**ARED$\text{SUM}$: Adaptive Redundancy-Aware Iterative Sentence Ranking for Extractive Document Summarization**

Keping Bi$^1$, Rahul Jha$^2$, W. Bruce Croft$^1$, Asli Celikyilmaz$^3$

$^1$University of Massachusetts Amherst, $^2$Microsoft, $^3$Microsoft Research AI

{kbi, croft}@cs.umass.edu, rajh@microsoft.com, asli@ieee.org

**Abstract**

Redundancy-aware extractive summarization systems score the redundancy of the sentences to be included in a summary either jointly with their salience information or separately as an additional sentence scoring step. Previous work shows the efficacy of jointly scoring and selecting sentences with neural sequence generation models. It is, however, not well-understood if the gain is due to better encoding techniques or better redundancy reduction approaches. Similarly, the contribution of salience versus diversity components on the created summary is not studied well. Building on the state-of-the-art encoding methods for summarization, we present two adaptive learning models: $\text{ARED}$$\text{SUM}$-SEQ that jointly considers salience and novelty during sentence selection; and a two-step $\text{ARED}$$\text{SUM}$-CTX that scores salience first, then learns to balance salience and redundancy, enabling the measurement of the impact of each aspect. Empirical results on CNN/DailyMail and NYT50 datasets show that by modeling diversity explicitly in a separate step, $\text{ARED}$$\text{SUM}$-CTX achieves significantly better performance than $\text{ARED}$$\text{SUM}$-SEQ as well as state-of-the-art extractive summarization baselines.

**1 Introduction**

Extractive summarization is the task of creating a summary by identifying and concatenating the most important sentences in a document (Liu and Lapata, 2019; Zhang et al., 2019; Zhou et al., 2018). Given a partial summary, the decision to include another sentence in the summary depends on two aspects: salience, which represents how much information the sentence carries; and redundancy, which represents how much information in the sentence is already included in the previously selected sentences.

Although there have been a few studies on redundancy a long time ago, most recent research on extractive summarization focuses on salience alone. They usually model sentence salience as a sequence labeling task (Kedzie et al., 2018; Cheng and Lapata, 2016) or classification task (Zhang et al., 2019) and do not conduct redundancy removal. Previous methods that consider redundancy usually use a separate step after salience scoring to handle redundancy, denoted as sentence selection (Carbonell and Goldstein, 1998; McDonald, 2007; Lin and Bilmes, 2011). Sentence selection often follows a greedy iterative ranking process that outputs one sentence at a time by taking into account the redundancy of candidate sentences with previously selected sentences.

Several approaches for modeling redundancy in sentence selection have been explored: heuristics-based methods such as Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998), Trigram Blocking (TriBLK) (Liu and Lapata, 2019), or model based approaches (Ren et al., 2016), etc. Heuristic-based methods are not adaptive since they usually apply the same rule to all the documents. Model-based approaches depend heavily on feature engineering and learn to score sentences via regression with point-wise loss, which has been shown to be inferior to pairwise loss or list-wise loss in ranking problems (Liu et al., 2009).

Redundancy has also been handled jointly with salience during the scoring process using neural sequence models (Zhou et al., 2018). $\text{NEU}$$\text{SUM}$ (Zhou et al., 2018) scores sentences considering their salience as well as previous sentences in the output sequence and learns to predict the sentence with maximum relative gain given the partial output summary. Despite the improved efficacy, it is not well-understood if the gain is due to better encoding or better redundancy-aware iterative ranking approaches (i.e., the sequence generation).

In this work, we propose to study different types of redundancy-aware iterative ranking techniques.
for extractive summarization that handle redundancy separately or jointly with salience. Extending BERTSUMEXT (Liu and Lapata, 2019), a state-of-the-art extractive summarization model, which uses heuristic-based Trigram Blocking (TriBLK) for redundancy elimination, we propose two supervised redundancy-aware iterative sentence ranking methods for summary prediction. Our first model, AREDSUM-SEQ, introduces a transformer-based conditional sentence order generator network to score and select sentences by jointly considering their salience and diversity within the selected summary sentences. Our second model, AREDSUM-CTX, uses an additional sentence selection model to learn to balance the salience and redundancy of constructed summaries. It incorporates surface features (such as n-gram overlap ratio and semantic match scores) to instrument the diversity aspect. We compare the performance of our redundancy-aware sentence ranking methods with trigram-blocking (Liu and Lapata, 2019), as well as summarization baselines with or without considering redundancy on two commonly used datasets, CNN/DailyMail and New York Times (NYT50). Experimental results show that our proposed AREDSUM-CTX can achieve better performance by reducing redundancy and outperform all the baselines on these two datasets. The model’s advantage can be attributed to its adaptiveness to scenarios in which redundancy removal has different potential gains.

In summary, our contributions are: 1) we propose two redundancy-aware iterative ranking methods for extractive summarization extending BERTSUMEXT; 2) we conduct comparative studies between our redundancy-aware models as well as the heuristic-based method that BERTSUMEXT uses; 3) our proposed AREDSUM-CTX significantly outperforms BERTSUMEXT and other competitive baselines on CNN/DailyMail and NYT50.

2 Related Work

Extractive summarization methods are usually decomposed into two subtasks, i.e., sentence scoring and sentence selection, which deal with salience and redundancy, respectively.

**Salience Scoring.** Graph-based models are widely used methods to score sentence salience in summarization (Erkan and Radev, 2004; Mihalcea and Tarau, 2004; Wan and Yang, 2006). There are also extensions to such methods, e.g., with clustering (Wan and Yang, 2008) or leveraging graph neural networks (Wang et al., 2020). Classical supervised extractive summarization uses classification or sequence labeling methods such as Naive Bayes (Kupiec et al., 1999), maximum entropy (Osborne, 2002), conditional random fields (Galley, 2006) or hidden markov model (Conroy et al., 2004). Human engineered features are heavily used in these methods such as word frequency and sentence length (Nenkova et al., 2006).

In recent years, neural models have replaced older models to score the salience of sentences. Hierarchical LSTMs and CNNs have replaced manually engineered features. LSTM decoders are employed to do sequence labeling (Cheng and Lapata, 2016; Nallapati et al., 2017; Kedzie et al., 2018). These architectures are widely used and also extended with reinforcement learning (Narayanan et al., 2018; Dong et al., 2018). More recently, summarization methods based on BERT (Devlin et al., 2018) have been shown to achieve state-of-the-art performance (Liu and Lapata, 2019; Zhang et al., 2019; Zhong et al., 2019; Zhou et al., 2020) on salience for extractive summarization.

**Sentence Selection.** There are relatively fewer methods that study sentence selection to avoid redundancy. Integer Linear Programming based methods (McDonald, 2007) formulate sentence selection as an optimizing problem under the summary length constraint. Lin and Bilmes (2011) propose to find the optimal subset of sentences with submodular functions. Greedy strategies such as Maximal Marginal Relevance (MMR) (Carbonell and Goldstein, 1998) select the sentence that has maximal salience score and is minimally redundant iteratively. Trigram blocking (Liu and Lapata, 2019) follows the intuition of MMR and filters out sentences that have trigram overlap with previously extracted sentences. Ren et al. (2016) leverage two groups of handcrafted features to capture informativeness and redundancy respectively during sentence selection. In contrast to learning a separate model for sentence selection, Zhou et al. (2018) propose to jointly learn to score and select sentences with a sequence generation model. However, it is not compared with other redundancy-aware techniques, and it is not clear whether its improvement upon other methods is from the sequence generation method or the encoding technique.

In this paper, we compare the efficacy of different sentence selection techniques grounded on the
We formulate single document extractive summarization as a task of iterative sentence ranking. At each step, the goal is to extract \( t \) sentences, i.e., \( \hat{S}_k = \{\hat{s}_k|1 \leq k \leq t, \hat{s}_k \in D\} \), from \( D \) that can best summarize it. With a limit of selected sentence count \( l \), the process of extracting sentences can be considered as a \( l \)-step iterative ranking problem. 

At each \( k \)-th step (\( 1 \leq k \leq l \)), given the current summary \( \hat{S}_{k-1} \), a new sentence \( s_k \) is selected from the remaining sentences \( D \setminus \hat{S}_{k-1} \) and added to the summary. Function \( M(\hat{S}_k; S^*) \)\(^1\) measures the similarity between the extracted summary \( \hat{S}_k \) and the ground truth summary \( S^* \). The objective is to learn a scoring function \( f(\cdot) \) so that the best sentence \( \hat{s}_k \) selected according to \( f(\cdot) \) can maximize the gain of the output summary:

\[
\arg\max_f M(\{\hat{s}_k\} \cup \hat{S}_{k-1}; S^*) \\
\hat{s}_k = \arg\max_{s_i \in D \setminus \hat{S}_{k-1}} f(\{s_i\} \cup \hat{S}_{k-1}) 
\]

\( \hat{s}_k \) needs to be both salient in the document and novel in the current context \( \hat{S}_{k-1} \). Note that at the beginning \( \hat{S}_0 = \emptyset \).

Since ground truth summaries \( S^* \) of existing summarization corpora are usually abstractive summaries written by experts, previous studies on extractive summarization usually extract a group of pseudo ground truth sentences \( S^* \) from \( D \) based on their similarities to the ground truth summaries \( S^* \) for training purposes. Then labels 1 and 0 are assigned to sentences in \( S^* \) and the other sentences in \( D \). In this case, \( M(\hat{S}_i; \hat{S}^*) \) is used to guide training instead of \( M(\hat{S}_i; S^*) \).

### 4 Redundancy-Aware Summarization

Most recent redundancy-aware extractive summarization systems use heuristics to select diverse sentences after salience scoring (Cao et al., 2015; Ren et al., 2017; Liu and Lapata, 2019). Among them, BERTSUMEXT (Liu and Lapata, 2019) is a state-of-the-art model with trigram-blocking (TriBlk) that reduces redundancy by filtering out sentences that have trigram overlap with previously selected ones at each time step. As we empirically show later in § 6, heuristics can be effective on some datasets yet harmful on others since it applies the same rule to all the documents.

In contrast, we present an adaptive learning process for redundancy-aware extractive summarization, AREDSUM, and introduce two methods, AREDSUM-SEQ and AREDSUM-CTX, extending BERTSUMEXT by either consider redundancy jointly with salience during sentence scoring or separately with an additional selection model.

#### 4.1 Document Encoder

First, we introduce the sentence and document encoder shared by both our variations of AREDSUM, shown in Figure 1. In sentence-level encoding, a [SEP] token is appended to each sentence to indicate the sentence boundaries and a [CLS] token is inserted before each sentence in the document to aggregate the information of the sentence. In addition to token and positional embeddings, as in BERTSUMEXT (Liu and Lapata, 2019), we also use interval segment embeddings \( E_A \) and \( E_B \) to distinguish sentences at odd and even positions in the document respectively. Following multiple transformer encoder layers, we represent each sentence \( s_i \) by the output representation of the [CLS] symbol preceding \( s_i \). These symbols capture the features of the following tokens in the sentence while attending over all other tokens in the document through the transformer layers.

We further conduct document-level encoding on the sentence-level representations from the [CLS] tokens, denoted as \( E_s \), as well as their positional embeddings, \( E'_s \), with another stack of transformer layers. We add a document embedding \( E_D \) before the sequence of sentence embeddings to represent the whole document. The final representation of \( D \) and each sentence \( s_i \) can be obtained from the output of the multiple transformer layers, denoted as \( h_s \) and \( h_D \).

#### 4.2 AREDSUM-SEQ: Sequence Generation

Our first model, AREDSUM-SEQ, strictly considers the order of the target selected sentences while jointly modeling the redundancy and salience of the next sentence. It uses a transformer decoder

\(^1\)In § 4 and experiments, we use ROUGE to define \( M(\cdot) \).
module (Vaswani et al., 2017) to learn to select and order a sequence of sentences from the document as a summary. Our model is different from standard auto-regressive decoders. Each decoder block takes in a sequence of tokens (word-units) as input to generate the next possible token from a pre-defined vocabulary. Instead, our decoder is a conditional model that takes a sequence of sentence representations as input and selects a sentence with the maximum gain to be included in the summary from the rest of the document’s sentences.

Following a standard transformer encoder-decoder architecture (Vaswani et al., 2017), at each decoding step $k$, a current hidden state is obtained with a stack of transformer decoder layers:

$$h_{s_{k-1}}' = \text{DEC}([E_{s_1}; \cdots; E_{s_{k-1}}]; [h_{s_1}; h_{s_2}; \cdots; h_{s_L}])$$ (2)

where $[h_{s_1}, h_{s_2}, \cdots, h_{s_L}]$ are the output sentence representations after the document-level encoding in Figure 1 and $[E_{s_1}, \cdots, E_{s_{k-1}}]$ are the embeddings of the so far selected sentences. $\hat{S}_{k-1} = \emptyset$ and $E_{s_0} = 0$ when $k = 1$. Note that sentence embeddings that are fed to the document-level transformer encoders, i.e., $E_{s_1}, \cdots, E_{s_L}$, are used to represent the sentences in the target decoding space. Then, a two-layer MLP is used to score a candidate sentence $s_i$ given the hidden state $h_{s_{k-1}}'$:

$$o_g(s_i) = W_{2s} \tanh(W_{1s} [h_{s_{k-1}}'; E_{s_i}])$$ (3)

where $W_{2s}$ and $W_{1s}$ are the weights of the MLP (we omit the bias parameters for simplicity), and $[; ;]$ denotes vector concatenation. In case the salience of $s_i$ is not sufficiently captured in $o_g(s_i)$, we calculate a matching score $o_f(s_i)$ between $s_i$ and the global context $D$, regardless of which sentences are selected previously $^2$:

$$o_f(s_i) = \tanh(h_D W_{ds} h_{s_i})$$ (4)

where $W_{ds}$ is matrix for bilinear matching; $h_D$ and $h_{s_i}$ are the embeddings of the document $D$ and sentence $s_i$ output by the document-level encoder. The final score is the linear combination of $o_f$ and $o_g$ using the weight $W_o$:

$$o(s_i) = W_o (o_f(s_i) + o_g(s_i))$$ (5)

The probability of any sentence $s_i$ being selected at the $k$-th step is the softmax of $o(s_i)$ over the remaining candidate sentences $s_j$ in $D$:

$$P(\hat{s}_k = s_i | \hat{S}_{k-1}) = \frac{\exp(o(s_i))}{\sum_{s_j \in D \setminus \hat{S}_{k-1}} \exp(o(s_j))}$$ (6)

Following NEUSUM (Zhou et al., 2018), we train AREDSUM-SEQ to optimize for the relative ROUGE-F1 gain of each sentence with respect to so-far selected sentences $\hat{S}_{k-1}$.

$$g(s_i) = M(\{s_i\} \cup \hat{S}_{k-1}; S^*) - M(\hat{S}_{k-1}; S^*)$$ (7)

where $M(\{s_i\} \cup \hat{S}_{k-1}; S^*)$ and $M(\hat{S}_{k-1}; S^*)$ measure the ROUGE-F1 between the golden summary $S^*$ and the so-far selected sentences $\hat{S}_{k-1}$ with and without the candidate $s_i$ respectively. We rescale the gain $g(s_i)$ to $[0,1]$ in case of negative values using a min-max normalization and get $\tilde{g}$. Then we use a softmax function with a temperature $\tau$ on the rescaled gain to produce a target distribution:

$$Q(s_i) = \frac{\exp(\tau \tilde{g}(s_i))}{\sum_{s_j \in D \setminus \hat{S}_{k-1}} \exp(\tau \tilde{g}(s_j))}$$ (8)

$^2$Emphasizing salience with $o_g$ enhances the performance.
The final training objective is to minimize the KL divergence between the probability distribution of sentence scores (Eq. 6) and their relative rouge gain (Eq. 8), i.e., $KL(P(\cdot)||Q(\cdot))$. This objective can be considered as a listwise ranking loss (Ai et al., 2018) that maximizes the probability of the target sentence while pushing down the probabilities of the other sentences. In this way, AREdSum-SEQ combined the sentence scoring and selection in the same decoder framework, and the redundancy is implicitly captured by optimizing the ROUGE gain.

4.3 AREdSum-CTX: Context-aware Sentence Ranker

We introduce a second model, AREdSum-CTX, a context-aware ranker that scores salience first and then selects a sentence according to both its salience and redundancy adhering to the previously extracted sentences as context, as shown in Figure 1. In AREdSum-CTX, we use a two-step process for scoring and selecting sentences for learning to construct a summary: In the salience ranking step, we focus on learning the salience of the sentences, while in the ranking for sentence selection step, we represent the redundancy explicitly via surface features and use a ranker to decide to promote or demote sentences based on their scores given the joint degree of their salience and redundancy.

Salience Ranking. By assuming that the sentence salience is independent of the previously selected sentences, we design the salience ranking of AREdSum-CTX as a single step process rather than an iterative one. We measure the probability of a sentence to be included in $\hat{S}^*$ using a scoring function $F_{\text{sal}}$ based on the bilinear matching between $h_D$ and $h_{s_i}$, the transformer output after the document-level encoding, same as in Eq. 4.

$$F_{\text{sal}}(s_i) = \frac{\exp h_D W_{ds} h_{s_i}}{\sum_{j=1}^{L} \exp h_D W_{ds} h_{s_j}} \tag{9}$$

The learning objective is to maximize the log likelihood of the summary sentences in the training data:

$$\mathcal{L} = \sum_{s_i \in \hat{S}^*} \log F_{\text{sal}}(s_i) \tag{10}$$

Redundancy Features. In the selection step, we represent redundancy explicitly to let the model focus on learning how to balance salience and redundancy. We extract ngram-matching and semantic-matching features at each $k$-th step to indicate the redundancy of a candidate sentence $s_i$ given the so-far selected sentences, i.e., $\hat{S}_{k-1}$. The ngram-matching feature $f_{n\text{-gram}}$ is computed as:

$$f_{n\text{-gram}} = \frac{|n\text{-gram}(\hat{S}_{k-1}) \cap n\text{-gram}(s_i)|}{n\text{-gram}(s_i)} \tag{11}$$

where $n\text{-gram}(x)$ is the set of $n$ contiguous words in $x$. We collect $f_{n\text{-gram}}$ for $n = \{1, 2, 3\}$. We also compute the semantic-matching feature $f_{\text{sem}}$:

$$f_{\text{sem}} = \max_{\tilde{s}_j \in \hat{S}_{k-1}} \cos(h_{s_i}, h_{\tilde{s}_j}) \tag{12}$$

Since most cosine values between output embeddings from the transformer layers fall in a small range near to 1, we apply a min-max normalization on $f_{\text{sem}}$ to enlarge the value differences and obtain a updated feature $f'$.

The impact of redundancy features on final scores is not linear. Sentences with high redundancy values should be punished more. To capture the effect of the redundancy features at different value sections, we equally divide the range of $[0, 1]$ to $m$ bins and discretize each feature to the corresponding bin according to its value, as shown in Figure 1. In this way, we convert each feature into a one-hot vector of length $m$ and then we concatenate them to obtain a overall redundancy feature vector $F_{\text{red}}(s_i) = [f'_{1\text{-gram}}; f'_{2\text{-gram}}; f'_{3\text{-gram}}; f'_{\text{sem}}]$ where $f'$ represents the one-hot vector after binning $f$.

Ranker for Sentence Selection. In the sentence selection step, AREdSum-CTX only needs to learn how to score a sentence based on its redundancy features $F_{\text{red}}(s_i)$. We use a three-dimensional matrix $W_{F}$ to do a bilinear matching between the redundancy features and salience score and obtain a output matching vector with dimension $d$. Then we apply a single-layer MLP on top to output a final score:

$$f(s_i) = W_F \tanh(F_{\text{sal}}(s_i) W_{F} F_{\text{red}}(s_i)) \tag{13}$$

During training, we randomly select $l$ sentences from the extracted ground-truth set $\hat{S}^*$ as the context and let the model learn to find the next sentence that is both salient and novel, where $l$ is the maximum number of sentences to be included in the predicted summary. The training objective is the same as in § 4.2 except that $o(s_i)$ in Eq. 6 is replaced with $f(s_i)$ in Eq. 13. In contrast to
\textbf{ARED\textsc{SUM}-SEQ} where the target output is an ordered sequence, the loss of \textbf{ARED\textsc{SUM}-CTX} is not order-sensitive since the goal is always to predict the next best sentence given a set of unordered selected sentences as context.

5 Experimental Setup

5.1 Datasets

We evaluate our model on two standard extractive summarization datasets, namely CNN/DailyMail \cite{Hermann2015} and NewYork Times (NYT) \cite{Sandhaus2008}.

\textbf{CNN/DailyMail} contains news articles associated with a few bullet points as the article’s highlight. We use the standard splits of \textcite{Hermann2015} which has 287,226 documents for training, 13,368 for validation, and 11,490 for testing. We conduct preprocessing following the same method in \textcite{Liu2019}. Entities are not anonymized in our experiments as in \textcite{Zhou2018}; \textcite{See2017}; \textcite{Zhang2019}; Liu and Lapata (2019). We truncate articles up to 512 tokens. To collect sentence labels for extractive summarization, we use a greedy strategy similar to \textcite{Nallapati2017; Zhang2019}. We label the subset of sentences that can maximize ROUGE scores against the human-generated summary as 1 (sentence to be included in the summary). The remaining ones are labeled as 0.

\textbf{NYT50} is an annotated corpus of the New York Times. Following \textcite{Paulus2016} and \textcite{Dur2015}, we discard marks and words such as “(s)” and “photo” at the end of the abstract and filter out the articles with summaries shorter than 50. We sort the articles chronologically and split the data into training/validation/test sets according to the ratio of 0.8/0.1/0.1, yielding 133,602/16,700/16,700 documents, respectively. We following the same remaining steps for preprocessing and extractive label collection as the CNN/DailyMail.

5.2 Implementation Details

Our implementation is based on PyTorch and BERTSum\cite{Liu2019}. \textcite{Hermann2016} use “bert-base-uncased” version of BERT to do sentence-level encoding. We fine-tune our models using the objective functions in § 4. We set the number of document-level transformer layers to 2. The dropout rate in all layers is 0.1. We search the best value of $\tau$ in Eq. 8 in \{10, 20, 40, 60\}. We train our models using the Adam optimizer with $\beta_1 = 0.9$, and $\beta_2 = 0.999$ for 2 epochs. We schedule the learning rate according to \textcite{Vaswani2017} with initial value $2e^{-3}$ and 10,000 warm-up steps.

We use teacher-forcing to train \textbf{ARED\textsc{SUM}-SEQ}. To learn the $k$-th sentence in the target sequence, we replace the first $k − 1$ input sentences with other random sentences in the document with the probability of 0.2.\textsuperscript{6} We hypothesize that if the previously selected sentence is not always the golden (right) sentence, it can improve the model’s robustness during training. We use two transformer layers in the decoder.

For \textbf{ARED\textsc{SUM}-CTX}, we train the salience ranker using the same settings as in \textcite{Liu2019}, and all the parameters in the salience ranker are fixed when we train the ranker for selection. This ensures that the salience score of each sentence stays the same during sentence selection. We select the optimum size of the bins for discretized redundancy features by sweeping the values in \{10, 20, 30\} and the size of the output dimension $d$ of $W_r$ in Eq. 13 using \{5, 10, 20, 30\}.

5.3 Baselines

We compare our models to the state-of-the-art extractive summarization model \textsc{BertSumExt} \cite{Liu2019}, which uses Trigram Blocking (\textsc{TriBLK}) \cite{Paulus2017} to filter out sentences with trigram overlap with previously extracted sentences. We report the performance of \textsc{BertSumExt} with and without \textsc{TriBLK} separately to show the impact of this heuristic.

We also compare against other baselines including: \textsc{Lead3}, \textsc{NN-SE} \cite{Cheng2016}, \textsc{SummaRunner} \cite{Nallapati2017}, \textsc{Seq2Seq} \cite{Kedzie2018}, \textsc{NeuSum} \cite{Zhou2018}, and \textsc{HiBERT} \cite{Zhang2019} along with \textsc{Oracle} for upper bound on performance. \textsc{Lead3} is a commonly used effective baseline that extracts the first 3 sentences in the document. \textsc{NN-SE} \cite{Cheng2016} and \textsc{SummaRunner} \cite{Nallapati2017} both formulate extractive summarization as a sequence labelling task. \textsc{NN-SE} uses unidirectional GRU for both the encoding and decoding processes. \textsc{SummaRunner} encodes

\textsuperscript{3}https://github.com/kepingbi/AREDSumSentRank
\textsuperscript{4}https://github.com/nlpyang/BertSum
\textsuperscript{5}https://git.io/fhbJQ
\textsuperscript{6}We found this number after light parameter value sweep.
sentences with BiGRU and considers salience, redundancy, absolute and relative positions of sentences during scoring. Seq2Seq (Kedzie et al., 2018) conducts binary classification by encoding the sentences with a bidirectional GRU (BiGRU) and using a separate decoder BiGRU to transform each sentence as a query vector that attends to the encoder output. NEUSUM (Zhou et al., 2018) learns to jointly score and select sentences using a sequence-to-sequence model to optimize the marginal ROUGE gain and reduce redundancy implicitly. HiBERT (Zhang et al., 2019) pre-trains a hierarchical BERT for extractive summarization without dealing with redundancy. Among these methods, NEUSUM implicitly reduces redundancy by jointly scoring and selecting sentences with a sequence generation model. SUMMARUNNER considers redundancy during the sequence labeling. The other baselines do not conduct redundancy removal.

6 Results and Discussion

6.1 Automatic Evaluation Results

Following earlier work (Zhou et al., 2018; Liu and Lapata, 2019), we include 3 sentences as the summaries for each system for a fair comparison. We evaluate the full-length ROUGE-F1 (Lin, 2004) of the extracted summaries and report ROUGE-1, ROUGE-2 and ROUGE-L which indicates the unigrams, bigrams overlap and longest common subsequence against human edited summaries. The full-length ROUGE-F1 (Lin, 2004) scores of the extracted summaries are evaluated using the official Perl script\(^7\) for both CNN/DailyMail and NYT50. The results of NEUSUM and HiBERT are taken from their original papers while we obtained the rest of the results by re-running the models. Since in the previous work (Liu and Lapata, 2019; Zhang et al., 2019; Paulus et al., 2017) there are no consistent ways of pre-processing the NYT dataset for extractive summarization, we only report the evaluation results from the models we re-trained on this dataset in Table 2.

CNN/DailyMail. Results shown in Table 1 are all comparable as we use the same non-anonymized version of CNN/DailyMail. For BERTSUMEXT-based methods, we observe that redundancy removal helps improve the ROUGE score compared to BERTSUMEXT. TriBLK has considerably better performances; AREDSUM-SEQ achieves better ROUGE-1/L scores; AREDSUM-CTX significantly outperforms the other redundancy elimination methods. \(^8\) The performance differences of AREDSUM-CTX and TriBLK comes from 30.6% summaries output by the two systems in the test set. In other cases, they agrees with each other. This shows that by adaptively balancing salience and diversity, AREDSUM-CTX is superior to TriBLK when redundancy removal is promising.

We also find that the sequence generation models, i.e., NEUSUM and our AREDSUM-SEQ, do not have clear advantage over other models regardless of their encoder network structure (i.e., BERT or other neural architectures). For instance, SUMMARUNNER and SEQ2SEQ models have the best performance among methods that are not based on BERT\(^9\). Our AREDSUM-SEQ perform similarly to BERTSUMEXT. AREDSUM-SEQ is inferior to AREDSUM-CTX due to its order-sensitive optimization objective. While AREDSUM-CTX learns to optimize towards all the possible ordering of the ground truth sentence set, AREDSUM-SEQ is optimized towards only one sequence of them. Another ordering of the same set will be penalized by AREDSUM-SEQ even though they have the same ROUGE score. Its significant worse \(P@1\) (shown in Section 6.2) also confirms this point.

<table>
<thead>
<tr>
<th>Model</th>
<th>RG-1</th>
<th>RG-2</th>
<th>RG-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE</td>
<td>52.59</td>
<td>31.24</td>
<td>48.87</td>
</tr>
<tr>
<td>ORACLE+TriBLK</td>
<td>51.65</td>
<td>30.50</td>
<td>47.89</td>
</tr>
<tr>
<td>LEAD3</td>
<td>40.42</td>
<td>17.62</td>
<td>36.67</td>
</tr>
<tr>
<td>NN-SE</td>
<td>40.81</td>
<td>17.91</td>
<td>37.03</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>41.83</td>
<td>19.29</td>
<td>38.28</td>
</tr>
<tr>
<td>SUMMARUNNER</td>
<td>41.84</td>
<td>19.31</td>
<td>38.31</td>
</tr>
<tr>
<td>NEUSUM</td>
<td>41.59</td>
<td>19.01</td>
<td>37.98</td>
</tr>
<tr>
<td>HiBERT</td>
<td>42.37</td>
<td>19.95</td>
<td>38.83</td>
</tr>
<tr>
<td>BERTSUMEXT</td>
<td>42.61</td>
<td>19.99</td>
<td>39.09</td>
</tr>
<tr>
<td>BERTSUMEXT+TriBLK</td>
<td>43.25</td>
<td>20.24</td>
<td>39.63</td>
</tr>
<tr>
<td>AREDSUM-SEQ</td>
<td>42.72(^a)</td>
<td>19.82</td>
<td>39.15</td>
</tr>
<tr>
<td>AREDSUM-CTX</td>
<td>43.43(^{AB})</td>
<td>20.44(^{AB})</td>
<td>39.83(^{AB})</td>
</tr>
</tbody>
</table>

\(^7\)https://github.com/andersjo/pyrouge/tree/master/tools/ROUGE-1.5.5

\(^8\)The salience ranker of AREDSUM-CTX alone performs similarly to BERTSUMEXT.

\(^9\)The ROUGE scores of SUMMARUNNER are lower than NEUSUM in Zhou et al. (2018) because the results of SUMMARUNNER are from the anonymized version CNN/DailyMail, which are not comparable with the results of NEUSUM on the non-anonymized version.
**NYT50.** In contrast to CNN/DailyMail, we observe that TriBLK has the performance of BERTSUMEXT on NYT50. In fact, as shown in Table 2, applying TriBLK on ORACLE also causes reduction in ROUGE-1,2,L scores by 1.91, 1.59 and 1.92 absolute point respectively, which are much larger than those drops in CNN/DailyMail (0.94, 0.74 and 0.98). This indicates that TriBLK filters out more sentences that have high ROUGE gain on NYT50 than CNN/DailyMail, causing more drop of ROUGE. It also shows that sentences in oracle summaries have more trigram overlap on NYT50 than CNN/DailyMail, which implies that redundancy removal on NYT50 may have limited gains and a simple unified rule (TriBLK) applying on all the documents could harm the performance.

We also observe that AREDSUM-SEQ performs better than BERTSUMEXT+TriBLK but worse than BERTSUMEXT. In contrast, AREDSUM-CTX achieves higher performance than BERTSUMEXT+TriBLK and AREDSUM-SEQ by representing redundancy explicitly and controlling its effect dynamically. Since redundancy removal has a limited potential gain on NYT50, the predictions of AREDSUM-CTX differ from BERTSUMEXT only in 10.1% of the test set. However, these differences still lead to significant overall improvements.

Note that the gain of AREDSUM-CTX comes only from redundancy removal, which takes effect from the second step of selecting sentences. The improvements can be larger when redundancy removal has higher potentials (e.g., CNN/DailyMail) and smaller on datasets (e.g., NYT50) with lower potentials. In either case, it does not harm the performances as the other methods, which shows that it is adaptive and robust.

**6.2 Model Analysis**

**Precision at Each Step.** Since the content of sentences with positive labels, i.e., $\hat{S}^*$, could vary from the original human-generated abstractive summaries $S^*$, models that have higher precision with respect to $\hat{S}^*$ do not necessarily yield better ROUGE scores against $S^*$. Because BERTSUMEXT is optimized towards $S^*$ while AREDSUM-CTX and AREDSUM-SEQ aim to learn to select sentences with best ROUGE gain against $S^*$, they behave differently in terms of ROUGE and precision. Thus, we analyze how our model’s selection at each $k$-th step affects the ROUGE performance. We only present the precision on CNN/DailyMail in Figure 2 since similar trends are observed on NYT50. Note all the models except AREDSUM-SEQ have the same $P@1$ because initially, the model’s selection is only based on salience. Filtering and demoting the selected sentences starts to take effect only after the second step.

As shown in the figure, BERTSUMExt has the best $P@1$, $P@2$ and $P@3$ among all, which is reasonable since $\hat{S}^*$ is the target which it is optimized to learn. When TriBLK is applied, $P@2$ and $P@3$ drop a lot while the ROUGE scores are up (as in Table 1). It indicates that TriBLK could filter out some informative but redundant sentences during selection, which harms precision but improves ROUGE. In contrast, $P@2$ and $P@3$ of AREDSUM-CTX is between BERTSUMEXT with and without TriBLK. Through learning towards ROUGE gain given the previously extracted sentences, AREDSUM-CTX achieves the best ROUGE scores with less harm to precision, which means that AREDSUM-CTX can better balance salience and redundancy.
We also conduct human evaluations to analyze how our best model compares against the best baseline model. On both datasets, we randomly sample 20 summaries constructed by the best baseline and our best model from the cases where their ROUGE-2 score difference is more than 0.05 points. Following Zhou et al. (2018), we asked two graduate student volunteers to rank the summaries extracted by different models from best to worst in terms of informativeness, redundancy and the overall quality. We allowed ties in the analysis. Average ranks of the systems are shown in Table 3.

On CNN/DailyMail, AREDSum-CTX ranks higher than BERTSumEXT+TriBLK in terms of each aspect. On NYT50, AREDSum-CTX has a more compelling performance in terms of redundancy than informativeness. This is consistent with the fact that BERTSumEXT only focuses on learning salience and does not deal with redundancy during sentence selection. From both automatic and human evaluation of our best model and the best baseline, we can see that removing redundancy with our models is better than redundancy removal with heuristics and no redundancy removal.

7 Conclusions

Extending a state-of-the-art extractive summarization model, we propose AREDSum-SEQ that jointly scores and selects sentences with a sequence generation model and AREDSum-CTX that learns to balance salience and redundancy with a separate model. Experimental results show that AREDSum-CTX outperforms AREDSum-SEQ and all other strong baselines, which yields that redundancy reduction helps improve summary quality, and it is better to model the effect of redundancy explicitly than jointly with salience during sentence scoring.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.
References


