDY.

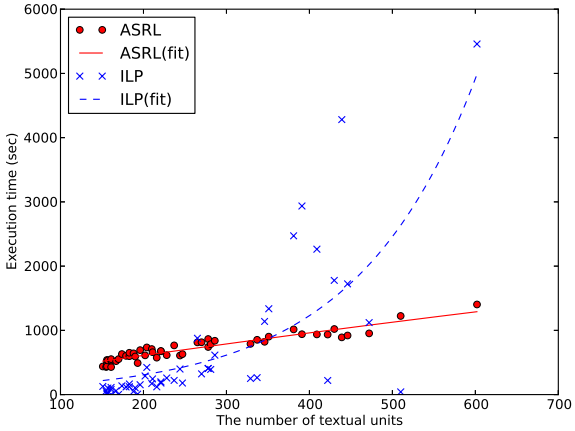

Figure 2: Execution time on number of textual units for each problem in DUC2004. Plot of ASRL is fitted to linear function and that of ILP is fitted to exponential function.

optimization capabilities.

The experimental results are plotted in Figures 1 and 2. Figure 1 plots the average for the rewards (i.e., scores) that the agent obtained for each episode. The horizontal line for ILP is the average for the optimal scores of (14). The score in ASRL increases as the number of episodes increases, and overtakes the score of GREEDY at some episode. The agent attempts to come close to the optimal score line of ILP but seems to fail, and finally converges to some local optimal solution. We should increase the number of episodes, adjust parameters α and τ, and select more appropriate features for the state to improve the optimization capabilities of ASRL.

Figure 2 plots the execution time for each peer. The horizontal axis is the number of textual units, i.e., the number of sentences in this experiment. The vertical axis is the execution time taken by the task. The plots of ASRL and ILP fit a linear function for the former and an exponential function for the latter. The experimental results indicate that while the execution time for ILP tends to increase exponentially, that for ASRL increases linearly. The time complexity of ASRL is linear with respect to the number of actions because the agent has to select the next action from the available actions for each episode, whose time complexity is naively $O(|\mathcal{A}|)$. As $\mathbf{insert}_i$ actions are dominant in the extractive approach, the execution time increases linearly with respect to the number of textual units. However, ILP has to take into account the combinations of textual units, whose number increases exponentially.

In conclusion, both the experimental results indicate that ASRL efficiently calculated a summary that was sub-optimal, but that was of relatively high-quality in terms of ROUGE metrics, with the experimental settings we used.

7.3 Evaluation of Effects of Sentence Compression

We also evaluated the combined approach with sentence compression. We evaluated the method described in Section 5 called ASRLC in this experiment for the sake of convenience. We used a very naive method of sentence compression for this experiment, which compressed a sentence to only important words, i.e., selecting word order by using the *tf*idf* score to compress the length to about half. This method of compression did not take into consideration either readability or linguistic quality. Note we wanted to confirm what effect the other methods would have, and we expected this to improve the ROUGE-1 score. We used the ROUGE-3 score in this evaluation instead of ROUGE-L, to confirm whether naive sentence compression occurred.

The experimental results are summarized in Ta-

	ROUGE-1	ROUGE-2	ROUGE-3
ASRL	0.39013	0.09479	0.03435
ASRLC	0.39141	0.09259	0.03239

Table 3: Evaluation of combined methods.

ble 3, which indicates ROUGE-1 increases but ROUGE-2 and ROUGE-3 decrease as expected. The variations, however, are small. This phenomenon was reported by Lin (2003) in that the effectiveness of sentence compression by local optimization at the sentence level was insufficient. Therefore, we would have to consider the range of applications with the combined method.

8 Discussion

8.1 Local Optimality of ASRL

We will discuss why ASRL seems to converge with some "good" local optimum with the described experimental settings in this section.

Since our model of the state value function was simply linear and our parameter estimation was implemented by TD (λ), which is a simple method in RL, it seems simply employing more efficient or state-of-the-art reinforcement learning methods may improve the performance of ASRL, such as GTD and GTD2 (Sutton et al., 2009b; Sutton et al., 2009a). These methods basically only contribute to faster convergence, and the score that they will converge to might not differ significantly. As a result, it would not matter much which method was used for optimization.

The main point of this problem is modeling the feature representation of states, and this causes sub-optimality. The vector representation of states shrinks the different states to a single representation, i.e., the agent regards states whose features are similar to be similar states. Due to this property, the policy of reinforcement learning is learned to maximize *the expected score* of each feature vector, which includes many states. Such sub-optimality *averagely balanced* by the feature representation raises the possibility of achieving states that have a high-quality summary with a low score, since we do not have a genuine score function.

Thus, the most important thing in our method is to intentionally design the features of states and the score function, so that the agent can generalize states, while taking into consideration truly-essential features for the required summarization. It would be useful if the forms of features and the score function could be arbitrarily designed by the user because there is the capability of obtaining a high-quality summaries.

8.2 Potential of Combined Method

Other useful methods, even other summarization systems, can easily be adapted to ASRL as was described in Section 5. The experimental results revealed that sentence compression has some effect. In fact, all operations that produce a new summary from an old summary can be used, i.e., even other summarizing methods can be employed for an action. We assumed a general combined method may have a great deal of potential to enhance the quality of summaries.

8.3 Can We Obtain "a Global Policy"?

We formulated each summarization task as a reinforcement learning task in this paper, i.e., where each learned policy differs. As this may be a little unnatural, we wanted to obtain a single learned policy, i.e., a global policy.

However, we assessed that we cannot achieve a global policy with these feature and score function settings because the best vector, which is the feature representation of the summary that achieves an optimal score under the current settings, seems to vary for each cluster, even if the domain of the clusters is the same (e.g., a news domain). Having said that, we simultaneously surmised that we could obtain a global policy if we could obtain a highly general, crucial, and efficient feature representation of a summary. We also think a global policy is essential in terms of reinforcement learning and we intend to attempt to achieve this in future work.

9 Conclusion

We presented a new approach to the problem of automatic text summarization called ASRL in this paper, which models the process of constructing a summary with the framework of reinforcement learning and attempts to optimize the given score function with the given feature representation.

The experimental results demonstrated ASRL tends to converge sub-optimally, and excessively depends on the formulation of features and the score function. Although it is difficult, we believe this formulation would enable us to improve the quality of summaries by designing them freely.

We intend to employ the ROUGE score as the score function in future work, and obtain the parameters of the state value function. Using these results, we will attempt to obtain a single learned policy by employing the ROUGE score or human evaluations as rewards. We also intend to consider efficient features and a score to achieve stable convergence. In addition, we plan to use other methods of function approximation, such as RBF networks.

References

J.M. Conroy, J.D. Schlesinger, J. Goldstein, and D.P. O ' leary. 2004. Left-brain/right-brain multi-document summarization. In *Proceedings of the Document Understanding Conference (DUC 2004)*.

E. Filatova and V. Hatzivassiloglou. 2004. A formal model for information selection in multi-sentence text extraction. In *Proceedings of the 20th international conference on Computational Linguistics*, page 397. Association for Computational Linguistics.

J. Goldstein, V. Mittal, J. Carbonell, and M. Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization-Volume 4*, pages 40–48. Association for Computational Linguistics.

C.Y. Lin. 2003. Improving summarization performance by sentence compression: a pilot study. In *Proceedings of the sixth international workshop on Information retrieval with Asian languages-Volume 11*, pages 1–8. Association for Computational Linguistics.

C.Y. Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the workshop on text summarization branches out (WAS 2004)*, volume 16.

I. Mani. 2001. *Automatic summarization*, volume 3. John Benjamins Pub Co.

R. McDonald. 2007. A study of global inference algorithms in multi-document summarization. *Advances in Information Retrieval*, pages 557–564.

MF Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

R.S. Sutton and A.G. Barto. 1998. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press.

R.S. Sutton, H.R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. 2009a. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 993–1000. ACM.

R.S. Sutton, C. Szepesvári, and H.R. Maei. 2009b. A convergent o (n) algorithm for off-policy temporal-difference learning with linear function approximation.

H. Takamura and M. Okumura. 2009. Text summarization model based on maximum coverage problem and its variant. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 781–789. Association for Computational Linguistics.