Reward Learning for Efficient Reinforcement Learning in Extractive Document Summarisation

Yang Gao[1*], **Christian M. Meyer**[2], **Mohsen Mesgar**[2] and **Iryna Gurevych**[2]
[1]Dept. of Computer Science, Royal Holloway, University of London
[2] Ubiquitous Knowledge Processing Lab (UKP-TUDA), Technische Universität Darmstadt
gaoyang_alex@hotmail.com, https://www.ukp.tu-darmstadt.de

Abstract

Document summarisation can be formulated as a sequential decision-making problem, which can be solved by *Reinforcement Learning (RL)* algorithms. The predominant RL paradigm for summarisation learns a *cross-input* policy, which requires considerable time, data and parameter tuning due to the huge search spaces and the delayed rewards. Learning *input-specific* RL policies is a more efficient alternative but so far depends on handcrafted rewards, which are difficult to design and yield poor performance. We propose RELIS, a novel RL paradigm that learns a reward function with *Learning-to-Rank (L2R)* algorithms at training time and uses this reward function to train an input-specific RL policy at test time. We prove that RELIS guarantees to generate near-optimal summaries with appropriate L2R and RL algorithms. Empirically, we evaluate our approach on extractive multi-document summarisation. We show that RELIS reduces the training time by two orders of magnitude compared to the state-of-the-art models while performing on par with them.

1 Introduction

Extractive document summarization, as a challenging instance of *natural language generation* (NLG), is a popular summarisation paradigm, which builds summaries by selecting an appropriate sequence of important phrases or sentences from the input document(s). Extractive summarisation can be formulated as a sequential decision-making problem, and hence can be tackled by the *Reinforcement Learning (RL)* algorithms. RL search (near-)optimal *trajectories* (i.e. sequences of decisions) by directly optimising the objective functions, e.g. the ROUGE metrics [Lin, 2004]. Such objectives are non-differentiable and therefore difficult to directly optimise by deep neural networks. In addition, RL alleviates the *exposure bias* problem faced by sequential supervised learning paradigms in NLG. Combining RL algorithms such as REINFORCE [Williams, 1992] with neural techniques (e.g. sequence-to-sequence) yields state-of-the-art performance in summarisation [Narayan *et al.*, 2018; Yao *et al.*, 2018].

Existing RL-based summarisation systems fall into two categories: *cross-input RL* and *input-specific* RL. For cross-input RL (upper part in Fig. 1), at training time, RL agents interact with a *ground-truth reward oracle* for multiple *episodes* so as to learn a *policy* that maximises the accumulated rewards in the episode; at test time, the learnt policy is applied to unseen data to generate the summary. However, learning such cross-input policies is very expensive because of the huge search space. Another issue is the *delayed rewards*: the ROUGE scores can be calculated only when the complete summary is generated. Such delayed rewards cause RL-based summarisers to take even longer time to converge. Although multiple techniques have been proposed to speed up the training of RL, e.g. memory replay [Mnih *et al.*, 2015], MIXER [Ranzato *et al.*, 2015] and imitation learning [Cheng *et al.*, 2018], training cross-input RL yet requires considerable time, data and parameter tuning.

On the other hand, input-specific RL (middle part in Fig. 1) neither requires parallel data (i.e. input documents and reference summaries for them) nor a reward oracle. Instead, it employs a handcrafted reward function with which the RL agent interacts in order to learn a policy specifically for the given input. By doing so, the size of the search space significantly decreases, which consequently diminishes the training time and computational resources. However, designing such a reward function is highly challenging as it should fit all inputs [Rioux *et al.*, 2014]. This explains the poor performance of input-specific RL summarisers [Ryang and Abekawa, 2012].

To tackle the problems faced by the above two RL-based summarisation paradigms, we propose a novel paradigm called *REward Learning for Input-Specific reinforcement learning (RELIS).* Instead of learning a cross-input policy, RELIS learns a *cross-input reward oracle* at training time, and then uses the learned reward to train an input-specific policy for each input at test time (bottom part in Fig. 1). RELIS is inspired by *inverse RL* [Abbeel and Ng, 2004], which requires a demonstrator to present optimal trajectories. Because such a demonstrator is hardly available in summarisation, RELIS leverages *Learning-to-Rank (L2R)* algorithms [Li, 2011] to approximate the ground-truth reward oracle from "weak supervisions", e.g., numeric scores that indicate the quality of the summary or preferences over summary pairs.

*Work performed while at UKP-TUDA.

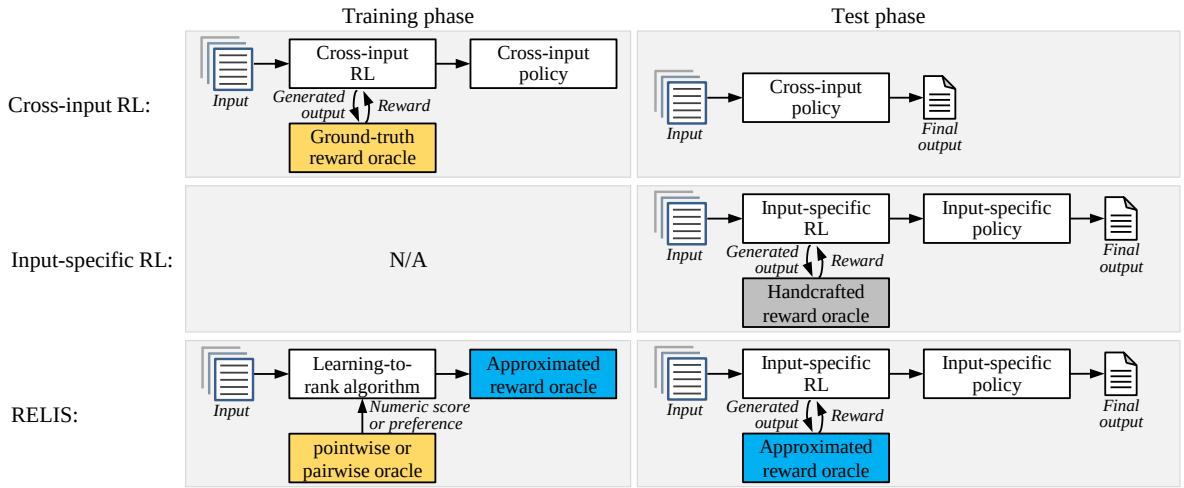

Figure 1: The workflows of cross-input RL (top), input-specific RL (middle) and RELIS (bottom). The ground-truth reward can by provided by humans or automatic metrics (e.g. BLEU or ROUGE) measuring the similarity between generated output text and the reference output.

Our contributions are threefold: **(i)** We propose a new RL-based summarisation framework (§2), which enjoys the strong performance of cross-input RL and the low computational cost of input-specific RL. **(ii)** Theoretically, we prove that by employing appropriate L2R and RL algorithms, RELIS guarantees to generate near-optimal outputs (§3). **(iii)** Empirically, we evaluate RELIS in multi-document extractive summarisation (§4). Compared to the state-of-the-art methods, RELIS provides comparable performance but requires much less the training time or data. Because the proof of RELIS is generic, we believe RELIS has the potential to be applied to other NLG tasks, e.g. translation and sentence simplification. Source code and supplementary material are available at https://github.com/UKPLab/ijcai2019-relis.

2 RELIS

We first formally define the summarisaiton task, and then detail the L2R module and RL module of RELIS.

Extractive summarisation. In line with Peyrard and Eckle-Kohler [2017], we formulate summarisation as a *discrete optimisation problem*. Let $\mathcal{X}$ be the set of all possible input documents and $\overline{\mathcal{X}}$ be the set of inputs available at training time. An input can be either a single document or a cluster of documents on the same topic. For input $x \in \mathcal{X}$, let $\mathcal{Y}_x$ indicate the set of all extractive summaries for x that comply with the length requirement. Then, the task of summarisation is to map each input x to its best summary in $\mathcal{Y}_x$ with respect to a *ranking function* $\sigma_x \colon \mathcal{Y}_x \to \mathbb{N}$. For a candidate summary $y \in \mathcal{Y}_x$, σ_x returns the number of candidates in $\mathcal{Y}_x$ that have equal or lower quality than y, including y itself. For example, if y is the highest-quality candidate in $\mathcal{Y}_x$, then $\sigma_x(y) = |\mathcal{Y}_x|$. σ_x can be obtained from human evaluators, automatic metrics (e.g. ROUGE) or heuristics measuring the quality of outputs. We denote the ground-truth ranking function on $\mathcal{Y}_x$ by σ_x^*.

Given the above definition of summarisation, we define a summariser agent as a tuple (σ_x, M_x), where M_x is an *optimisation model* for finding the (near-)optimal summary with respect to σ_x. In the cross-input paradigm, the RL agent learns a policy that solves the optimisation problem M_x for any $x \in \mathcal{X}$ at training time. In RELIS, instead, an agent learns a ranking $\hat{\sigma}_x^U \colon \mathcal{Y}_x \to \mathbb{N}$ at training time so that $\hat{\sigma}_x^U$ is as "close" as possible to σ_x^* ("close" will be formally defined in §3). At test time, for each $x' \in \mathcal{X} \setminus \overline{\mathcal{X}}$, RELIS formulates $M_{x'}$ as a *Markov Decision Process (MDP)* and learns an RL policy specifically for $M_{x'}$.

Learning to rank (L2R). L2R algorithms learn to reconstruct the ranking over objects from an *oracle* [Li, 2011]. L2R induces the approximated ranking $\hat{\sigma}_x^U$ by learning a *utility function* U from the oracle, such that $\hat{\sigma}_x^U(y_1) > \hat{\sigma}_x^U(y_2)$ if and only if $U(y_1, x) > U(y_2, x)$. L2R can learn from different types of oracles, including *pointwise* oracles that provide point-based scores for objects, *pairwise* oracles that provide preferences over pairs of objects, and *listwise* oracles that provide certain metrics (e.g. accuracy) for candidate rankings. Here, we focus on pointwise and pairwise oracles as humans reliably provide such judgements for short texts [Kreutzer *et al.*, 2018]. Function U can be learnt by any function approximation technique, e.g., neural networks.

In pointwise L2R, for every $x \in \overline{\mathcal{X}}$, we draw N sample summaries from $\mathcal{Y}_x$ using some sampling strategy, e.g., random sampling without replacement. Then, we query their σ_x^* values from the oracle and use a regression algorithm to minimise the averaged mean squared error (MSE) between U and σ_x^*. Formally, the loss function is

$$\mathcal{L}^{\text{MSE}} = \frac{1}{N \cdot |\overline{\mathcal{X}}|} \sum_{x \in \overline{\mathcal{X}}} \sum_{i=1}^{N} (\sigma_x^*(y_i) - U(y_i, x))^2. \quad (1)$$

In pairwise L2R, for every $x \in \overline{\mathcal{X}}$ we sample K pairs of summaries from $\mathcal{Y}_x$ and then query the oracle about their preferences. We denote the collected preferences for input x by $P_x = \{p_1(y_{1,1}, y_{1,2}), \cdots, p_K(y_{K,1}, y_{K,2})\}$, where $y_{i,1}, y_{i,2} \in \mathcal{Y}_x$ are the candidate outputs presented to the oracle in the ith iteration; $p_i(y_{i,1}, y_{i,2})$ equals 1 if the oracle

prefers $y_{i,1}$ over $y_{i,2}$, and equals 0 otherwise. Different loss functions can be used to learn U from the preferences in P_x. First, we consider the cross-entropy loss:

$$\mathcal{L}^{\text{CE}} = -\sum_{x\in\overline{\mathcal{X}}} \sum_{p_i(y_{i,1},y_{i,2})\in P_x} [\, p_i(y_{i,1},y_{i,2}) \log \mathcal{P}(y_{i,1},y_{i,2}) + p_i(y_{i,2},y_{i,1}) \log \mathcal{P}(y_{i,2},y_{i,1})\,], \quad (2)$$

where $\mathcal{P}(y_1,y_2) = (1+\exp[U(y_2,x)-U(y_1,x)])^{-1}$. An alternative is the margin ranking (a.k.a. pairwise hinge) loss:

$$\mathcal{L}^{\text{MR}} = \frac{1}{N\cdot|\overline{\mathcal{X}}|} \sum_{x\in\overline{\mathcal{X}}} \sum_{p_i(y_{i,1},y_{i,2})\in P_x} \max[0, 1 - e_i\cdot(U(y_{i,1},x)-U(y_{i,2},x))], \quad (3)$$

where $e_i = 1$ if $y_{i,1}$ is preferred over $y_{i,2}$, and $e_i = -1$ otherwise. Additionally, we consider an improved margin ranking loss proposed by Agarwal and Collins [2010], which gives large penalties to mis-ranked pairs with wide margins:

$$\mathcal{L}^{\text{IM}} = \frac{1}{N\cdot|\overline{\mathcal{X}}|} \sum_{x\in\overline{\mathcal{X}}} \sum_{p_i(y_{i,1},y_{i,2})\in P_x} \max[0, |\sigma^*_x(y_{i,1})-\sigma^*_x(y_{i,2})| - e_i(U(y_{i,1},x)-U(y_{i,2},x))]. \quad (4)$$

Note that the improved margin ranking loss is a mix of pointwise and pairwise L2R as it requires both σ^*_x and P_x.

Reinforcement learning (RL). RL amounts to algorithms for searching optimal solutions in *Markov Decision Processes (MDPs)*. We formulate the optimisation problems M_x for NLG tasks as episodic MDP [Ryang and Abekawa, 2012; Rioux *et al.*, 2014]. An episodic MDP is a tuple (S, A, P, R, T), where S is a set of *states*, A is a set of *actions*, $P\colon S\times A\to S$ is the *transition function* with $P(s,a)$ giving the next state after performing action a in state s, $R\colon S\times A\to\mathbb{R}$ is the *reward function* with $R(s,a)$ giving the immediate reward for performing action a in state s and $T\subseteq S$ is a set of *terminal states* that mark the end of an episode. Furthermore, to formulate the delayed rewards in summarisation, we let $R(s,a)=0$ if $P(s,a)\notin T$, so as to ensure that non-zero rewards only appear in the last step of each episode.

In extractive summarisation, the components in M_x are defined as follows. S is the set of all possible states, i.e. permutations of sentences from x. Note that $\mathcal{Y}_x$ is a subset of S because $\mathcal{Y}_x$ only includes the summaries complying with the length requirement, but S includes all possible extractive summaries. Two types of action constitute A: *add* a new sentence from the input document cluster x to the current draft summary and *terminate* the generation. T includes all the over-length summaries and an *absorbing state* s_T. If action a is terminate, $P(s,a)=s_T$ regardless of the current state s. The reward function $R(s,a)$ returns $\sigma_x(s)$ if action a is terminate, a negative reward if the current state s is an over-length summary and 0 otherwise. We denote this MDP by $M_x(\sigma_x)$ to highlight that it uses σ_x as rewards.

A *policy* $\pi\colon S\times A\to[0,1]$ defines how actions are selected: $\pi(s,a)$ is the probability of selecting action a in state s. In the context of summarisation, we slightly abuse the notation by letting $\pi(y|x)$ denote the probability of policy π generating $y\in\mathcal{Y}_x$ given input x. We say a policy π is *proper* if $\sum_{y\in\mathcal{Y}_x}\pi(y|x)=1$, i.e. π does not generate illegal outputs. Then, the expected reward of performing proper policy π is:

$$\mathcal{R}_{\sigma_x}(\pi|x) = \mathbb{E}_{y\sim\pi}\sigma_x(y) = \sum_{y\in\mathcal{Y}_x}\pi(y|x)\sigma_x(y). \quad (5)$$

The goal of RL is to find the optimal policy π^* that has the highest expected reward: $\pi^* = \arg\max_\pi \mathcal{R}_{\sigma_x}(\pi|x)$. Note that π^*_x is a probability distribution that assigns non-zero probabilities only to the optimal outputs for x, i.e. $\pi^*(y|x)>0$ only if $\sigma_x(y)=|\mathcal{Y}_x|$. Hence $\mathcal{R}_{\sigma_x}(\pi^*|x)=|\mathcal{Y}_x|$.

3 Proof of Convergence for RELIS

In this section, we prove that if the errors in both L2R and RL are bounded, the error of the final output of RELIS is a linear polynomial of the errors from the two steps. First, we define *convergent* L2R, which can learn an approximated ranking $\hat{\sigma}^U_x$ close to the ground-truth ranking σ^*_x.

Definition 1. *Let U be the utility function learnt by an L2R algorithm. For $x\in\mathcal{X}$, let $\hat{\sigma}^U_x$ be the ranking derived from U on $\mathcal{Y}_x$. The L2R algorithm is* ϵ-convergent *with respect to σ^*_x if there exists a non-negative integer ϵ such that*

$$\max_{y\in\mathcal{Y}_x}|\hat{\sigma}^U_x(y)-\sigma^*_x(y)|\le\epsilon. \quad (6)$$

Some L2R algorithms are ϵ-convergent under realistic conditions. For example, the quick-sort based L2R algorithm proposed by Maystre and Grossglauser [2017] is ϵ-convergent with respect to pairwise oracles, when it obtains a sufficiently large number of preferences. Next, we define convergent RL algorithms that guarantee to learn near-optimal policies.

Definition 2. *Let $M_x(\sigma_x)$ denote the episodic MDP for an NLG task given input $x\in\mathcal{X}$, which uses a ranking function σ_x over $\mathcal{Y}_x$ as its reward function. An RL algorithm is* δ-convergent *for $M_x(\sigma_x)$ if there exists $\delta\in\mathbb{R}_{\ge0}$ such that*

$$\lim_{n\to\infty}\mathcal{R}_{\sigma_x}(\pi^*|x)-\mathcal{R}_{\sigma_x}(\pi_n|x)\le\delta, \quad (7)$$

where π^* is the optimal policy for $M_x(\sigma_x)$ and π_n is the policy of the RL algorithm after n episodes of learning.[1]

Many RL algorithms have been proven to be δ-convergent, including value-based RL [Sutton *et al.*, 2009], actor-critic [Sutton *et al.*, 1999] and policy gradient [Fazel *et al.*, 2018].

Recall that for an input x from the test set $\mathcal{X}\setminus\overline{\mathcal{X}}$, our goal is to find the optimal policy π^* for the NLG tuple $(\sigma^*_x, M_x(\sigma^*_x))$ (see §2). The theorem below shows that if RELIS uses convergent L2R and RL, its output is near-optimal, i.e. the error between the RELIS output and the optimal output is bounded.

Theorem 1. *For a test input $x\in\mathcal{X}\setminus\overline{\mathcal{X}}$, we denote the optimal NLG agent for x by $(\sigma^*_x, M_x(\sigma^*_x))$, and its RELIS approximation by $(\hat{\sigma}^U_x, M_x(\hat{\sigma}^U_x))$, where $\hat{\sigma}^U_x$ is the approximation of σ^*_x learnt by an ϵ-convergent L2R algorithm. Let π^* be the optimal policy for $M_x(\sigma^*_x)$ and $\hat{\pi}_n$ the proper policy for $M_x(\hat{\sigma}^U_x)$ learnt by a δ-convergent RL algorithm for n episodes. Then, as n approaches positive infinity, $\mathcal{R}_{\sigma^*_x}(\hat{\pi}_n|x)\ge\mathcal{R}_{\sigma^*_x}(\pi^*|x)-\delta-\epsilon$.*

[1] Since π^* is the optimal policy for σ_x, $\mathcal{R}_{\sigma_x}(\pi^*|x)$ is always higher than the expected reward of any other policy (cf. Eq. 5).

Proof. We denote the optimal policy for $M_x(\hat{\sigma}_x^U)$ as $\hat{\pi}^*$. Hence from Eq. (7) we have:

$$\lim_{n\to\infty} \mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}_n|x) \geq \mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}^*|x) - \delta. \tag{8}$$

For any $y \in \mathcal{Y}_x$, from Eq. (6) we have

$$\hat{\sigma}_x^U(y) \leq \sigma_x^*(y) + \epsilon. \tag{9}$$

Using Eq. (9), we can bound $\mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}_n|x)$ with any positive n value (see Eq. (5)):

$$\begin{aligned}\mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}_n|x) &= \sum_{x\in\mathcal{Y}_x} \hat{\pi}_n(y|x)\hat{\sigma}_x^U(y) \\ &\leq \sum_{x\in\mathcal{Y}_x} \hat{\pi}_n(y|x)[\sigma_x^*(y) + \epsilon] \\ &= \sum_{y\in\mathcal{Y}_x} \hat{\pi}_n(y|x)\sigma_x^*(y) + \epsilon \sum_{y\in\mathcal{Y}_x} \hat{\pi}_n(y|x) \\ &= \mathcal{R}_{\sigma_x^*}(\hat{\pi}_n|x) + \epsilon. \end{aligned} \tag{10}$$

Note that $\sum_{y\in\mathcal{Y}_x} \hat{\pi}_n(y|x) = 1$ in Eq. (10) because $\hat{\pi}_n$ is a proper policy (see §2). Combining Eq. (8) and (10), we get

$$\lim_{n\to\infty} \mathcal{R}_{\sigma_x^*}(\hat{\pi}_n|x) \geq \mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}^*|x) - \delta - \epsilon. \tag{11}$$

Since $\hat{\pi}^*$ is the optimal policy for $M_{\hat{\sigma}_x^U}$, according to Eq. (5), $\mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}^*|x) = |\mathcal{Y}_x|$. Similarly, $\mathcal{R}_{\sigma_x^*}(\pi^*|x) = |\mathcal{Y}_x|$. Hence we can replace $\mathcal{R}_{\hat{\sigma}_x^U}(\hat{\pi}^*|x)$ in Eq. (11) with $\mathcal{R}_{\sigma_x^*}(\pi^*|x)$ and obtain $\lim_{n\to\infty} \mathcal{R}_{\sigma_x^*}(\hat{\pi}_n|x) \geq \mathcal{R}_{\sigma_x^*}(\pi^*|x) - \delta - \epsilon$. □

4 Experimental Setup

Task and datasets. We evaluate RELIS for extractive multi-document summarisation on three commonly employed benchmark datasets from the Document Understanding Conferences (DUC)[2] described in Table 1. Each dataset contains a set of document clusters. For each cluster, the target is to create a summary with at most 100 words. Each cluster is accompanied with several human-generated summaries, denoted as r_x, that we use for training and evaluation. To decide the best parameters, we perform 10-fold cross validation on DUC'01.

Ground-truth reward oracle. For $x \in \overline{\mathcal{X}}$, we use a ROUGE-based U^* to induce the ground-truth σ_x^*. We measure the quality of a summary $y \in \mathcal{Y}_x$ by $U^*(y, x) = R_1(y, r_x) + 2R_2(y, r_x)$, where R_1 and R_2 denote the average ROUGE-1 and ROUGE-2 recall metrics, respectively. R_1 and R_2 are arguably the most widely used metrics to approximate human evaluation of summary quality. The R_2 scores for optimal extractive summaries are usually half of their R_1 scores [Gao *et al.*, 2018], so we multiply R_2 by 2 to balance the impact of R_1 and R_2.

Approximated reward oracle. Recall that the goal of L2R is to learn the utility function U using pointwise or pairwise preferences (see §2). In pointwise L2R, we sample $N = 3000$ different summaries for each document cluster in $x \in \overline{\mathcal{X}}$ by randomly selecting sentences from x until the length limit is reached. Using larger N does not significantly increase the quality of $\hat{\sigma}_x^U$ but increases the training time. We use the ground-truth ranking σ_x^* over the $|\overline{\mathcal{X}}| \cdot N$ samples to learn the utility function U by minimising the MSE loss from Eq. (1).

[2]https://duc.nist.gov/

Dataset	# Cluster	# Doc	# Sent/Cluster
DUC '01	30	308	378
DUC '02	59	567	271
DUC '04	50	500	265

Table 1: Some statistics of the DUC datasets: the number of document clusters (# Cluster), the total number of documents (# Doc) and the average number of sentences per cluster (# Sent/Cluster).

In pairwise L2R, we randomly draw $K = 10^5$ of the $|\overline{\mathcal{X}}|\frac{N\cdot(N-1)}{2}$ possible pairs of the $|\overline{\mathcal{X}}| \cdot N$ sampled summaries to compute U by minimising $\mathcal{L}^{\text{CE}}$, $\mathcal{L}^{\text{MR}}$ or $\mathcal{L}^{\text{IM}}$ from Eq. (2) – (4). Preliminary results suggest that increasing K to 10^6 significantly slows down the training, while decreasing it to 10^4 significantly harms the quality of $\hat{\sigma}_x^U$.

We use a linear model to approximate the utility function, i.e. $U(y, x) = w \cdot \phi(y, x)$, where $\phi(y, x)$ encodes y for input x as a vector by concatenating the following features:

- ***JS** The negative Jensen-Shannon divergence between unigrams, bigrams and named entities in y and x.
- **ASRL** The summary quality evaluation heuristics proposed by Ryang and Abekawa [2012].
- ***REAPER** The summary quality evaluation heuristics proposed by Rioux *et al.* [2014].
- ***TFIDF** values averaged over all tokens and named entities in y [Peyrard and Gurevych, 2018].
- ***CF** The average number of document clusters a word or named entity appears in.
- ***NE-Recall** Rate of named entities from x appearing in y.
- ***NE-Density** The percentage of tokens in y that belong to some named entities in x.
- **LSA** The topic-modelling based feature proposed by Peyrard and Eckle-Kohler [2017].
- ***Red** The redundancy feature proposed by Peyrard and Gurevych [2018], applied to unigrams, bigrams and named entities.
- **InferSent** Cosine similarity between the averaged InferSent [Conneau *et al.*, 2017] embeddings of y and x.
- **DSS** The distributional semantic similarity between y and x, proposed by ShafieiBavani *et al.* [2018].
- **TopicRelevance** proposed by ShafieiBavani *et al.* [2018]. We generate 10 topics for x, and use the top 30 words in each topic to construct the topic embedding.
- **Coherence** feature proposed by [ShafieiBavani *et al.*, 2018].

We additionally use two Convolutional Neural Network (CNN) architectures to generate auto-encoded features: StdCNN [Kim, 2014] and PriorSum [Cao *et al.*, 2015]. To train these auto-encoders, we follow Cao *et al.* [2015] and feed only the embeddings of the words of y into these models. We add a linear layer as the last layer to map the output vector of these models to a U value. Full settings of these models are in the supplementary material.

	DUC'01		DUC'02		DUC'04	
	ρ	NDCG	ρ	NDCG	ρ	NDCG
ASRL	.176	.555	.131	.537	.145	.558
REAPER	.316	.638	.301	.639	.372	.701
JS	.549	.736	.525	.700	.570	.763
Our $\hat{\sigma}_x^U$	**.601**	**.764**	**.560**	**.727**	**.617**	**.802**

Table 2: The correlation of approximated and ground-truth ranking. $\hat{\sigma}_x^U$ has significantly higher correlation over all other approaches.

Feature selection. For each $x \in \mathcal{X} \setminus \overline{\mathcal{X}}$, we measure the quality of the approximated ranking $\hat{\sigma}_x^U$ by its ranking correlation to the ground-truth σ_x^*. We consider two ranking correlation metrics: Spearman's $\rho \in [-1, 1]$ and the Normalized Discounted Cumulative Gain (NDCG) on the top 1% items. NDCG $\in [0, 1]$ puts more emphasis on the top elements with logarithmic decay weighting . For both metrics, higher values indicate stronger correlations .

We select features that yield higher ρ + NDCG, averaged over results for all 10 folds of DUC'01. We train the utility function U with the loss functions in Eq. (1)-(4) and we find that using different loss functions does not significantly[3] change performance (see supplementary material). However, the cross-entropy loss consistently results in marginally better performance than the others, and hence we use it throughout our experiments. Following standard feature selection methods, we omit features that are highly correlated with other features or have small weights in the linear model. The final features we used to represent ϕ are marked with an asterisk in the aforementioned feature list. We find that under all examined settings, the CNN-based features underperform the other features. Using the CNN-based features together with other features also worsen the quality of $\hat{\sigma}_x^U$. The reason is because both PriorSum and StdCNN only encode the summaries' *document-independent features* [Cao *et al.*, 2015]. Encoding document-dependent features requires more sophisticated neural models [Wu and Hu, 2018; Narayan *et al.*, 2018] which require considerable time, data and parameter tuning. This undermines the benefits of RELIS. We leave the research for efficient reward representation learning for future work.

RELIS setup. After feature selection, we test RELIS on all three DUC datasets as follows. In line with Cao *et al.* [2017] and Ren *et al.* [2018], we split train and test data in the "leave-one-out" manner so that documents from two datasets are used as the training set, $\overline{\mathcal{X}}$, and documents from the rest as the test set $\mathcal{X} \setminus \overline{\mathcal{X}}$. In each run in the "leave-one-out" experiments, we randomly select 30% data from the training set as the dev set, and select the model with the best performance on the dev set. We use Adam as the gradient optimiser with initial learning rate 10^{-2}. The number of epochs is 10 and batch size is 2. As for RL in RELIS, we use the same *temporal difference* algorithm as Rioux *et al.* [2014]. Full details of our parameter selection and results of using different loss functions are in the supplementary material[4].

[3]In all experiments, we preform statistical significant test by two-tailed t-test and p-value < 0.01.

	DUC'01		DUC'02		DUC'04	
	R_1	R_2	R_1	R_2	R_1	R_2
ICSI	33.31	7.33	35.04	8.51	37.31	9.36
PriorSum	35.98	7.89	36.63	8.97	38.91	10.07
TCSum	**36.45**	7.66	36.90	8.61	38.27	9.66
TCSum$^-$	33.45	6.07	34.02	7.39	35.66	8.66
SRSum	36.04	8.44	**38.93**	**10.29**	39.29	10.70
DeepTD [CI]	28.74	5.95	31.63	7.09	33.57	7.96
REAPER [IS]	32.43	6.84	35.03	8.11	37.22	8.64
RELIS	34.73	**8.66**	37.11	9.12	**39.34**	**10.73**

Table 3: Results of non-RL (top), cross-input [CI] and input-specific [IS] RL approaches (middle) compared with RELIS (bottom).

5 Results

Reward learning. Table 2 compares the quality of our $\hat{\sigma}_x^U$ with other widely used rewards for input-specific RL (see §4). $\hat{\sigma}_x^U$ has significantly higher correlation to the ground-truth ranking compared with all other approaches confirming that our proposed L2R method yields a superior reward oracle.

Summary quality. In Table 3, we compare RELIS with non-RL-based and RL-based summarisation systems. For non-RL-based systems, we report *ICSI* [Gillick and Favre, 2009] maximising the bigram overlap of summary and input using integer linear programming, *PriorSum* [Cao *et al.*, 2015] learning sentence quality with CNNs, *TCSum* [Cao *et al.*, 2017] employing text classification of the input documents, the variant of TCSum without the text classification pre-training (TCSum$^-$) and *SRSum* [Ren *et al.*, 2018] learning sentence relations with word- and sentence-level attentive neural networks to estimate salience.

For RL-based systems, we re-implement REAPER [Rioux *et al.*, 2014] as an input-specific RL, and DeepTD as a cross-input RL. DeepTD is adapted from the DQN-based RL summariser [Yao *et al.*, 2018] and is trained by taking U^* as the rewards (see §4). It uses InferSent to represent summaries. To improve the efficiency and performance of DeepTD, we use *memory replay* and *periodic update*. Further details and the algorithm of DeepTD are in the supplementary material.

RELIS significantly outperforms the other RL-based systems. Note that RELIS and REAPER use the identical RL algorithm for input-specific policy learning; hence the improvement of RELIS is due to the higher quality of the L2R-learnt reward $\hat{\sigma}_x^U$. RELIS outperforms DeepTD because training cross-input policies requires much more data than available in the DUC datasets. At the same time, RELIS performs on par with neural-based TCSum and SRSum, while it requires significantly less data and time to train, as shown next.

Time and resources. We run RELIS, SRSum, DeepTD and REAPER on the same workstation with a 4-core CPU, 8 GB memory and no GPUs. Table 4 compares their average training and test time for each document cluster. RELIS

[4]https://www.dropbox.com/s/slddgj5yfor5u2g/RELIS_IJCAI19_SupplementaryMaterial.pdf?dl=0

reduces the training time of SRSum by two orders of magnitude. At test time, RELIS takes reasonable time to train the input-specific policy, and we believe that it can be further reduced by using more efficient RL algorithms and employing techniques like memory replay or reward shaping.

Unlike TCSum, RELIS requires no additional training data: TCSum uses 1.8 million news articles from New York Times and their category annotations (e.g. Health, Politics, Business) for training. It is worth noting that without using such a massive extra data for the text classification step, the performance of TCSum significantly drops (see TCSum$^-$ in Table 3). The training time of TCSum is unlikely to be shorter than RELIS since TCSum requires to train a CNN-based text classifier before training a CNN-based sentence selector.

	SRSum	DeepTD	REAPER	RELIS
Training	810 s	1,560 s	N/A	2 s
Test	7 s	4 s	31 s	34 s

Table 4: Averaged training and test time for each document cluster. Note that REAPER does not have a training phase.

To summarise, RELIS significantly outperforms RL-based models, and it yields competitive performance compared with the state-of-the-art neural summarisers, with the benefit of needing less training time and data.

6 Discussion & Related Work

RELIS is proposed as a RL-based summarisation paradigm, but it can be applied to other NLG tasks where RL is widely used, e.g., translation, sentence simplification and dialogue generation. For example, to apply RELIS to translation, we just let $\mathcal{X}$ (see §2) be the set of texts in the source language and let $\mathcal{Y}_x$ be the set of all possible translations in the target language for input x; and the error bound of RELIS (Theorem 1) still holds as it is indifferent to the contents in $\mathcal{X}$ and $\mathcal{Y}_x$. Hence, in this section, we discuss the related work in the context of NLG in general.

Reward learning recently receives increasing interests from the machine learning community [Ibarz *et al.*, 2018] but is largely overlooked in NLG until now. Unlike classic RL applications, e.g. robotics and games, where rewards are provided or easy to design, NLG tasks lack strong metrics to measure the quality of the output. Well-known rewards, e.g. ROUGE, are criticised for their low correlation with human evaluations [Chaganty *et al.*, 2018]. Recent work suggests that improving the reward function boosts the performance of RL-based NLG [Kryscinski *et al.*, 2018].

Besides using metrics such as BLEU and ROUGE as rewards to train RL-based NLG, novel rewards have been designed. Kryscinski *et al.* [2018] propose a new reward function for encouraging RL-based summarisers to prefer *novel* words in abstractive summaries. For sentence simplification, Zhang and Lapata [2017] propose a reward function measuring the simplicity, relevance and grammatical correctness of candidate outputs. For machine translation, Nguyen *et al.* [2017] propose a simulated user, which simulates human's ratings on translations, and they use the simulated user to provide rewards for an RL-based translator. However, all rewards discussed above require reference outputs, unlike our learnt reward function which can be used to provide rewards at test time when no reference outputs are available.

Wu and Hu [2018] and Bosselut *et al.* [2018] recently propose to use a large volume of unlabelled data to learn a scorer measuring the *discourse coherence degree* of sentences, and use the scorer as rewards to train cross-input RL. We go beyond their work by proving the soundness of the combination of reward learning and RL, and training input-specific instead of cross-input policies so as to reduce training time and data.

RL-based *interactive NLG* methods elicit human feedback as rewards. Kreutzer *et al.* [2018] elicit pointwise and pairwise feedback on candidate translations to train a cross-input policy. Gao *et al.* [2018] propose using active learning to select appropriate candidate summary pairs and acquire human preferences to improve a heuristics-based reward. However, these methods require much feedback to bring satisfactory results, e.g., at least 50 summary pairs to yield significant improvements [Gao *et al.*, 2018]. RELIS can be used as a pre-training stage for interactive methods, as RELIS first learns a high-quality cross-input reward function and then uses the interactive NLG techniques to elicit a small amount of feedback to fine-tune a user- and input-specific reward function, so at to generate higher-quality and personalised results.

7 Conclusion

We propose a novel RL paradigm called RELIS, which learns a reward function from a reward oracle with learning-to-rank (L2R) algorithms at training time, and then uses the learned reward to train input-specific RL policies at test time. Compared with the widely employed cross-input RL-based summarisation approaches, RELIS avoids the expensive learning of cross-input policies but, instead, efficiently performs L2R and input-specific RL learning. Moreover, RELIS avoids the arduous reward design required in input-specific RL-based summarisation approaches. We prove that, with proper L2R and RL algorithms, RELIS guarantees to produce near-optimal outputs. Our experiments in show that even with linear L2R and standard RL algorithms, RELIS yields performance on par with the state-of-the-art while only requiring a small fraction of data and time to train. Our work lays the theoretical foundation for reward learning in NLG, and we hope it will encourage further research in this direction.

Acknowledgements

This work has been supported by the German Research Foundation (DFG), as part of the QA-EduInf project (GU 798/18-1 and RI 803/12-1) and through the German-Israeli Project Cooperation (DIP, DA 1600/1-1 and GU 798/17-1).

References

[Abbeel and Ng, 2004] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[Agarwal and Collins, 2010] S. Agarwal and M. Collins. Maximum margin ranking algorithms for information retrieval. In *ECIR*, 2010.

[Bosselut *et al.*, 2018] A. Bosselut, A. Celikyilmaz, X. He, J. Gao, P.-S. Huang, and Y. Choi. Discourse-aware neural rewards for coherent text generation. In *NAACL*, 2018.

[Cao *et al.*, 2015] Z. Cao, F. Wei, S. Li, W. Li, M. Zhou, and H. Wang. Learning summary prior representation for extractive summarization. In *ACL-IJCNLP*, 2015.

[Cao *et al.*, 2017] Z. Cao, W. Li, S. Li, and F. Wei. Improving multi-document summarization via text classification. In *AAAI*, 2017.

[Chaganty *et al.*, 2018] A. Chaganty, S. Mussmann, and P. Liang. The price of debiasing automatic metrics in natural language evalaution. In *ACL*, 2018.

[Cheng *et al.*, 2018] C. Cheng, X. Yan, N. Wagener, and B. Boots. Fast policy learning through imitation and reinforcement. In *UAI*, 2018.

[Conneau *et al.*, 2017] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*, 2017.

[Fazel *et al.*, 2018] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi. Global convergence of policy gradient methods for the linear quadratic regulator. In *ICML*, 2018.

[Gao *et al.*, 2018] Y. Gao, C. M. Meyer, and I. Gurevych. APRIL: Interactively learning to summarise by combining active preference learning and reinforcement learning. In *EMNLP*, 2018.

[Gillick and Favre, 2009] D. Gillick and B. Favre. A scalable global model for summarization. In *ILP*, 2009.

[Ibarz *et al.*, 2018] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei. Reward learning from human preferences and demonstrations in Atari. In *NeurIPS*, 2018.

[Kim, 2014] Y. Kim. Convolutional neural networks for sentence classification. In *EMNLP*, 2014.

[Kreutzer *et al.*, 2018] J. Kreutzer, J. Uyheng, and S. Riezler. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *ACL*, 2018.

[Kryscinski *et al.*, 2018] W. Kryscinski, R. Paulus, C. Xiong, and R. Socher. Improving abstraction in text summarization. In *EMNLP*, 2018.

[Li, 2011] H. Li. A short introduction to learning to rank. *IEICE Transactions*, 94-D(10):1854–1862, 2011.

[Lin, 2004] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In *ACL Workshop "Text Summarization Branches Out"*, 2004.

[Maystre and Grossglauser, 2017] L. Maystre and M. Grossglauser. Just sort it! A simple and effective approach to active preference learning. In *ICML*, 2017.

[Mnih *et al.*, 2015] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[Narayan *et al.*, 2018] S. Narayan, S. B. Cohen, and M. Lapata. Ranking sentences for extractive summarization with reinforcement learning. In *NAACL*, 2018.

[Nguyen *et al.*, 2017] K. Nguyen, H. Daumé III, and J. L. Boyd-Graber. Reinforcement learning for bandit neural machine translation with simulated human feedback. In *EMNLP*, 2017.

[Peyrard and Eckle-Kohler, 2017] M. Peyrard and J. Eckle-Kohler. A principled framework for evaluating summarizers: Comparing models of summary quality against human judgments. In *ACL*, 2017.

[Peyrard and Gurevych, 2018] M. Peyrard and I. Gurevych. Objective function learning to match human judgements for optimization-based summarization. In *NAACL*, 2018.

[Ranzato *et al.*, 2015] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba. Sequence level training with recurrent neural networks. *CoRR*, abs/1511.06732, 2015.

[Ren *et al.*, 2018] P. Ren, Z. Chen, Z. Ren, F. Wei, L. Nie, J. Ma, and M. de Rijke. Sentence relations for extractive summarization with deep neural networks. *ACM Trans. Inf. Syst.*, 36(4):39:1–39:32, 2018.

[Rioux *et al.*, 2014] C. Rioux, S. A. Hasan, and Y. Chali. Fear the REAPER: A system for automatic multi-Document summarization with reinforcement learning. In *EMNLP*, 2014.

[Ryang and Abekawa, 2012] S. Ryang and T. Abekawa. Framework of automatic text summarization using reinforcement learning. In *EMNLP/CoNLL*, 2012.

[ShafieiBavani *et al.*, 2018] E. ShafieiBavani, M. Ebrahimi, R. K. Wong, and F. Chen. Summarization evaluation in the absence of human model summaries using the compositionality of word embeddings. In *COLING*, 2018.

[Sutton *et al.*, 1999] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.

[Sutton *et al.*, 2009] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *ICML*, 2009.

[Williams, 1992] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

[Wu and Hu, 2018] Y. Wu and B. Hu. Learning to extract coherent summary via deep reinforcement learning. In *AAAI*, 2018.

[Yao *et al.*, 2018] K. Yao, L. Zhang, T. Luo, and Y. Wu. Deep reinforcement learning for extractive document summarization. *Neurocomputing*, 284, 2018.

[Zhang and Lapata, 2017] X. Zhang and M. Lapata. Sentence simplification with deep reinforcement learning. In *EMNLP*, 2017.