A Reinforcement Learning Approach for Adaptive Single- and Multi-Document Summarization

Stefan Henß
TU Darmstadt,
Germany
stefan.henss@gmail.com

Margot Mieskes
h_da Darmstadt & AIPHES*
Germany
margot.mieskes@h-da.de

Iryna Gurevych
TU Darmstadt & AIPHES*
Germany
gurevych@ukp.informatik.tu-darmstadt.de

Abstract

Reinforcement Learning (RL) is a generic framework for modeling decision making processes and as such very suited to the task of automatic summarization. In this paper we present a RL method, which takes into account intermediate steps during the creation of a summary. Furthermore, we introduce a new feature set, which describes sentences with respect to already selected sentences. We carry out a range of experiments on various data sets – including several DUC data sets, but also scientific publications and encyclopedic articles. Our results show that our approach a) successfully adapts to data sets from various domains, b) outperforms previous RL-based methods for summarization and state-of-the-art summarization systems in general, and c) can be equally applied to single- and multi-document summarization on various domains and document lengths.

1 Introduction

In the history of research on automatic summarization, only few systems have proven themselves capable of handling different summarization scenarios, domains and summarization needs (e.g. single-document summarization vs. multi-document summarization, summarization of news, e-mails, tweets or meetings). Additionally, they rarely take into account that the human summarization procedure involves *decisions* about keeping and/or deleting information (Friend, 2001).

Therefore, we propose Reinforcement Learning (RL) for the task of summarization to model the decision making process involved in producing an extractive summary, i.e. selecting sentences that make up a summary. In our model, the algorithm decides at each step during this selection process which sentence to choose in order to compile an "optimal" summary. As the definition of optimality depends on various factors such as summarization task, needs, domain etc., RL-based methods are in principle highly adaptive to these factors.

Our major contributions are in introducting a new feature set which makes use of the RL methodology in describing sentences with respect to already selected sentences. Second, we use Q-learning in combination with supervised machine learning instead of TD-learning, to model the effects of adding information with respect to any given quality score or error function. Finally, we evaluate our method on several data sets from various domains such as news, scientific publications and encyclopedic articles. Additionally, we tested our method on single- and multi-document summarization scenarios. We compare our results both to available systems and results published in the literature and show that our proposed method outperforms previous RL methods as well as common summarization methods.

The paper is structured as follows: Section 2 presents background and related work. Section 3 contains details of our RL approach and how it differs from previous RL-based summarization methods. Section 4 describes the evaluation of our methods, which data sets we use and the comparison systems. Section 5 presents the results and a discussion of our findings. Section 6 contains the summary and future work.

2 Foundations and Related Work

The work presented here is based on two research areas: automatic summarization and Reinforcement Learning. As reviewing both in detail is beyond the scope of this article, we would like to point the interested reader to works by Nenkova and McKeown (2011), Mani and Maybury (1999)

*Part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) funded by DFG under grant No. GRK 1994/1.

and Mani (2001) inter alia for an overview of the major developments in automatic summarization. For a general introduction to RL, we refer to Sutton and Barto (1998). RL itself has been adopted by the Natural Language Processing (NLP) community for various tasks, among others dialog modeling in Question-Answer-Policies (Misu et al., 2012), for learning dialog management models (Ha et al., 2013), parsing (Zhang and Kwok, 2009) and natural language generation (Dethlefs et al., 2011), which we will not go into details about.

2.1 Reinforcement Learning

RL models contain at least a set of *states* (s_t), possible *actions* (a_t) for each state, and *rewards* (r_t) (or penalties) received for performing actions or reaching certain states. The objective of a RL algorithm is to learn from past observations a *policy* π that seeks desirable states and chooses optimal actions with respect to cumulative future rewards.

Reward Function Rewards or penalties are an important concept in RL, which can be used directly ("online") for example through customer feedback or indirectly ("offline") during training. In many scenarios, collecting the maximum possible immediate rewards at each state (greedy approach) does not yield the best long-term rewards. Optimizing long-term rewards is often solved in RL using *temporal-difference* (TD) learning, where states are valued in terms of their long-term quality, i.e., the maximum sum of rewards one can collect from them. The value of a state s_t can be expressed as follows:

$$V(s_t) = r_t + \mathbb{E}\left[\sum_{i=t+1}^{n} r_i | \pi^*\right] = r_t + \max_{s_{t+1}} V(s_{t+1}) \qquad (1)$$

That is, the value of a state (s_t) equals the immediate reward r_t plus the expected maximum sum of future rewards following an optimal policy π^* from s_t on. This equals the immediate reward r_t plus the maximum value of any possible next state s_{t+1}. Including expected future rewards also allows providing rewards for finals states s_n only (e.g., rating the final summary). These rewards are thus passed through to a function $V(s_t)$.

With large state spaces, V has to be approximated using features of s_t: $\hat{V}(s_t) \approx V(s_t)$, as due to the recursion $V(s_{t+1})$ when calculating $V(s_t)$, computing an exact $V(s_t)$ for each s_t is unfeasible, as one would have to consider all possible paths $s_{t+1},...,s_n$ through states following s_t. Finding an approximation $\hat{V}$ can be achieved through various training algorithms, such as $TD(\lambda)$ (Sutton and Barto, 1998). Given any $\hat{V}$, defining a policy π is straight-forward: At each state s_t, perform the action that yields the maximum (estimated) next-state value $\hat{V}(s_{t+1})$.

***Q* Learning** models the value $Q(s_t, a_t)$ of performing an action a_t in the current state s_t, instead of estimating the value of each possible next state. Facing the large state space of all pairs (s_t, a_t), Q values are also typically not computed exactly for each possible pair individually, but approximated using features of s_t and a_t. As one knows which state s_{t+1} an action a_t leads to in a deterministic environment, the value of *leading* to s_{t+1} is equivalent to the value of *being* at s_{t+1}. Otherwise, Q learning is based on optimizing cumulative future rewards, and the definition of an optimal $Q(s_t, a_t)$ reflects the value of a state-action pair.

2.2 RL in Automatic Summarization

To our knowledge, Ryang and Abekawa (2012) have so far been the first ones who employed RL for the task of summarization. The authors consider the extractive summarization task as a *search* problem, finding the textual units to extract for the summary, where the "final result of evaluation [...] is not available until it finishes" (Ryang and Abekawa, 2012, p. 257). In their framework, a *state* is a subset of sentences and *actions* are transitions from one state to the next. *Rewards* are given "if and only if the executed action is `Finish` and the summary length is appropriate" (Ryang and Abekawa, 2012, p. 259). Otherwise a penalty (i.e. a negative reward) is awarded. Therefore, they only consider the final score of the whole summary. They define the optimal policy as a conditional distribution of an action with regards to the state and the rewards. For learning, they use $TD(\lambda)$. The method was evaluated using the DUC2004 data set (see Section 4 below), and for each cluster, an individual policy was derived.

Recently, Rioux et al. (2014) extended this approach, also using TD. As features, they used bi-grams instead of $tf * idf$ values and employed ROUGE (Lin, 2004b) as part of their reward function. Their evaluation was carried out on the DUC2004 and 2006 general and topic-based multi document summarization and showed that they significantly outperformed previous approaches.

3 Our Method for RL-based Summarization

Similar to R&A(2012), we model each summarization state s_t as a subset of sentences (i.e. a potentially incomplete summary) from the source document(s) to be summarized. For any state s_t, there exists a set of possible actions $\mathscr{A}_s$ to proceed. For us, those are *select* actions for all remaining candidate sentences $c \in D \setminus S$, whose selection would not violate a length threshold *LC*:

$$\mathscr{A}_s = \{c \,|\, c \in D \setminus S, length(\{c\} \cup S) \leq LC\} \tag{2}$$

There are three fundamental differences between our approach and the approach proposed by R&A(2012): First, we define the *reward* function differently. We use rewards during training, based on the reference summaries available. R&A(2012) did not use reference summaries for their rewards, but only define an intrinsic reward function as their focus is on finding an optimal summary with respect to a fixed quality model. We focus on learning selection policies for optimal summaries from external feedback during a training phase. The formal details of this are given below.

The second difference lies in using *Q learning*. This helps us in determining the value of the partial summary s_{t+1} and the value of adding sentence a_t to *state* s_t. The formal details of this will be presented below.

Finally, our method learns one global policy for a specific summarization task, instead of one policy for each document cluster as in R&A(2012).

Reward Functions During training, we give rewards to a specific *action* by comparing the resulting *state* to an expected outcome (e.g. given through reference summaries). In the case of summarization, the *state* is a summary which can still be incomplete and the *action* is the addition of a sentence to this summary.

From our experiments, we found that the increase of the partial summary's evaluation score is a good training feedback for a sentence addition, which is reflected in the equation below:

$$r_t = score(s_{t+1}; H_D) - score(s_t; H_D) \tag{3}$$

In principle, any scoring function for rating the quality of the summary is applicable, thus allowing a flexible adaptation to different summarization objectives and quality criteria. In our evaluation, we use `ROUGE` (Lin, 2004b) to rate each summary with respect to the corresponding human reference summaries H_D (see Section 4 for details).

***Q* Learning** Previous approaches to RL-based summarization used *TD* learning. But despite many recent variations of *TD* learning (see Section 2.1) with linear approximation, for example by Sutton et al. (2009), issues remain in their application for complex tasks such as summarization. First, especially when not using feature transformations like kernel methods, linear models may lack the power of approximating state values precisely. Second, we only know the latest model coefficients, but lack records of past observations – i.e., specific (s_t, a_t) and their rewards – that may be leveraged by more advanced learning methods to discover complex patterns.

Therefore, we use reward functions that depend on human summaries H_D, during a dedicated training phase, i.e., learning an approximation of $Q(s_t, a_t)$. During training, we create summaries, compare them with given H_D and compute rewards as shown above. Finally, we use those rewards in a Q learning algorithm. This is different to R&A(2012) who do not use reference summaries in learning their reward function and thus do not make use of the available, separate training data for learning the state values $\hat{V}(s_t)$. By using H_D, our approach has more capabilities of adopting features of a specific data set by receiving rewards aligned with the training data and evaluation metrics.

As stated earlier, *Q learning* allows us to model the value of the next state s_t after performing action a_t. Q values are typically learned through updates, where the old model is changed according to the difference between the expected $Q(s_t, a_t)$ and its recalculation based on the reward r_{t+1} just received:

$$\begin{aligned} Q(s_t, a_t) \leftarrow{} & Q(s_t, a_t) \\ & + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \end{aligned} \tag{4}$$

The difference in Q is added to the old value with a scaling factor α (the *learning rate*). The discount factor γ emphasizes short-term rewards (see also Table 1). Using approximations of $Q(s_t, a_t)$, this typically means updating the global coefficients used for the linear combination of features of any pair (s_t, a_t), such as in the gradient descent algorithm (Sutton and Barto, 1998).

We learn our policy on a fixed number of training summaries (so-called *episodes*). In case of less training summaries than episodes desired, summaries can be used multiple times. As the observations made from a training summary depend on the strategy learned so far, re-visiting summaries can yield new information each time they are used. During those episodes, a limited number of pairs of (s_t, a_t) are observed, and statistical models based on features of those pairs may suffer from insufficient observations. For example, there may have been few examples of selecting short sentences during training and any correlation between sentence length and summary quality thus may be insignificant. We therefore consider a trade-off between following the most promising actions and exploring seemingly bad decisions that have rarely been made so far. The former strategy repeatedly performs similar actions to learn to better distinguish between the most promising actions, while the latter accounts for wrong estimates by performing "bad" actions and updating the model accordingly if they prove to be rewarding instead. Therefore, during training, we use an ε-greedy strategy, which sometimes selects a random action rather than the most promising one. This is shown in the equation below,

$$\pi_\varepsilon(s_t) = \begin{cases} \arg\max_{a_t} \hat{Q}(s_t, a_t), & x \sim [0,1] \geq \varepsilon^{ep} \\ a_{t+1} \sim \mathscr{A}_t, & else \end{cases} \quad (5)$$

where, ep denotes the number of training episodes, i.e. for $\varepsilon < 1$, selecting the most promising action over a random selection becomes more likely with more training episodes. Using 1,000 training episodes, we chose $\varepsilon = 0.999$, i.e., for the first episode the selection is purely random, but during the second half of the training, we only follow the best strategy for optimizing the model coefficients along those decisions. Once training is completed, our policy is to always choose the action a_t with the highest corresponding $\hat{Q}(s_t, a_t)$, resulting in one policy for the whole task/data set.

To summarize, during training we collect the features of pairs (s_t, a_t) and their corresponding $\hat{Q}$ values at the time after observing r_{t+1}. Knowing the following state s_{t+1}, we not only use features of (s_t, a_t) but also include features of (s_t, a_t, s_{t+1}). We can then use any supervised machine learning algorithm to learn correlations between those triples and corresponding $\hat{Q}$ values. In our observations, this allows for more precise estimates of Q. The supervised machine learning algorithm in our system is a *gradient boosting model* (Friedman, 2002), where Q is updated every 500 actions during our training phase, using the samples of (s_t, a_t, s_{t+1}) and corresponding $\hat{Q}$ as described. With several thousand actions during training, this update rate is sufficient and allows for more complex models that would take too much time with more frequent updates. Gradient boosting iteratively reduces the error of simple regression trees by training a new tree predicting the previous' error. Thereby, our method is able to capture non-linear feature interactions and it is not prone to overfitting, due to the discretization in the basic regression trees and optimization parameters, such as maximum tree depth.

Algorithm 1 Learning Q

```
samples ← $\emptyset$
for $i = 1$ to episodes do
    $ep \leftarrow i \bmod |training\ summaries|$
    $t \leftarrow 0, s_t \leftarrow \emptyset$
    while $length(s_t) \leq LC$, $\mathscr{A}_{s_t,ep} \neq \emptyset$ do
        if $x \sim U(0,1) < 1 - \varepsilon^i$ then
            $a_t \leftarrow \arg\max_{a \in \mathscr{A}_{s_t,ep}} \hat{Q}(s_t, a)$
        else
            $a_t \sim \mathscr{A}_{ep,s_t}$
        end if
        $s_{t+1} \leftarrow s_t \cup \{a_t\}$
        $r_t \leftarrow \mathrm{reward}(s_t, a_t, s_{t+1}; H_{ep})$
        $R_t \leftarrow r_t + \gamma \max_{a \in \mathscr{A}_{ep,s_{t+1}}} \hat{Q}(s_{t+1}, a)$
        $samples \leftarrow samples \cup \{((s_t, a_t, s_{t+1}), R_t)\}$
        if $|samples| \bmod 500 = 0$ then
            $\hat{Q} \leftarrow$ learn-gradient-boosting-model($samples$)
        end if
        $t \leftarrow t + 1$
    end while
end for
```

Our algorithm for learning the RL policy is shown in Algorithm 1. The regression, which predicts features for states and actions, we use gradient boosting as described in Friedman (1999).

Finally, once the training phase is completed, we use the latest gradient boosting model of $\hat{Q}$ to define our policy, i.e., always selecting the most promising actions in its application.

4 Experimental Setup

In this section we describe the data sets, system configuration and evaluation method we used to assess the quality of our algorithm.

Data sets In order to evaluate our method and to compare it to the results published by R&A(2012), we use the DUC2004[1] data set. Additionally, we use the DUC2001 and DUC2002 data sets, as they have been frequently used in the past as evaluation data sets. These also offer the advantage, that they do not only contain multi-document summarization (MDS) tasks, but also single-document summarization (SDS), which allows us to prove the applicability of our proposed method also to SDS. Using the standard training-/test-set splits provided by NIST, we are able to compare our results to those published in the literature.

But as these three data sets entirely consist of news texts, we decided to add other genres as

[1]for all DUC related information see `http://duc.nist.gov/`

well. Two less explored data sets are the ACL-Anthology Reference Corpus (ACL-ARC)[2] (Bird et al., 2008), which contains scientific documents from the NLP domain and Wikipedia[3] (Kubina et al., 2013), which contains encyclopedic documents from a wide range of domains. Both are used in a single document summarization task. Additionally, both the documents and the data sets themselves are considerably larger than the DUC data sets. These data sets allow us to test our method on a range of genres and domains and on considerably larger documents and data sets.

For the DUC data sets several manual summaries are available for the evaluation. For the ACL-ARC we use the abstracts as reference summaries, as it has been done in the past by e.g. Ceylan et al. (2010). Whereas for the Wikipedia, the first paragraph can be regarded as a reference summary, as it has been done by e.g. Kubina et al. (2013). The target lengths for the DUC summarization scenarios are taken from the respective guidelines[4]. The target lengths for ACL and Wikipedia have been determined through the average length of the reference summaries.

System Setup Our method uses several parameters which have to be set prior to training. Table 1 lists these and the settings we used. The main difference between the setup for the DUC and the ACL/Wikipedia-Data is the number of *boosting iterations* (400 vs. 800) and the maximal tree depth (16 vs. 10), which is due to the length differences in the three document sets.

Parameter	DUC	ACL/Wiki
Training episodes	1200	1200
Discount factor	0.01	0.01
ε-greedy	$0.999^{episode}$	$0.999^{episode}$
Boosting iterations	400	800
Shrinkage	0.04	0.04
Max. tree depth	16	10
Min. leaf observations	50	50

Table 1: Experimentally determined parameters used during training and evaluation.

We determined the settings for the listed parameters experimentally. Our aim was to avoid overfitting, while still training predictive models in reasonable time. The parameter settings in Table 1 were found to give the best performance.

[2]`http://acl-arc.comp.nus.edu.sg/`

[3]`http://goo.gl/ySgOS` based on (Kubina et al., 2013)

[4]`http://www-nlpir.nist.gov/projects/duc/guidelines.html`

The individual parameters influence various aspects of the training. The more *training episodes* used, the better the results were. But the number of episodes had to be balanced against overfitting caused by the other parameters. The *Discount factor* weights the contribution of a specific reward once an action has been performed. A too high factor can lead to overfitting. The *ε-greedy* parameter guides how likely it is, that a random action is performed, as this can potentially also lead to an optimal result and is therefore worth exploring. During training, the likelihood of choosing a random action is decreased and the likelihood of choosing an optimal action is increased. The *boosting iterations* guide the training for the gradient boosting. Here, it is crucial to find the balance between good results and computing time, as each training iteration is very time-consuming. *Shrinkage* is similar to the learning rate in other learning methods. We had to balance this parameter between good results and time. The smaller this value is set, the longer each iteration takes and accordingly the training. *Max. tree depth* refers to the size of the regression trees trained by the gradient boosting method. Small trees can hardly generalize, whereas big trees tend to overfit on the training data. *Min. leaf observations* also refers to the regression trees. If the leafs are based on too few training observations, the resulting rules might be based on random observations or overfit on too few observations.

Features The features we use can be grouped into three categories: basic features, linguistic and information retrieval (IR) based features and RL-specific features, which we describe in detail below. The three lists presented here make up the whole set of features used in this work.

Basic and IR-based features The group of basic and IR-based features contains features that are generally used in a wide variety of NLP-tasks, such as text classification (see for example (Manning and Raghavan, 2009, Chp. 13)). They capture surface characteristics of documents, sentences and words, such as the number of tokens, the position of a sentence in a document and the relation between the number of characters and the number of tokens. In addition to the already mentioned surface features, we make use of the ratio for example of the numbers of characters per token. We take into account the stop words in a sentence and the number of stop words in relation to tokens. These features focus on describing the elements of a sin-

Basic/Surface Features	Linguistics and IR-based Features
# of tokens in sentence	mean/max/sum of the sentence's stop word-filtered tokens
# of characters in sentence	total/relative term frequencies (tf) in the source document(s) (docs)
# of characters per #tokens	mean tf compared to the entire corpus, using stemming and $tf * idf$
# of upper case characters per #tokens	the sentence's mean/min/max cosine similarity (*cs*) compared to all other sentences in the docs, stemmed, stop words filtered, bi-grams
absolute position of sentence	*cs* between the $tf * idf$ of the sentence and the combined source docs' $tf * idf$
relative position of sentence	mean/max/min *cs* compared to the sentence's tf vector with those of each source doc
distance of sentence from end	readability score of the sentence
# of chars in sentence before/after	mean/total information content of the tokens (Resnik, 1995)
total # of stop words in sentence	
# of stop words per # of tokens	

Table 2: Basic and commonly used features to describe candidate documents, sentences and words in isolation.

gle sentence or token viewed in isolation.

The surface features only describe sentences or words in the context of the local sentence. We use a set of similar features to describe words and sentences in relation to the whole document. Additionally, we make use of standard linguistic and IR-based features. These features characterize a sentence in terms of the accumulated $tf * idf$ values compared to the document or the document cluster. Other, more linguistically oriented features are based on the cosine similarity between a sentence and all other sentences in the document. Finally, we make use of higher level analysis, such as the readability score (Flesch, 1948; Kincaid et al., 1975). Table 2 shows the full list of basic features (right side) and IR-based features (left side).

RL-based features The third group of features makes use of the specific characteristics of RL and are to our knowledge new to the area of machine learning based summarization. The previous two feature groups describe words and sentences in their local context or in relation to the document they occur in. The RL-based features describe a sentence in the context of the previously selected sentences and how adding this sentence changes the current, hypothetical summary. We also use surface features, such as the number of characters or tokens after the candidate sentence has been added to the already selected sentences. We consider the cosine similarity between the candidate sentence and the sentences selected so far as well. Additionally, we determine the `ROUGE` scores of the hypothetical summary and use the difference between the summary with and without the candidate sentence as a feature. This is based on the definition of "optimality" we use in this work (see also Section 1 above). Using `ROUGE` as part of the features is not problematic in this case, as we use explicit training data to train our reward function, which is then applied to the testing data. The splits are based on the NIST training- and test-sets for the DUC data. The ACL-ARC and Wikipedia data are sufficiently large to be split into two different sets: 5506 for training, 614 for testing for ACL-ARC and 1936 for training, 900 for testing for Wikipedia.

Baselines and Reference Systems We use various baselines and references: First, we use standard baselines such as `HEAD` and `RANDOM` to produce summaries of the data. Second, we use figures reported in the literature. Finally, we make use of available summarization algorithm implementations such as `MEAD`, `SVM` and `SUMY`[5] to produce summaries of the data. `SUMY` contains implementations of several well-known summarization methods, among them the algorithm described by Luhn (1958) (`Luhn` (sumy)), the LSA-based summarization method described by Gong and Liu (2001) (`LSA` (sumy)), the LexRank algorithm (Erkan and Radev, 2004) (`LexRank` (sumy)) and the TextRank algorithm (Mihalcea and Tarau, 2005) (`TextRank` (sumy)). This is especially useful for those data sets that have not yet been extensively used, such as the ACL-ARC and the Wikipedia.

In order to test the contribution of our features and the RL methodology, we used the RL methodology with the individual feature groups. *RL-basic* uses the surface features, *RL-advanced* uses the IR-based features, *RL-non-RL* uses both groups and *RL-RL* uses the RL methodology with the RL features only. Additionally, we implement a Learning-to-Rank (L2R) algorithm to examine the performance of our features, regardless of the RL methodology and use a standard regression-based learning as implemented in WEKA[6].

Evaluation We use the `ROUGE` framework, which is a standard automatic evaluation metric and which allows for comparison be-

[5]`https://github.com/miso-belica/sumy`
[6]`http://www.cs.waikato.ac.nz/ml/weka/`

Feature Description
• new total length in characters and tokens when adding the sentence associated with an RL action • partial summaries before and after adding a sentence are compared to each source document using `ROUGE` precision and recall, and cosine similarity; we add features for the mean/min/max/summed differences between both summaries • mean/min/max cosine similarities between the new sentence and each sentence already included in the summary

Table 3: Reinforcement learning specific features to reflect changes during the creation of the summary.

tween previously reported results and ours. We use `ROUGE` with the following parameters: `-n 4 -m -c 95 -r 1000 -f -A -p 0.5 -t 0 -w 1.2 -2`. Changes for the length constraint were made for DUC 2004 as required (`-b 665` vs. `-l 100`) in the guidelines[7]. For the ACL data, we used the target length of 100 words (`-l 100`), whereas for the Wikipedia data, we used a target length of 290 words (`-l 290`), to reflect the average summary length.

5 Results and Discussion

Our results are indicated with *RL-full*, which is the RL method using the full feature set. Additionally, we use *L2R*, which is the learning-to-rank method, using the non-RL features and *Regression*, which is a standard regression method using the non-RL features. We also determined the benefit of individual feature groups, such as using the RL-method only in combination with the surface features (RL-Surface), the IR- and linguistic based features (RL-Basic) or only the RL-specific features (RL-RL).

Previous RL-based summarization methods were evaluated on the DUC 2004 data set. Table 4 shows the previously reported results compared to our methods. As can be seen, our method clearly outperforms previously published results on R-1. Rioux et al. (2014) achieved a higher R-2 score. This is based on our choice of R-1 as the optimality score, which was based on the correlation between human scores and R-1 (Lin, 2004a).

Rouge	R&A(2012)	R(2014)	RL-full
R-1	0.3901	0.4034	**0.4042**
R-2	0.0948	**0.1140**	0.1012

Table 4: Results for the Multi-Document Scenario based on the DUC 2004 data set, compared to previously reported results.

Table 5 shows the results on the other two MDS tasks (DUC 2001 and 2002), compared to the best result in the literature and the best baseline system. On the DUC2002 data set, the `Luhn`(sumy) baseline performs better on R-1 than our method. On

Year	System	R-1	R-2
2001	Manna et al. (2012)	0.3306	
	`Luhn`(sumy)	0.3218	0.0454
	RL-full	**0.3387**	**0.0740**
2002	Manna et al. (2012)	0.3371	
	`Luhn`(sumy)	**0.3706**	0.0741
	RL-full	0.3660	**0.0810**

Table 5: Results on DUC 2001 and 2002 Multi-Document Summarization Task.

DUC2001 and R-2 in general, our method gives the best performance.

In order to show that our method is also applicable to single document summarization and can also handle larger document collections and longer documents, we also applied our method to SDS tasks of DUC2001 and 2002, ACL and Wikipedia. Table 6 shows our results in comparison to baseline methods. All results show that the full RL setup is superior to other methods, including the TextRank implementation. On DUC 2001, we found a reported R-2 value of 0.204 by Ouyang et al. (2010). The feature analysis shows that for ACL-ARC and Wikipedia the results of the different feature setups and regression learning methods are significantly worse than the full RL setup.

Error Analysis We observed a range of error sources: *First*, manual inspection of the summaries revealed that the automatic summaries could serve as a valid summary, but the overlap between the automatic and the reference summaries are very small. For example in the document on "Superman" from the Wikipedia data (document ID d34b0d339f3f88fe15a8baa17c9c5048), the RL-based summary contained more information about the character and in-world events, whereas the reference summary contained more information about real-world development.

The *second* problem is the too narrow focus and too few details of our summaries. Considering the cluster on the Hurricane Mitch (D30002, DUC2004), we observed that our summary focuses exclusively on the events regarding Honduras and does neither mention the events on the other islands nor the international call for aid.

[7] `http://duc.nist.gov/duc2004/tasks.html`

	DUC 2001		DUC 2002		ACL		Wiki	
System	R-1	R-2	R-1	R-2	R-1	R-2	R-1	R-2
`TextRank`(sumy)	0.4450	0.1866	0.4799	0.2240	0.3739	0.0844	0.4625	0.1256
L2R	0.4490	0.1934	0.4770	0.2181	0.3966	0.1052	0.4706	0.1276
Regression	0.4572	0.1942	0.4847	0.2187	0.3899	0.0883	0.4768	0.1261
RL-surface	0.4384	0.1849	0.4684	0.2130	0.3765	0.0875	0.4542	0.1086
RL-Basic	0.4264	0.1657	0.4539	0.1926	0.3693	0.0782	0.4645	0.1196
RL-RL	0.4005	0.1377	0.4350	0.1700	0.3325	0.0542	0.4721	0.1211
RL-full	**0.4584**	0.1993	**0.4862**	**0.2252**	**0.4117**	**0.1102**	**0.4850**	**0.1321**

Table 6: Results on the Single-Document-Summarization Scenario based on DUC, ACL and Wikipedia data sets, compared to standard methods used in automatic summarization.

Third, we observe that temporal information, dates and numerical facts in general were rare in our summaries (for example in the cluster on the North Korean famine (D30017, DUC2004)). Where numbers are included, we find that they are mentioned in different formats, as opposed to the reference, which makes it hard for `ROUGE` to spot them. One example is from D30017, DUC2004, where the references state that “Two thirds of children under age 7 …”, whereas our summary contains “Two thirds of children under age seven …”.

Fourth, we notice that on the ACL-ARC data very often rows and columns of numbers are extracted, which represent results. While to some extent this is valid in a summary, adding whole tables is not beneficial. Work on translating figures and tables into text has been carried out in the past, but is still an ongoing research topic (see for example (Govindaraju et al., 2013)).

Fifth, we observe that the RL summarizer picked direct speech for the summaries, which did not provide additional information, whereas, direct speech rarely occurs in the references. Detecting direct speech is also its own research topic (see for example (Pareti et al., 2013)).

Finally, we notice that our method extracts considerably longer sentences from the sources, than are those contained in the reference summaries. This problem could be reduced by adding sentence compression to the whole setup.

6 Conclusion and Future Work

In this work, we presented our method for extractive summarization based on RL. We made use of exemplary summaries in the training phase, improved on the learning algorithm through immediate RL rewards and modeling features of states *and* actions, proposed a new, memory-based Q learning algorithm, and used non-linear approximation models. Our method produced global policies for each summarization scenario, rather than a local policy for individual clusters. Finally, we introduced a novel feature set, which exploits the capabilities of reinforcement learning to take into account intermediate results in order to determine the next optimal step. We showed that our system outperforms state-of-the-art methods both on single- and multi-document summarization tasks. Through several, systematic experiments, we showed that the combination of the RL method and the features we employed considerably outperform comparison systems and comparable system setups. Additionally, our method can be adapted to various summarization tasks, such as single- and multi-document summarization, but also to other data sets, such as scientific and encyclopedic articles.

As our error analysis in Section 5 shows, there is room for further improvement on various aspects. Some of these refer to other research topics – such as textually describing tables and figures and detecting direct speech. But some aspects will be tackled in the future: First, reducing the sentence length by applying sentence compression methods. This would allow us to add more information to the summary without violating the length constraint, since we can include more shorter sentences describing various aspects of the summarized topic. The problem of different formats of numbers and abbreviations could be addressed through a normalization step before evaluating. In general, names of persons, places and organizations could be given more importance through Named Entity Recognition features.

Finally, we would like to test our method in other summarization scenarios, such as query-based summarization or data sets such as Twitter.

Acknowledgements

This work has been supported by the German Research Foundation as part of the Research Training Group “Adaptive Preparation of Information from Heterogeneous Sources” (AIPHES) under grant No. GRK 1994/1.

References

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC),* Marrakech, Morocco, 26 May – 1 June 2008.

Hakan Ceylan, Rada Mihalcea, Umut Özertem, Elena Lloret, and Manuel Palo. 2010. Quantifying the limits and success of extractive summarization systems across domains. In *Human Lanugage Technlogies: The 2010 Annual Conference of the North American Chapter of the ACL,* Los Angeles, California, June 2010, pages 903–911.

Nina Dethlefs, Heriberto Cuyahuitl, and Jette Viethen. 2011. Optimising natural language generation decision making for situated dialogue. In *Proceedings of the 12th SIGdial Workshop on Discourse and Dialogue,* Portland, Oregon, 17-18 June 2011.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research*, 22:457–479.

Rudolf Flesch. 1948. A new readability yardstick. *The Journal of applied psychology*, 32(3):221–233.

Jerome H. Friedman. 1999. Stochastic gradient boosting. `http://astro.temple.edu/~msobel/courses_files/StochasticBoosting%28gradient%29.pdf`, March.

Jerome H Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

Rosalie Friend. 2001. Effects of Strategy Instruction on Summary Writing of College Students. *Contemporary Educational Psychology*, 26(1):3–24.

Yihong Gong and Xin Liu. 2001. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and development in information retrieval (SIGIR-01)*, pages 19–25.

Vidhya Govindaraju, Ce Zhang, and Christopher Ré. 2013. Understanding tables in context using standard NLP toolkits. In *Proceedings of the 51st Conference of the Association for Computational Linguistics* Sofia, Bulgaria 4–9 August 2013, pages 658–664.

Eun Young Ha, Christopher M. Mitchell, Kristy Elizabeth Boyer, and James C. Lester. 2013. Learning dialogue management models for task-oriented dialogue with parallel dialogue and task streams. In *Proceedings of the 14th SIGdial Workshop on Discourse and Dialogue,* Metz, France, 22-24 August 2013.

Peter Kincaid, Robert Fishburne Jr, Richard Rogers, and Brad Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Jeff Kubina, John Conroy, and Judith Schlesinger. 2013. ACL 2013 MultiLing Pilot Overview. In *Proceedings of the MultiLing 2013 Workshop on Multilingual Multi-document Summarization*, pages 29–38, Sofia, Bulgaria. Association for Computational Linguistics.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation – how many samples are enough? In *Proceedings of NTCIR Workshop 4, Tokyo, Japan, June 2-4, 2004.*

Chin-Yew Lin. 2004b. ROUGE: A Package for Automatic Evaluation of Summaries. In *Proceedings of the Workshop on Text Summarization Branches Out at ACL 2004,* Barcelona, Spain, 25–26 July, 2006, pages 74–81.

Hans Peter Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development*, 2(2):159–165.

Inderjeet Mani and Mark T. Maybury, editors. 1999. *Advances in Automatic Text Summarization.* Cambridge/MA, London/England: MIT Press.

Inderjeet Mani. 2001. *Automatic Summarization.* Number 3 in Natural Language Processing (NLP). John Benjamins Publishing Company, P.O Box 36224, 1020 Amsterdam, The Netherlands.

Sukanya Manna, Byron J. Gao, and Reed Coke. 2012. A subjective logic framework for multi-document summarization. In *Proceedings of the 24th International Conference on Computational Linguistics,* Mumbay, India, December, 2012, pages 797–808.

Christopher D. Manning and Prabhakar Raghavan. 2009. *An Introduction to Information Retrieval.* Cambridge University Press.

Rada Mihalcea and Paul Tarau. 2005. A language independent algorithm for single and multiple document summarization. In *Proceedings of the 2nd International Joint Conference on Natural Language Processing,* Jeju Island, South Korea, 11–13 October 2005, pages 19–24.

Teruhisa Misu, Kallirroi Georgila, Anton Leuski, and David Traum. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. In *Proceedings of the 13th SIGdial Workshop on Discourse and Dialogue,* Seoul, South Korea, 05-06 July 2012.

Ani Nenkova and Kathleen McKeown. 2011. *Automatic Summarization*. Foundations and Trends in Information Retrieval. Now Publishers Inc.

You Ouyang, Wenjie Li, Qin Lu, and Renxian Zhang. 2010. A study on position information in document summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING),* Beijing, China, 23–27 August 2010, pages 919–927.

Silvia Pareti, Tim O'Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* Seattle, Washington, USA, October 2013, pages 989–999.

Philip Resnik. 1995. Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada*, pages 448–453.

Cody Rioux, Sadid A. Hasan, and Yllias Chali. 2014. Fear the reaper: A system for automatic multi-document summarization with reinforcement learning. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), October 25-29, 2014, Doha, Qatar.*, pages 681–690.

Seonggi Ryang and Takeshi Abekawa. 2012. Framework of automatic text summarization using reinforcement learning. In *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* Seattle, Washington, USA, October 2013, pages 256–265.

Richard S Sutton and Andrew G Barto. 1998. *Reinforcement Learning: An Introduction*, volume 1. Cambridge Univ Press.

Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. 2009. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Canada, June 14-18, 2009*, pages 993–1000. ACM.

Lidan Zhang and Chan Kwok. 2009. Dependency parsing with energy-based reinforcement learning. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT), Paris, October 2009.*