Unsupervised Compressive Text Summarisation with Reinforcement Learning

Anonymous ACL submission

Abstract

Recently, compressive text summarisation offers a balance between the conciseness issue of extractive summarisation and the factual hallucination issue of abstractive summarisation. However, most existing compressive summarisation methods are supervised, relying on the expensive effort of creating a new training dataset with corresponding compressive summaries. In this paper, we propose an unsupervised compressive summarisation method that utilises reinforcement learning to optimise a summary's semantic coverage and fluency by simulating human judgment on summarisation quality. Our model consists of an extractor agent and a compressor agent, and both agents have a multi-head attentional pointer-based structure. The extractor agent first chooses salient sentences from a document, and then the compressor agent compresses these extracted sentences by selecting salient words to form a summary without using reference summaries to compute the summary reward. That is, a parallel dataset with document-summary pairs is not required to train the proposed model. To the best of our knowledge, our proposed method is the first work on unsupervised compressive summarisation. Experimental results on three widely used datasets, Newsroom, CNN/DM, and XSum, show that our model achieves promising performance and significant improvement on Newsroom in terms of the ROUGE metric. [1]

1 Introduction

Text summarisation aims to condense a given document into a short and succinct summary that best covers the document's semantics with the least redundancy. It helps users quickly browse and understand long documents by focusing on their most important content (Mani, 2001).

The majority of existing summarisation methods are either extractive or abstractive. Extractive methods (Narayan et al., 2018b; Zhang et al., 2019; Liu and Lapata, 2019; Zhong et al., 2020) select salient sentences from a document to form its summary. It ensures the production of grammatically and factually correct summaries, though the output summaries could be inflexible. Abstractive methods (See et al., 2017; Paulus et al., 2018; Zhang et al., 2020; Laban et al., 2020) involve natural language generation to generate a summary for a given document. However, it is highly prone to produce contents that are unfaithful and non-factual to the original document (Maynez et al., 2020).

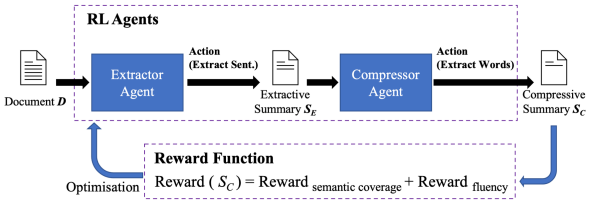

Figure 1: Illustration of our proposed URLComSum. It summarises a given text document $\mathbf{D}$ first into an extractive summary $\mathbf{S_E}$ then into a compressive summary $\mathbf{S_C}$. The agents are trained with reinforcement learning to mimic human judgment for achieving high summary quality in terms of semantic coverage and fluency.

A recent approach to text summarisation is compressive summarisation which selects words, instead of sentences, from an input document to assemble a summary. Compressive summarisation offers a sidestep to improve the factuality and conciseness of a summary. Existing compressive methods (Mendes et al., 2019; Xu and Durrett, 2019; Desai et al., 2020) often first select the salient sentences of a given document and then further compress these selected sentences by a sentence compressor to form a summary. However, most of these methods are supervised, which require a parallel dataset with document-summary pairs to train. Since the ground-truth summaries of existing datasets are usually obtained abstractively and do not contain supervision information of extrac-

[1] Our source code will be publicly available at GitHub.

tion or compression, existing compressive methods require creating new datasets of which the ground-truth summaries are compressive.

Therefore, we propose an unsupervised compressive summarisation method with reinforcement learning, namely URLComSum. As illustrated in Figure 1, URLComSum consists of two modules, an extractor agent and a compressor agent. We model the sentence and word representations using a Bi-LSTM (Graves and Schmidhuber, 2005) with multi-head attention (Vaswani et al., 2017) to capture both the long-range dependencies and the relationship between each word and the others, and each sentence and other sentences. We use a Pointer Network (Vinyals et al., 2015) to solve our task of finding the optimal subset of sentences and words to be extracted since the Pointer Network is well-known for tackling combinatorial optimization problems. The extractor agent uses a hierarchical multi-head attentional Bi-LSTM model for learning the sentence representation of the input document and a Pointer Network for extracting the salient sentences of a document given a length budget. To further compress these extracted sentences all together, the compressor agent uses a multi-head attentional Bi-LSTM model for learning the word representation and a Pointer Network for selecting the words to assemble a summary.

Note that we also investigated the popular transformer model (Vaswani et al., 2017) in our proposed framework to replace Bi-LSTM for learning the sentence and word representations. However, it was noticed the transformer-based agents do not perform as well as the Bi-LSTM-based ones while training from scratch with the same training procedure. The difficulties of training a transformer model have also been discussed in (Popel and Bojar, 2018; Liu et al., 2020). Besides, the commonly used pre-trained transformer models, such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020), require high computational resources and usually use subword-based tokenizers. They are not suitable for URLComSum since our compressor agent points to words instead of subwords. Therefore, at this stage Bi-LSTM is a simpler and better choice.

As an unsupervised method, URLComSum does not require a parallel training dataset with document-summary pairs. We propose an unsupervised reinforcement learning training procedure to mimic human judgment: to reward the model that achieves high summary quality in terms of semantic coverage and language fluency. Inspired by Word Mover's Distance (Kusner et al., 2015), the semantic coverage reward of a summary is measured by Wasserstein distance (Peyré et al., 2019) between the semantic distribution of the original document and that of the summary. The fluency reward of a summary is measured by Syntactic Log-Odds Ratio (SLOR) (Pauls and Klein, 2012), which is a referenceless fluency evaluation metric. According to prior works, SLOR is effective in sentence compression (Kann et al., 2018), and has better correlation to human acceptability judgments (Lau et al., 2017).

Note that our RL reward shares similarity with (Laban et al., 2020) in terms of optimising summary quality by both coverage and fluency. However, our definitions and calculations of coverage and fluency are different. Specifically, (Laban et al., 2020) defined coverage as a TF-IDF based keyword coverage and trained a separate model to compute the coverage reward. In addition, for the fluency reward, (Laban et al., 2020) utilised the score from a pretrained language model directly. The key contributions of this paper are:

- We propose an unsupervised method for compressive text summarisation with reinforcement learning, namely URLComSum. To the best of our knowledge, URLComSum is the first work on unsupervised compressive summarisation.

- URLComSum consists of an extractor agent and a compressor agent to extract and compress a document to a summary given a length budget. We design a multi-head attentional pointer-based neural network for learning the representation and for extracting salient sentences and words.

- We propose to mimic human judgment by optimising summary quality in terms of the semantic coverage reward, measured by Wasserstein distance, and the fluency reward, measured by Syntactic Log-Odds Ratio (SLOR).

- Comprehensive experimental results on three widely used datasets, including CNN/DM, XSum, Newsroom, demonstrate that URLComSum achieves great performance in terms of the ROUGE metric.

2 Related Work

Most of the existing works on neural text summarisation are extractive, abstractive, and compressive-based. In this section, we review existing text summarisation methods in these three categories.

2.1 Extractive Methods

Extractive methods usually follow the sentence ranking conceptualisation, and an encoder-decoder scheme is generally adopted. An encoder formulates document or sentence representations, and a decoder predicts extraction classification labels. The supervised models commonly rely on creating proxy extractive training labels for training (Cheng and Lapata, 2016; Nallapati et al., 2017; Jia et al., 2021), which can be noisy and may not be reliant. Some methods were proposed to tackle this issue by training with reinforcement learning (Narayan et al., 2018b; Luo et al., 2019) to optimise the ROUGE metric directly. Various unsupervised methods (Zheng and Lapata, 2019; Padmakumar and He, 2021) were also proposed to leverage pre-trained language models to compute sentences similarities and select important sentences. Although these methods have significantly improved summarisation performance, since the entire sentences are extracted individually, the redundant information that appears in the salient sentences may not be minimized effectively.

2.2 Abstractive Methods

Abstractive methods formulate text summarisation as a sequence-to-sequence generation task, with the source document as the input sequence and the summary as the output sequence. Most existing methods follow the supervised RNN-based encoder-decoder framework (See et al., 2017; Zhang et al., 2020). As supervised learning with ground-truth summaries may not provide useful insights on human judgment approximation, reinforcement training was proposed to optimise the ROUGE metric (Paulus et al., 2018), and to fine-tune a pre-trained language model (Laban et al., 2020). These models naturally learn to integrate knowledge from the training data while generating an abstractive summary. Prior studies showed that these generative models are highly prone to external hallucination, thus may generate contents that are unfaithful to the original document (Maynez et al., 2020).

2.3 Compressive Methods

Compressive methods select words from a given document to assemble a summary. Due to the lack of training dataset, not until recently there have emerged works for compressive summarisation (Zhang et al., 2018; Mendes et al., 2019; Xu and Durrett, 2019; Desai et al., 2020). The formulation of compressive document summarisation is usually a two-stage extract-then-compress approach: it first extracts salient sentences from a document, then compresses the extracted sentences to form its summary. Most of these methods are supervised, which require a parallel dataset with document-summary pairs to train. However, the ground-truth summaries of existing datasets are usually abstractive-based and do not contain supervision information needed for extractive summarisation or compressive summarisation (Xu and Durrett, 2019; Mendes et al., 2019; Desai et al., 2020). Several reinforcement learning based methods (Zhang et al., 2018) use existing abstractive-based datasets for training, which is not aligned for compression. Note that existing compressors often perform compression sentence by sentence. As a result, the duplicated information among multiple sentences could be overlooked. Therefore, to address these limitations, we propose a novel unsupervised compressive method by exploring the reinforcement learning strategy to mimic human judgment and perform text compression instead of sentence compression.

3 Methodology

As shown in Figure 1, our proposed compressive summarisation method, namely URLComSum, consists of two components, an extractor agent and a compressor agent. Specifically, the extractor agent selects salient sentences from a document $\mathbf{D}$ to form an extractive summary $\mathbf{S_E}$, and then the compressor agent compresses $\mathbf{S_E}$ by selecting words to assemble a compressive summary $\mathbf{S_C}$.

3.1 Extractor Agent

Given a document $\mathbf{D}$ consisting of a sequence of M sentences $\{\mathbf{s}_i | i = 1, ..., M\}$, and each sentence $\mathbf{s}_i$ consisting of a sequence of N words $\{\mathbf{we}_{ij} | j = 1, ..., N\}$ [2], the extractor agent aims to produce an extractive summary $\mathbf{S_E}$ by learning sentence representation and selecting L_E sentences from $\mathbf{D}$.

[2] We have pre-fixed the length of each sentence and each document by padding.

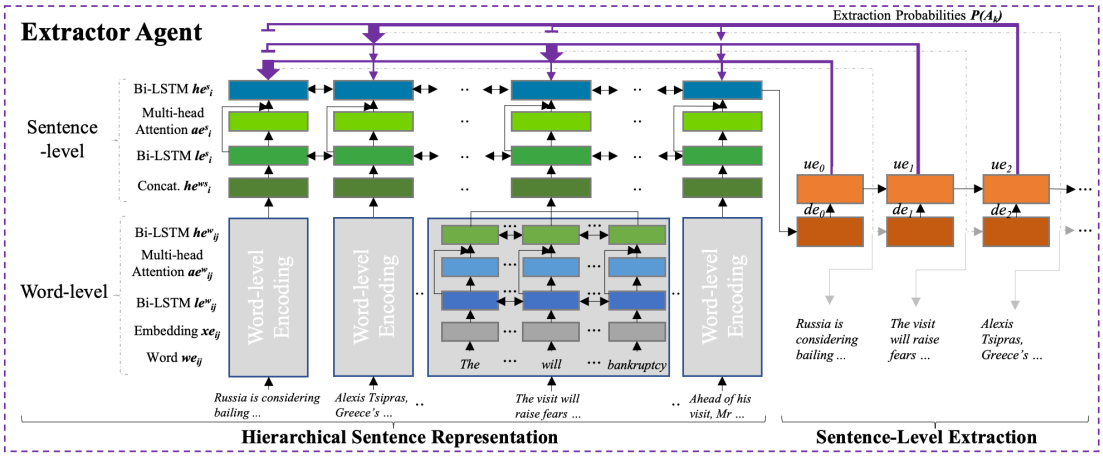

Figure 2: Illustration of the extractor agent.

As illustrated in Figure 2, we design a hierarchical multi-head attentional Bi-LSTM model for learning the sentence representations of the document and using a Pointer Network to extract sentences based on their representations.

3.1.1 Hierarchical Sentence Representation

To model the local context of each sentence and the global context between sentences, we use two-levels Bi-LSTMs to model this hierarchical structure, one at the word level to encode the word sequence of each sentence, one at the sentence level to encode the sentence sequence of the document. To model the context-dependency of the importance of words and sentences, we apply two levels of multi-head attention mechanism (Vaswani et al., 2017), one at each of the two-level Bi-LSTMs.

Given a sentence $\mathbf{s}_i$, we encode its words into word embeddings $\mathbf{xe}_i = \{\mathbf{xe}_{ij}|j = 1, ..., N\}$ by $\mathbf{xe}_{ij} = Enc(\mathbf{we}_{ij})$, where $Enc()$ denotes a word embedding lookup table. Then the sequence of word embeddings are fed into the word-level Bi-LSTM to produce an output representation of the words $\mathbf{le}^w$:

$$\mathbf{le}^w_{ij} = \overleftrightarrow{\text{LSTM}}(\mathbf{xe}_{ij}), j \in [1, N] \ . \quad (1)$$

To utilize the multi-head attention mechanism to obtain $\mathbf{ae}^w_i = \{\mathbf{ae}^w_{i1}, ..., \mathbf{ae}^w_{iN}\}$ at word level, we define $Q_i = \mathbf{le}^w_i$, $K_i = V_i = \mathbf{xe}_i$,

$$\mathbf{ae}^w_i = \text{MultiHead}(Q_i, K_i, V_i) \ . \quad (2)$$

The concatenation of $\mathbf{le}^w_i$ and $\mathbf{ae}^w_i$ of the words are fed into a Bi-LSTM and the output of the Bi-LSTM is concatenated to obtain the local context representation $\mathbf{he}^{ws}_i$ for each sentence $\mathbf{s_i}$:

$$\begin{aligned} \mathbf{he}^w_{ij} &= \overleftrightarrow{\text{LSTM}}([\mathbf{le}^w_{ij}; \mathbf{ae}^w_{ij}]), j \in [1, N] \ , \\ \mathbf{he}^{ws}_i &= [\mathbf{he}^w_{i1}, ..., \mathbf{he}^w_{iN}] \ . \end{aligned} \quad (3)$$

To further model the global context between sentences, we apply a similar structure at sentence level. $\mathbf{he}^{ws} = \{\mathbf{he}^{ws}_i|i = 1, ..., M\}$ are fed into the sentence-level Bi-LSTM to produce output representation of the sentences $\mathbf{le}^s$:

$$\mathbf{le}^s_i = \overleftrightarrow{\text{LSTM}}(\mathbf{he}^{ws}_i), i \in [1, M] \ . \quad (4)$$

To utilize the multi-head attention mechanism to obtain $\mathbf{ae}^s = \{\mathbf{ae}^s_1, ..., \mathbf{ae}^s_M\}$ at sentence level, we define $Q = \mathbf{le}^s$, $K = V = \mathbf{he}^{ws}$,

$$\mathbf{ae}^s = \text{MultiHead}(Q, K, V) \ . \quad (5)$$

The concatenation of the Bi-LSTM output $\mathbf{le}^s$ and the multi-head attention output $\mathbf{ae}^s$ of the sentences are fed into a Bi-LSTM to obtain the sentences' final representation $\mathbf{he}^s = \{\mathbf{he}^s_1, ..., \mathbf{he}^s_M\}$:

$$\mathbf{he}^s_i = \overleftrightarrow{\text{LSTM}}([\mathbf{le}^s_i; \mathbf{ae}^s_i]), i \in [1, M] \ . \quad (6)$$

3.1.2 Sentence-Level Extraction

Similar to (Chen and Bansal, 2018), we use an LSTM-based Pointer Network to decode the above sentence representations $\mathbf{he}^s = \{\mathbf{he}^s_1, ..., \mathbf{he}^s_M\}$ and extract sentences recurrently to form an extractive summary $\mathbf{S_E} = \{A_1, ..., A_k, ..., A_{L_E}\}$ with L_E sentences, where A_k denotes the sentence extracted at the k-th time step.

At the k-th time step, the pointer network receives the sentence representation of the previous extracted sentence and has hidden state de_k. It first obtains a context vector de'_k by attending to $\mathbf{he}^s$:

$$\begin{aligned}\mathbf{ue}_i^k &= v^T \tanh(W_1\mathbf{he}_i^s + W_2 de_k), i \in (1, ..., M) \ , \\ \mathbf{ae}_i^k &= \text{softmax}(\mathbf{ue}_i^k), i \in (1, ..., M) \ , \\ de'_k &= \sum_{i=1}^{M} \mathbf{ae}_i^k \mathbf{he}_i^s \ ,\end{aligned} \tag{7}$$

where v, W_1, W_2 are learnable parameters of the pointer network. Then it predicts the extraction probability $p(A_k)$ of a sentence:

$$\begin{aligned}de_k &\leftarrow \left[de_k, de'_k\right] \ , \\ \mathbf{ue}_i^k &= v^T \tanh(W_1\mathbf{he}_i^s + W_2 de_k), i \in (1, ..., M) \ , \\ p(A_k|A_1, ..., A_{k-1}) &= \text{softmax}(\mathbf{ue}^k) \ .\end{aligned} \tag{8}$$

This decoding process iterates until L_E sentences are selected. The selected sentences are assembled to form S_E.

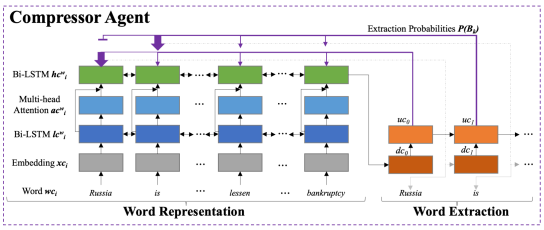

Figure 3: Illustration of the compressor agent.

3.2 Compressor Agent

Given an extractive summary $\mathbf{S_E}$ consisting of a sequence of words $\mathbf{wc} = \{\mathbf{wc}_i | i = 1, ..., N\}$, the compressor agent aims to produce a compressive summary $\mathbf{S_C}$ by selecting L_C words from $\mathbf{S_E}$. As illustrated in Figure 3, it has a multi-head attentional Bi-LSTM model to learn the word representations. It uses a pointer network to extract words based on their representations.

3.2.1 Word Representation

Given a sequence of words $\mathbf{wc}$, we encode the words into word embeddings $\mathbf{xc} = \{\mathbf{xc}_i | i = 1, ..., N\}$ by $\mathbf{xc}_i = Enc(\mathbf{wc}_i)$. Then the sequence of word embeddings are fed into a Bi-LSTM to produce the words' output representation $\mathbf{lc}^w$:

$$\mathbf{lc}_i^w = \overleftrightarrow{\text{LSTM}}(\mathbf{xc}_i), i \in [1, N] \ . \tag{9}$$

To utilize the multi-head attention mechanism to obtain $\mathbf{ac}^w = \{\mathbf{ac}_1^w, ..., \mathbf{ac}_N^w\}$, we define $Q = \mathbf{lc}^w$, $K = V = \mathbf{xc}$,

$$\mathbf{ac}^w = \text{MultiHead}(Q, K, V) \ . \tag{10}$$

The concatenation of $\mathbf{lc}^w$ and $\mathbf{ac}^w$ of the words are fed into a Bi-LSTM to obtain the representation $\mathbf{hc}_i^w$ for each word $\mathbf{wc_i}$:

$$\mathbf{hc}_i^w = \overleftrightarrow{\text{LSTM}}([\mathbf{lc}_i^w; \mathbf{ac}_i^w]), i \in [1, N] \ . \tag{11}$$

3.2.2 Word-Level Extraction

The word extractor of the compressor agent shares the same structure as that of the extractor agent's sentence extractor. To select the words based on the above word representations $\mathbf{hc}^w = \{\mathbf{hc}_1^w, ..., \mathbf{hc}_N^w\}$, the word extractor decodes and extracts words recurrently to produce $\{B_1, ..., B_k, ..., B_{L_C}\}$, where B_k denotes the word extracted at the k-th time step.

3.3 Reward in Reinforcement Learning

Our unsupervised reinforcement learning based training strategy is designed to mimic human judgment by maximising summary quality reward $\text{Reward}(\mathbf{D}, \mathbf{S})$ in terms of the semantic coverage reward $\text{Reward}_{\text{cov}}(\mathbf{D}, \mathbf{S})$ and the fluency reward $\text{Reward}_{\text{flu}}(\mathbf{S})$. We use the compressive summary $\mathbf{S_C}$ to compute the reward. For simplicity, we denote $\text{Reward}(\mathbf{D}, \mathbf{S_C})$ as $\text{Reward}(\mathbf{D}, \mathbf{S})$. Given an original document $\mathbf{D}$, $\text{Reward}(\mathbf{D}, \mathbf{S})$ is a weighted sum of $\text{Reward}_{\text{cov}}(\mathbf{D}, \mathbf{S})$ and $\text{Reward}_{\text{flu}}(\mathbf{S})$:

$$\begin{aligned}\text{Reward}(\mathbf{D}, \mathbf{S}) = w_{\text{cov}} &\text{Reward}_{\text{cov}}(\mathbf{D}, \mathbf{S}) \\ &+ w_{\text{flu}} \text{Reward}_{\text{flu}}(\mathbf{S}) \ ,\end{aligned} \tag{12}$$

where w_{cov} and w_{flu} denote the weights of semantic coverage reward and fluency reward, respectively.

3.3.1 Semantic Coverage Reward

Intuitively, a good summary is supposed to be close to the original document regarding their semantic distributions. We measure $\text{Reward}_{\text{cov}}$ by computing the Wasserstein distance between the corresponding semantic distributions of the document $\mathbf{D}$ and the summary $\mathbf{S}$, which is the minimum cost required to transport the semantics from $\mathbf{D}$ to $\mathbf{S}$.

We denote $\mathbf{D} = \{d_i | i = 1, ..., N\}$ to represent a document, where d_i indicates the count of the i-th token (i.e., word or phrase in a vocabulary of size N). Similarly, for a summary

$\mathbf{S} = \{s_j | j = 1, ..., N\}$, s_j is respect to the count of the j-th token . The semantic distribution of a document is characterized in terms of normalised term frequency. The term frequency of the i-th token in the document $\mathbf{D}$ and the j-th token in the summary $\mathbf{S}$ are denoted as $\text{TF}_\mathbf{D}(i)$ and $\text{TF}_\mathbf{S}(j)$, respectively. The frequency of each token within a document or a summary is formulated by counting the normalised token's occurrence over the text. Formally, $\text{TF}_\mathbf{D}(i)$ and $\text{TF}_\mathbf{S}(j)$ can be computed as:

$$\text{TF}_\mathbf{D}(i) = \frac{d_i}{\sum_{j=1}^{N} d_j}, \text{TF}_\mathbf{S}(j) = \frac{s_j}{\sum_{j=1}^{N} s_j} . \tag{13}$$

By defining $\text{TF}_\mathbf{D} = \{\text{TF}_\mathbf{D}(i)\} \in \mathbf{R}^N$ and $\text{TF}_\mathbf{S} = \{\text{TF}_\mathbf{S}(j)\} \in \mathbf{R}^N$, we have the semantic distributions within the document $\mathbf{D}$ and the summary $\mathbf{S}$ respectively. Note that after the normalization $\text{TF}_\mathbf{D}$ and $\text{TF}_\mathbf{S}$ have an equal total token quantities of 1 and can be completely transported from one to the other.

The transportation cost matrix $\mathbf{C}$ is obtained by measuring the semantic similarity between each of the tokens. Given a pre-trained tokeniser and token embedding model with N tokens, define $\mathbf{v}_i$ to represent the feature embedding of the i-th token. Then the transport cost c_{ij} from the i-th to the j-th token is computed based on the cosine similarity:

$$c_{ij} = 1 - \frac{< \mathbf{v}_i, \mathbf{v}_j >}{\|\mathbf{v}_i\|_2 \|\mathbf{v}_j\|_2} . \tag{14}$$

An optimal transport plan $\mathbf{T}^* = \{t^*_{i,j}\} \in \mathbf{R}^{N \times N}$ in pursuit of minimizing the transportation cost can be obtained by solving the following optimization problem:

$$\mathbf{T}^* = \underset{\mathbf{T}}{\text{argmin}} \sum_{i,j=1}^{N} t_{ij} c_{ij} \text{ , s.t. } \sum_{j=1}^{N} t_{ij} = d_i,$$
$$\sum_{i=1}^{N} t_{ij} = s_j, \quad t_{ij} \geq 0, \quad \forall i, j \in \{1, ..., N\} . \tag{15}$$

The Wasserstein distance measuring the distance between the two semantic distributions $\text{TF}_\mathbf{D}$ and $\text{TF}_\mathbf{S}$ with the optimal transport plan is computed by:

$$d_W(\text{TF}_\mathbf{D}, \text{TF}_\mathbf{S} | \mathbf{C}) = \sum_{i,j} t^*_{ij} c_{ij} . \tag{16}$$

$\text{Reward}_{\text{cov}}(\mathbf{D}, \mathbf{S})$ of the summary $\mathbf{S}$ in respect to the document $\mathbf{D}$ can be further defined as:

$$\text{Reward}_{\text{cov}}(\mathbf{D}, \mathbf{S}) = 1 - d_W(\text{TF}_\mathbf{D}, \text{TF}_\mathbf{S} | \mathbf{C}) . \tag{17}$$

3.3.2 Fluency Reward

We utilize Syntactic Log-Odds Ratio (SLOR) (Pauls and Klein, 2012) to measure $\text{Reward}_{\text{flu}}$. SLOR relies on a trained language model and is adjusted for rare words such as named entities.

Given a trained language model LM, $\text{Reward}_{\text{flu}}$ of a summary $\mathbf{S}$ is defined as:

$$\text{Reward}_{\text{flu}}(S) = \frac{1}{|S|}(\log(P_{LM}(S)) - \log(P_U(S))) , \tag{18}$$

where $P_{LM}(S)$ denotes the probability of the summary assigned by the LM, $p_U(S) = \prod_{t \in S} P(t)$ denotes the unigram probability for rare word adjustment, and $|S|$ denotes the sentence length.

URLComSum uses the Self-Critical Sequence Training (SCST) method (Rennie et al., 2017), since this training algorithm has demonstrated promising results in text summarisation (Paulus et al., 2018; Laban et al., 2020). For a given input document, the model produces two separate output summaries: the sampled summary $\mathbf{S}^s$, obtained by sampling the next pointer t_i from the probability distribution at each time step i, and the baseline summary $\hat{\mathbf{S}}$, obtained by always picking the most likely next pointer t at each i. The training objective is to minimise the following loss:

$$Loss = -(\text{Reward}(\mathbf{D}, \mathbf{S}^s) - \text{Reward}(\mathbf{D}, \hat{\mathbf{S}})) \cdot \frac{1}{N} \sum_{i=1}^{N} \log p(t^s_i | t^s_1, ..., t^s_{i-1}, \mathbf{D}) , \tag{19}$$

where N denotes the length of the pointer sequence, which is the number of extracted sentences for the extractor agent and the number of extracted words for the compressor agent.

Minimising the loss is equivalent to maximising the conditional likelihood of $\mathbf{S}^s$ if the sampled summary $\mathbf{S}^s$ outperforms the baseline summary $\hat{\mathbf{S}}$, i.e. $\text{Reward}(\mathbf{D}, \mathbf{S}^s) - \text{Reward}(\mathbf{D}, \hat{\mathbf{S}}) > 0$, thus increasing the expected reward of the model.

4 Experimental Results and Discussions

4.1 Datasets

To validate the effectiveness of our proposed URLComSum, we conducted comprehensive experiments on three widely used datasets: *Newsroom*

(Grusky et al., 2018), *CNN/DailyMail (CNN/DM)* (Hermann et al., 2015), and *XSum* (Narayan et al., 2018a). These datasets obtained from HuggingFace Datasets [3] are the standard English single-document datasets with manually-written summaries, as summarised in Table 4 in Appendix A.

4.2 Implementation Details

The details of hyperparameter settings and software used are in Appendix B and C respectively.

Method	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	33.9	23.2	30.7
LEAD-WORD	34.9	23.1	30.7
Supervised Methods			
PG+Coverage (Abs.)	27.5	13.3	23.6
EXCONSUMM (Ext.)*	31.9	16.3	26.9
EXCONSUMM (Ext.+Com.)*	25.5	11.0	21.1
Unsupervised Methods			
SumLoop (Abs.)	27.0	9.6	26.4
TextRank (Ext.)	24.5	10.1	20.1
URLComSum (Ext.)	33.9	**23.2**	30.0
URLComSum (Ext.+Com.)	**34.6**	22.9	**30.5**

Table 1: Comparisons on the **Newsroom** test set. The symbol * indicates that the model is not directly comparable to ours as it is based on a subset (the "Mixed") of the dataset.

Method	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	40.0	17.5	32.9
LEAD-WORD	39.7	16.6	32.5
Supervised Methods			
PG+Coverage (Abs.)	39.5	17.3	36.4
ML+RL (Abs.)	39.9	15.8	36.9
REFRESH (Ext.)	40.0	18.2	36.6
LATENTCOM (Ext.)	41.1	18.8	37.5
LATENTCOM (Ext.+Com.)	36.7	15.4	34.3
JECS (Ext.)	40.7	18.0	36.8
JECS (Ext.+Com.)	41.7	18.5	37.9
EXCONSUMM (Ext.)	41.7	18.6	37.8
EXCONSUMM (Ext.+Com.)	40.9	18.0	37.4
CUPS (Ext.)	43.7	20.6	40.0
CUPS (Ext.+Com.)	44.0	20.6	40.4
Unsupervised Methods			
SumLoop (Abs.)	37.7	14.8	**34.7**
TextRank (Ext.)	34.1	12.8	22.5
PacSum (Ext.)	**40.3**	**17.6**	24.9
PMI (Ext.)	36.7	14.5	23.3
URLComSum (Ext.)	40.0	17.5	32.9
URLComSum (Ext.+Com.)	39.3	16.0	32.2

Table 2: Comparisons on the **CNN/DM** test set.

[3] https://huggingface.co/docs/datasets/

Method	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	19.4	2.4	12.9
LEAD-WORD	18.3	1.9	12.8
Supervised Methods			
PG+Coverage (Abs.)	28.1	8.0	21.7
MatchSum (Ext.)	24.9	4.7	18.4
CUPS (Ext.)	24.2	5.0	18.3
CUPS (Ext.+Com.)	26.0	5.4	19.9
Unsupervised Methods			
TextRank (Ext.)	19.0	3.1	12.6
PacSum (Ext.)	**19.4**	2.7	12.4
PMI (Ext.)	19.1	**3.2**	12.5
URLComSum (Ext.)	**19.4**	2.4	**12.9**
URLComSum (Ext.+Com.)	18.0	1.8	12.7

Table 3: Comparisons on the **XSum** test set.

We compare our model with existing compressive methods which are all supervised, including *LATENTCOM* (Zhang et al., 2018), *EXCONSUMM* (Mendes et al., 2019), *JECS* (Xu and Durrett, 2019), *CUPS* (Desai et al., 2020). Since our method is unsupervised, we compare with unsupervised extractive and abstractive methods, including *TextRank* (Mihalcea and Tarau, 2004), *PacSum* (Zheng and Lapata, 2019), *PMI* (Padmakumar and He, 2021), and *SumLoop* (Laban et al., 2020). We also report the state-of-the-art methods evaluated on the three datasets for reference, including *PG+Coverage* (See et al., 2017), *REFRESH* (Narayan et al., 2018b), *ML+RL* (Paulus et al., 2018), *MatchSum* (Zhong et al., 2020). To better evaluate compressive summarisation methods, we followed a similar concept as LEAD baseline (See et al., 2017) and created *LEAD-WORD* baseline which extracts the first several words of a document as a summary.

4.3 Quantitative Analysis

The commonly used ROUGE metric (Lin, 2004) is adopted for our quantitative analysis. It evaluates the content consistency between a generated summary and a reference summary. In detail, the ROUGE-n score calculates the number of overlapping n-grams between a generated summary and a reference summary. The ROUGE-L score considers the longest common subsequence between a generated summary and a reference summary.

The experimental results of URLComSum on different datasets are shown in Table 1, Table 2 and Table 3 in terms of ROUGE-1, ROUGE-2 and ROUGE-L F-scores. (Ext.), (Abs.), and (Com.)

denote that the method is extractive, abstractive, and compressive, respectively. Note that on the three datasets, LEAD and LEAD-WORD baseline are considered strong baselines in the literature and sometimes perform better than the state-of-the-art supervised and unsupervised models. As (See et al., 2017; Padmakumar and He, 2021) also discussed, it could be due to the Inverted Pyramid writing structure (Pöttker, 2003) of news articles, in which important information is often located at the beginning of an article and a paragraph.

As shown in Table 1, our URLComSum method significantly outperforms all the unsupervised and supervised ones on Newsroom. This clearly demonstrates the effectiveness of our proposed method. Note that, unlike supervised EXCONSUMM, our reward strategy contributes to the performance improvement when the compressor agent is utilized. For example, in terms of ROUGE-L, EXCONSUMM (Ext.+Com.) does not outperform EXCONSUMM (Ext.), while URLComSum(Ext.+Com.) outperforms URLComSum(Ext.). Similarly, as shown in Table 3, our URLComSum methods achieve the best performance among all the unsupervised methods on XSum, in terms of ROUGE-1 and ROUGE-L. URLComSum underperforms in ROUGE-2, which may be due to the trade-off between informativeness and fluency. Note that the improvement on Newsroom is greater than those on CNN/DM and XSum, which could be the fact that the larger size of Newsroom is more helpful for training our model.

As shown in Table 2, our proposed URLComSum methods achieve comparable performance with other unsupervised methods on CNN/DM. Note that URLComSum does not explicitly take position information into account while some extractive methods take advantage of the lead bias of CNN/DM, such as PacSum and LEAD. Nevertheless, we observe that URLComSum(Ext.) achieves the same result as the LEAD baseline. Even URLComSum is unsupervised, eventually the extractor agent learns to select the first few sentences of the documents, which follows the principle of the aforementioned Inverted Pyramid writing structure.

It is noticed that URLComSum(Ext.+Com.) generally achieves higher ROUGE-1 and ROUGE-L scores than its extractive version on Newsroom. Meanwhile, on CNN/DM and XSum, the compressive version has slightly lower ROUGE scores than the extractive version. We observe similar behaviour in the literature of compressive summarisation. It may be because the datasets are from the news domain. The sentences of news articles have dense information, and compression does not help much to further condense the documents.

We also investigated the transferablility of URLComSum. As shown on Table 5 in Appendix D, the models are able to obtain similar performance even they are trained on one dataset and tested on another. This demonstrates that the acquired knowledge of the agents are transferable to other datasets, which is not always true to other learning-based summarisation methods.

We observed that the extraction and compressive approaches usually obtain better results than the abstractive in terms of ROUGE scores on CNN/DM and Newsroom, and vice versa on XSum. It may be because CNN/DM and Newsroom contain summaries that are usually more extractive, whereas XSum's summaries are highly abstractive.

4.4 Qualitative Analysis

In Figure 4, 5, 6 in Appendix E, summaries produced by URLComSum are shown together with the reference summaries of the sample documents in the CNN/DM, XSum, and Newsroom datasets. This demonstrates that our proposed URLComSum method is able to identify salient sentences and words and produce reasonably fluent summaries even without supervision information.

5 Conclusion

In this paper, we have presented URLComSum, the first unsupervised method for compressive text summarisation. Our model consists of an extractor agent and a compressor agent. The extractor agent first chooses salient sentences from a document, and then the compressor agent compresses these extracted sentences by selecting salient words to form a summary. As a result, our proposed URLComSum can help reduce redundant information existing in multiple sentences. To achieve unsupervised training of the extractor and compressor agents, we devise a reinforcement learning strategy to simulate human judgement on summary quality and optimize the summary's semantic coverage and fluency reward. Comprehensive experiments on three widely used benchmark datasets demonstrate the effectiveness of our proposed URLComSum and the great potential of unsupervised compressive summarisation.

References

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shrey Desai, Jiacheng Xu, and Greg Durrett. 2020. Compressive summarization with plausibility and salience modeling. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *International Conference on Neural Information Processing Systems (NeurIPS)*.

Ruipeng Jia, Yanan Cao, Haichao Shi, Fang Fang, Pengfei Yin, and Shi Wang. 2021. Flexible non-autoregressive extractive summarization with threshold: How to extract a non-fixed number of summary sentences. In *AAAI Conference on Artificial Intelligence*.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! In *Conference on Computational Natural Language Learning (CoNLL)*.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *International Conference on Machine Learning (ICML)*.

Philippe Laban, Andrew Hsi, John Canny, and Marti A. Hearst. 2020. The summary loop: Learning to write abstractive summaries without examples. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jey Han Lau, Alexander Clark, and Shalom Lappin. 2017. Grammaticality, acceptability, and probability: A probabilistic view of linguistic knowledge. *Cognitive Science*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han. 2020. Understanding the difficulty of training transformers. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Ling Luo, Xiang Ao, Yan Song, Feiyang Pan, Min Yang, and Qing He. 2019. Reading like HER: Human reading inspired extractive summarization. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Inderjeet Mani. 2001. *Automatic summarization*. John Benjamins Publishing.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Afonso Mendes, Shashi Narayan, Sebastião Miranda, Zita Marinho, André F. T. Martins, and Shay B. Cohen. 2019. Jointly extracting and compressing documents with summary state representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

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

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018a. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Ranking sentences for extractive summarization with reinforcement learning. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Vishakh Padmakumar and He He. 2021. Unsupervised extractive summarization using pointwise mutual information. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Adam Pauls and Dan Klein. 2012. Large-scale syntactic language modeling with treelets. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations (ICLR)*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*.

Martin Popel and Ondřej Bojar. 2018. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics (NeurIPS)*.

Horst Pöttker. 2003. News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*.

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *IEEE conference on computer vision and pattern recognition (CVPR)*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *International Conference on Neural Information Processing Systems (NeurIPS)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *International Conference on Neural Information Processing Systems (NeurIPS)*.

Jiacheng Xu and Greg Durrett. 2019. Neural extractive text summarization with syntactic compression. In *Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning (ICML)*.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xingxing Zhang, Furu Wei, and Ming Zhou. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Hao Zheng and Mirella Lapata. 2019. Sentence centrality revisited for unsupervised summarization. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Annual Meeting of the Association for Computational Linguistics (ACL)*.

A Dataset Details

We followed (Mendes et al., 2019) to set $\mathbf{L_E}$ for Newsroom and (Zhong et al., 2020) to set $\mathbf{L_E}$ for CNN/DM and XSum. We also followed their protocols to set $\mathbf{L_C}$ by matching the average number of words in summaries.

Dataset	Newsroom	CNN/DM	XSum
#Sentences in Doc.	27	39	19
#Tokens in Doc.	659	766	367
$\mathbf{L_E}$	2	3	2
$\mathbf{L_C}$	26	58	24
Train	995,041	287,113	204,045
Test	108,862	11,490	11,334

Table 4: Overview of the three datasets. #Sentences in Doc. and #Tokens in Doc. denote the average number of sentences and words in the documents respectively. $\mathbf{L_E}$ denotes the number of sentences to be selected by the extractor agent. $\mathbf{L_C}$ denotes the number of words to be selected by the compressor agent. Train and Test denote the size of train and test sets.

B Hyperparameter Details

We set the LSTM hidden size to 150 and the number of recurrent layers to 3. We used two layers of the Bi-LSTM and multi-head attention pairs. We performed hyperparameter searching for w_{cov} (ranging from 1 to 4) and w_{flu} (ranging from 1 to 4) and decided to set $w_{\text{cov}} = 1$, $w_{\text{flu}} = 2$ in all our experiments since it provides more balanced results across the datasets. We trained the URLComSum with AdamW (Loshchilov and Hutter, 2018) with learning rate 0.01 with a batch size of 3 for about 12 hours.

C Software and Hardware Used

We obtained the word embedding from the pre-trained GloVe (6B tokens, 300-hidden-size) (Pennington et al., 2014). For the pre-trained tokeniser and token embedding models used for computing semantic coverage reward, we used BERT (base version) from HuggingFace [4]. For the trained language model used for computing the fluency reward, we chose the pre-trained GPT2 (base version) from HuggingFace since it is a powerful language model. It was fine-tuned on the target domain. To compute the Wasserstein distances, we utilized the POT library [5]. Our experiments were run on a GeForce GTX 1080 GPU card.

We obtained our ROUGE scores by using the pyrouge package [6].

[4] https://huggingface.co

[5] https://pythonot.github.io

[6] https://pypi.org/project/pyrouge/

D Transferability Analysis

Trained on \Tested on		Newsroom			CNN/DM			XSum		
		ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-1	ROUGE-2	ROUGE-L
Newsroom	Ext.	33.88	23.24	29.97	40.03	17.47	32.92	19.44	2.39	12.94
	Ext.+Com.	34.57	22.92	30.51	39.32	16.28	32.30	18.23	1.87	12.78
CNN/DM	Ext.	33.88	23.24	29.97	40.03	17.47	32.92	19.44	2.39	12.94
	Ext.+Com.	34.03	21.97	30.01	39.27	15.97	32.18	18.08	1.80	12.69
XSum	Ext.	33.88	23.24	29.97	40.03	17.47	32.92	19.44	2.39	12.94
	Ext.+Com.	33.01	21.03	29.08	38.63	15.22	31.60	17.97	1.80	12.69

Table 5: Transferability analysis of URLComSum.

E Sample Summaries

The following shows the sample summaries generated by URLComSum on the CNN/DM, XSum, and Newsroom datasets. Sentences extracted by the URLComSum extractor agent are highlighted. Words selected by the URLComSum compressor agent are underlined in red. Our unsupervised method URLComSum can identify salient sentences and words to produce a summary with reasonable semantic coverage and fluency.

Source Document:

Russia is considering bailing out Greece in exchange for the country's 'assets', it was reported last night. Alexis Tsipras, Greece's prime minister, will meet Vladimir Putin in Moscow today, amid reports that the Kremlin will offer controversial loans and discounts on supplies of natural gas in a bid to lessen its dependence on the West . The visit will raise fears the radical left government is looking east in search of alternative sources of finance as it bids to avoid bankruptcy. Scroll down for video . Alexis Tsipras, Greece's (...)

Reference Summary:

Alexis Tsipras, Greece's prime minister, will meet Vladimir Putin in Moscow . The meeting comes amid reports Russia is considering bailing out Greece . Reports Kremlin may offer loans and discounts on supplies of natural gas .

URLComSum:

Russia is considering bailing out Greece in exchange for the country ' s ' assets ' , it was reported last night . Alexis Tsipras , Greece ' s prime minister , will meet Vladimir Putin in Moscow today , amid reports that the Kremlin will offer controversial loans and discounts on supplies of natural gas in a bid its raise alternative as bids to avoid bankruptcy .

Figure 4: A sample summary produced by URLComSum on the CNN/DM dataset. The summary generated by URLComSum has ROUGE-1, ROUGE-2, and ROUGE-L F-Scores of 68.8, 52.7, and 62.4 respectively, with semantic coverage reward 0.76 and fluency reward 0.64, while the reference summary has semantic coverage reward 0.80 and fluency reward 0.62.

Source Document:

Paul Robson is the second trader at the Dutch bank to plead guilty to trying to rig the Yen Libor rate and the first Briton to do so. Last year Rabobank paid $1bn (Â£597m) to US and European regulators for its part in the global rate-rigging scandal. Barclays Bank, Royal Bank of Scotland and Lloyds Bank have all previously been fined for rate rigging. (...)

Reference Summary:

A former senior trader at Rabobank has pleaded guilty to interest rate rigging in the US.

URLComSum:

Paul Robson is the second trader at the Dutch bank to plead guilty to trying to rig the Yen Libor rate and global rate-rigging scandal .

Figure 5: A sample summary produced by URLComSum on the XSum dataset. The summary generated by URLComSum has ROUGE-1, ROUGE-2, and ROUGE-L F-Scores of 38.1, 20.0, and 33.3 respectively, with semantic coverage reward 0.77 and fluency reward 0.56, while the reference summary has semantic coverage reward 0.73 and fluency reward 0.59.

Source Document:

A man armed with a rifle has killed four people in a rampage in Girona province, north-east Spain, police say. The gunman walked into a bar in the town of Olot, 120km (70 miles) north of Barcelona, and shot two men - reportedly a father and son who were both construction workers. Minutes later, he went to a bank and killed two staff, police said. (...)

Reference Summary:

A man armed with a rifle kills four people in a shooting rampage in north-east Spain, police say.

URLComSum:

A man armed with a rifle has killed four people in a rampage in Girona province , north-east Spain , police say . The gunman walked into a bar in

Figure 6: A sample summary produced by URLComSum on the Newsroom dataset. The summary generated by URLComSum has ROUGE-1, ROUGE-2, and ROUGE-L F-Scores of 76.6, 62.2, and 76.6 respectively, with semantic coverage reward 0.79 and fluency reward 0.61, while the reference summary has semantic coverage reward 0.76 and fluency reward 0.65.