Cutting-off Redundant Repeating Generations for Neural Abstractive Summarization

Jun Suzuki and **Masaaki Nagata**
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto, 619-0237 Japan
{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

Abstract

This paper tackles the reduction of redundant repeating generation that is often observed in RNN-based encoder-decoder models. Our basic idea is to jointly estimate the upper-bound frequency of each target vocabulary in the encoder and control the output words based on the estimation in the decoder. Our method shows significant improvement over a strong RNN-based encoder-decoder baseline and achieved its best results on an abstractive summarization benchmark.

1 Introduction

The RNN-based encoder-decoder (EncDec) approach has recently been providing significant progress in various natural language generation (NLG) tasks, *i.e.*, machine translation (MT) (Sutskever et al., 2014; Cho et al., 2014) and abstractive summarization (ABS) (Rush et al., 2015). Since a scheme in this approach can be interpreted as a conditional language model, it is suitable for NLG tasks. However, one potential weakness is that it sometimes repeatedly generates the same phrase (or word).

This issue has been discussed in the neural MT (NMT) literature as a part of a *coverage problem* (Tu et al., 2016; Mi et al., 2016). Such repeating generation behavior can become more severe in some NLG tasks than in MT. The *very short* ABS task in DUC-2003 and 2004 (Over et al., 2007) is a typical example because it requires the generation of a summary in a pre-defined limited output space, such as ten words or 75 bytes. Thus, the repeated output consumes precious limited output space. Unfortunately, the coverage approach cannot be directly applied to ABS tasks since they require us to optimally find salient ideas from the input in a *lossy compression* manner, and thus the summary (output) length hardly depends on the input length; an MT task is mainly *loss-less* generation and nearly one-to-one correspondence between input and output (Nallapati et al., 2016a).

From this background, this paper tackles this issue and proposes a method to overcome it in ABS tasks. The basic idea of our method is to jointly estimate the upper-bound frequency of each target vocabulary that can occur in a summary during the encoding process and exploit the estimation to control the output words in each decoding step. We refer to our additional component as a **word-frequency estimation (WFE) sub-model**. The WFE sub-model explicitly manages how many times each word has been generated so far and might be generated in the future during the decoding process. Thus, we expect to decisively prohibit excessive generation. Finally, we evaluate the effectiveness of our method on well-studied ABS benchmark data provided by Rush et al. (2015), and evaluated in (Chopra et al., 2016; Nallapati et al., 2016b; Kikuchi et al., 2016; Takase et al., 2016; Ayana et al., 2016; Gulcehre et al., 2016).

2 Baseline RNN-based EncDec Model

The baseline of our proposal is an RNN-based EncDec model with an attention mechanism (Luong et al., 2015). In fact, this model has already been used as a strong baseline for ABS tasks (Chopra et al., 2016; Kikuchi et al., 2016) as well as in the NMT literature. More specifically, as a case study we employ a *2-layer bidirectional LSTM* encoder and a *2-layer LSTM* decoder with a global attention (Bahdanau et al., 2014). We omit a detailed review of the descriptions due to space limitations. The following are the necessary parts for explaining our proposed method.

Let $\boldsymbol{X} = (\boldsymbol{x}_i)_{i=1}^{I}$ and $\boldsymbol{Y} = (\boldsymbol{y}_j)_{j=1}^{J}$ be input and output sequences, respectively, where $\boldsymbol{x}_i$ and

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 291–297,
Valencia, Spain, April 3-7, 2017.

Input: $\boldsymbol{H}^{\mathrm{s}} = (\boldsymbol{h}_i^{\mathrm{s}})_{i=1}^{I}$ ▷ list of hidden states generated by encoder
Initialize: $s \leftarrow 0$ ▷ s: cumulative log-likelihood
$\hat{\boldsymbol{Y}} \leftarrow$ 'BOS' ▷ $\hat{\boldsymbol{Y}}$: list of generated words
$\boldsymbol{H}^{\mathrm{t}} \leftarrow \boldsymbol{H}^{\mathrm{s}}$ ▷ $\boldsymbol{H}^{\mathrm{t}}$: hidden states to process decoder
1: $h \leftarrow (s, \hat{\boldsymbol{Y}}, \boldsymbol{H}^{\mathrm{t}})$ ▷ triplet of (minimal) info for decoding process
2: $\mathcal{Q}_{\mathtt{w}} \leftarrow \mathtt{push}(\mathcal{Q}_{\mathtt{w}}, h)$ ▷ set initial triplet h to priority queue $\mathcal{Q}_{\mathtt{w}}$
3: $\mathcal{Q}_{\mathtt{c}} \leftarrow \{\}$ ▷ prepare queue to store complete sentences
4: **Repeat**
5: $\tilde{\boldsymbol{O}} \leftarrow ()$ ▷ prepare empty list
6: **Repeat**
7: $h \leftarrow \mathtt{pop}(\mathcal{Q}_{\mathtt{w}})$ ▷ pop a candidate history
8: $\tilde{\boldsymbol{o}} \leftarrow \mathtt{calcLL}(h)$ ▷ see Eq. 2
9: $\tilde{\boldsymbol{O}} \leftarrow \mathtt{append}(\tilde{\boldsymbol{O}}, \tilde{\boldsymbol{o}})$ ▷ append likelihood vector
10: **Until** $\mathcal{Q}_{\mathtt{w}} = \emptyset$ ▷ repeat until $\mathcal{Q}_{\mathtt{w}}$ is empty
11: $\{(\hat{m}, \hat{k})_z\}_{z=1}^{K-C} \leftarrow \mathtt{findKBest}(\tilde{\boldsymbol{O}})$
12: $\{h_z\}_{z=1}^{K-C} \leftarrow \mathtt{makeTriplet}(\{(\hat{m}, \hat{k})_z\}_{z=1}^{K-C})$
13: $\mathcal{Q}' \leftarrow \mathtt{selectTopK}(\mathcal{Q}_{\mathtt{c}}, \{h_z\}_{z=1}^{K-C})$
14: $(\mathcal{Q}_{\mathtt{w}}, \mathcal{Q}_{\mathtt{c}}) \leftarrow \mathtt{SepComp}(\mathcal{Q}')$ ▷ separate $\mathcal{Q}'$ into $\mathcal{Q}_{\mathtt{c}}$ or $\mathcal{Q}_{\mathtt{w}}$
15: **Until** $\mathcal{Q}_{\mathtt{w}} = \emptyset$ ▷ finish if $\mathcal{Q}_{\mathtt{w}}$ is empty
Output: $\mathcal{Q}_{\mathtt{c}}$

Figure 1: Algorithm for a K-best beam search decoding typically used in EncDec approach.

$\boldsymbol{y}_j$ are one-hot vectors, which correspond to the i-th word in the input and the j-th word in the output. Let $\mathcal{V}^{\mathrm{t}}$ denote the vocabulary (set of words) of output. For simplification, this paper uses the following four notation rules:
(1) $(\boldsymbol{x}_i)_{i=1}^{I}$ is a short notation for representing a list of (column) vectors, *i.e.*, $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_I) = (\boldsymbol{x}_i)_{i=1}^{I}$.
(2) $\boldsymbol{v}(a, D)$ represents a D-dimensional (column) vector whose elements are all a, *i.e.*, $\boldsymbol{v}(1, 3) = (1, 1, 1)^{\top}$.
(3) $\boldsymbol{x}[i]$ represents the i-th element of $\boldsymbol{x}$, *i.e.*, $\boldsymbol{x} = (0.1, 0.2, 0.3)^{\top}$, then $\boldsymbol{x}[2] = 0.2$.
(4) $M = |\mathcal{V}^t|$ and, m always denotes the index of output vocabulary, namely, $m \in \{1, \ldots, M\}$, and $\boldsymbol{o}[m]$ represents the score of the m-th word in $\mathcal{V}^t$, where $\boldsymbol{o} \in \mathbb{R}^M$.

Encoder: Let $\Omega^{\mathrm{s}}(\cdot)$ denote the overall process of our 2-layer bidirectional LSTM encoder. The encoder receives input $\boldsymbol{X}$ and returns a list of final hidden states $\boldsymbol{H}^{\mathrm{s}} = (\boldsymbol{h}_i^{\mathrm{s}})_{i=1}^{I}$:

$$\boldsymbol{H}^{\mathrm{s}} = \Omega^{\mathrm{s}}(\boldsymbol{X}). \tag{1}$$

Decoder: We employ a K-best beam-search decoder to find the (approximated) best output $\hat{\boldsymbol{Y}}$ given input $\boldsymbol{X}$. Figure 1 shows a typical K-best beam search algorithm used in the decoder of EncDec approach. We define the (minimal) required information h shown in Figure 1 for the j-th decoding process is the following triplet, $h = (s_{j-1}, \hat{\boldsymbol{Y}}_{j-1}, \boldsymbol{H}_{j-1}^{\mathrm{t}})$, where s_{j-1} is the cumulative log-likelihood from step 0 to $j-1$, $\hat{\boldsymbol{Y}}_{j-1}$ is a (candidate of) output word sequence generated so far from step 0 to $j-1$, that is, $\hat{\boldsymbol{Y}}_{j-1} = (\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{j-1})$ and $\boldsymbol{H}_{j-1}^{\mathrm{t}}$ is the all the hidden states for calculating the j-th decoding process. Then, the function `calcLL` in Line 8 can be written as follows:

$$\begin{aligned} \tilde{\boldsymbol{o}}_j &= \boldsymbol{v}\big(s_{j-1}, M\big) + \log\big(\mathtt{Softmax}(\boldsymbol{o}_j)\big) \\ \boldsymbol{o}_j &= \Omega^{\mathrm{t}}\big(\boldsymbol{H}^{\mathrm{s}}, \boldsymbol{H}_{j-1}^{\mathrm{t}}, \hat{\boldsymbol{y}}_{j-1}\big), \end{aligned} \tag{2}$$

where $\mathtt{Softmax}(\cdot)$ is the *softmax* function for a given vector and $\Omega^{\mathrm{t}}(\cdot)$ represents the overall process of a single decoding step.

Moreover, $\tilde{\boldsymbol{O}}$ in Line 11 is a $(M \times (K - C))$-matrix, where C is the number of complete sentences in $\mathcal{Q}_{\mathtt{c}}$. The (m, k)-element of $\tilde{\boldsymbol{O}}$ represents a likelihood of the m-th word, namely $\tilde{\boldsymbol{o}}_j[m]$, that is calculated using the k-th candidate in $\mathcal{Q}_{\mathtt{w}}$ at the $(j-1)$-th step. In Line 12, the function `makeTriplet` constructs a set of triplets based on the information of index $(\hat{m}, \hat{k})$. Then, in Line 13, the function `selectTopK` selects the top-K candidates from union of a set of generated triplets at current step $\{h_z\}_{z=1}^{K-C}$and a set of triplets of complete sentences in $\mathcal{Q}_{\mathtt{c}}$. Finally, the function `sepComp` in Line 13 divides a set of triplets $\mathcal{Q}'$ in two distinct sets whether they are complete sentences, $\mathcal{Q}_{\mathtt{c}}$, or not, $\mathcal{Q}_{\mathtt{w}}$. If the elements in $\mathcal{Q}'$ are all complete sentences, namely, $\mathcal{Q}_{\mathtt{c}} = \mathcal{Q}'$ and $\mathcal{Q}_{\mathtt{w}} = \emptyset$, then the algorithm stops according to the evaluation of Line 15.

3 Word Frequency Estimation

This section describes our proposed method, which roughly consists of two parts: (1) a sub-model that estimates the upper-bound frequencies of the target vocabulary words in the output, and (2) architecture for controlling the output words in the decoder using estimations.

3.1 Definition

Let $\hat{\boldsymbol{a}}$ denote a vector representation of the frequency estimation. $\odot$ denotes element-wise product. $\hat{\boldsymbol{a}}$ is calculated by:

$$\begin{aligned} \hat{\boldsymbol{a}} &= \hat{\boldsymbol{r}} \odot \hat{\boldsymbol{g}} \\ \hat{\boldsymbol{r}} &= \mathtt{ReLU}(\boldsymbol{r}), \quad \hat{\boldsymbol{g}} = \mathtt{Sigmoid}(\boldsymbol{g}), \end{aligned} \tag{3}$$

where $\mathtt{Sigmoid}(\cdot)$ and $\mathtt{ReLu}(\cdot)$ represent the element-wise sigmoid and ReLU (Glorot et al., 2011), respectively. Thus, $\hat{\boldsymbol{r}} \in [0, +\infty]^M$, $\hat{\boldsymbol{g}} \in [0, 1]^M$, and $\hat{\boldsymbol{a}} \in [0, +\infty]^M$.

We incorporate two separated components, $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{g}}$, to improve the frequency fitting. The purpose of $\hat{\boldsymbol{g}}$ is to distinguish whether the target words occur or not, regardless of their frequency. Thus, $\hat{\boldsymbol{g}}$ can be interpreted as a *gate* function that resembles estimating the fertility in the coverage (Tu et al., 2016) and a switch probability in the copy mechanism (Gulcehre et al., 2016). These ideas originated from such gated recurrent networks as LSTM (Hochreiter and Schmidhuber, 1997) and GRU (Chung et al., 2014). Then, $\hat{\boldsymbol{r}}$ can much focus on to model frequency equal to or larger than 1. This separation can be expected since $\hat{\boldsymbol{r}}[m]$ has no influence if $\hat{\boldsymbol{g}}[m]=0$.

3.2 Effective usage

The technical challenge of our method is effectively leveraging WFE $\hat{\boldsymbol{a}}$. Among several possible choices, we selected to integrate it as prior knowledge in the decoder. To do so, we re-define $\tilde{\boldsymbol{o}}_j$ in Eq. 2 as:

$$\tilde{\boldsymbol{o}}_j = \boldsymbol{v}\big(s_{j-1}, M\big) + \log\left(\texttt{Softmax}(\boldsymbol{o}_j)\right) + \tilde{\boldsymbol{a}}_j.$$

The difference is the additional term of $\tilde{\boldsymbol{a}}_j$, which is an *adjusted* likelihood for the j-th step originally calculated from $\hat{\boldsymbol{a}}$. We define $\tilde{\boldsymbol{a}}_j$ as:

$$\tilde{\boldsymbol{a}}_j = \log\left(\texttt{ClipReLU}_1(\tilde{\boldsymbol{r}}_j) \odot \hat{\boldsymbol{g}}\right). \qquad (4)$$

$\texttt{ClipReLU}_1(\cdot)$ is a function that receives a vector and performs an element-wise calculation: $\boldsymbol{x}'[m] = \max\left(0, \min(1, \boldsymbol{x}[m])\right)$ for all m if it receives $\boldsymbol{x}$. We define the relation between $\tilde{\boldsymbol{r}}_j$ in Eq. 4 and $\hat{\boldsymbol{r}}$ in Eq. 3 as follows:

$$\tilde{\boldsymbol{r}}_j = \begin{cases} \hat{\boldsymbol{r}} & \text{if } j = 1 \\ \tilde{\boldsymbol{r}}_{j-1} - \hat{\boldsymbol{y}}_{j-1} & \text{otherwise} \end{cases}. \qquad (5)$$

Eq. 5 is updated from $\tilde{\boldsymbol{r}}_{j-1}$ to $\tilde{\boldsymbol{r}}_j$ with the estimated output of previous step $\hat{\boldsymbol{y}}_{j-1}$. Since $\hat{\boldsymbol{y}}_j \in \{0,1\}^M$ for all j, all of the elements in $\tilde{\boldsymbol{r}}_j$ are monotonically non-increasing. If $\tilde{\boldsymbol{r}}_{j'}[m] \leq 0$ at j', then $\tilde{\boldsymbol{o}}_{j'}[m] = -\infty$ regardless of $\boldsymbol{o}[m]$. This means that the m-th word will never be selected any more at step $j' \leq j$ for all j. Thus, the interpretation of $\tilde{\boldsymbol{r}}_j$ is that it directly manages the upper-bound frequency of each target word that can occur in the current and future decoding time steps. As a result, decoding with our method never generates words that exceed the estimation $\hat{\boldsymbol{r}}$, and thus we expect to reduce the redundant repeating generation.

Note here that our method never requires $\tilde{\boldsymbol{r}}_j[m] \leq 0$ (or $\tilde{\boldsymbol{r}}_j[m] = 0$) for all m at the last decoding time step j, as is generally required in the coverage (Tu et al., 2016; Mi et al., 2016; Wu et al., 2016). This is why we say *upper-bound* frequency estimation, not just *(exact) frequency*.

Input: $\boldsymbol{H}^{\mathrm{s}} = (\boldsymbol{h}^{\mathrm{s}}_i)_{i=1}^{I}$ ▷ list of hidden states generated by encoder
Parameters: $\boldsymbol{W}_1^r, \boldsymbol{W}_1^g \in \mathbb{R}^{H\times H}$, $\boldsymbol{W}_2^r \in \mathbb{R}^{M\times H}$, $\boldsymbol{W}_2^g \in \mathbb{R}^{M\times 2H}$,
1: $\boldsymbol{H}_1^r \leftarrow \boldsymbol{W}_1^r \boldsymbol{H}^{\mathrm{s}}$ ▷ linear transformation for frequency model
2: $\boldsymbol{h}_1^r \leftarrow \boldsymbol{H}_1^r \boldsymbol{v}(1, M)$ ▷ $\boldsymbol{h}_1^r \in \mathbb{R}^H$, $\boldsymbol{H}_1^r \in \mathbb{R}^{H\times I}$
3: $\boldsymbol{r} \leftarrow \boldsymbol{W}_2^r \boldsymbol{h}_1^r$ ▷ frequency estimation
4: $\boldsymbol{H}_1^g \leftarrow \boldsymbol{W}_1^g \boldsymbol{H}^{\mathrm{s}}$ ▷ linear transformation for occurrence model
5: $\boldsymbol{h}_2^{g+} \leftarrow \texttt{RowMax}(\boldsymbol{H}_1^g)$ ▷ $\boldsymbol{h}_2^{g+} \in \mathbb{R}^H$, and $\boldsymbol{H}_1^g \in \mathbb{R}^{H\times I}$
6: $\boldsymbol{h}_2^{g-} \leftarrow \texttt{RowMin}(\boldsymbol{H}_1^g)$ ▷ $\boldsymbol{h}_2^{g-} \in \mathbb{R}^H$, and $\boldsymbol{H}_1^g \in \mathbb{R}^{H\times I}$
7: $\boldsymbol{g} \leftarrow \boldsymbol{W}_2^g\big(\texttt{concat}(\boldsymbol{h}_2^{g+}, \boldsymbol{h}_2^{g-})\big)$ ▷ occurrence estimation
Output: $(\boldsymbol{g}, \boldsymbol{r})$

Figure 2: Procedure for calculating the components of our WFE sub-model.

3.3 Calculation

Figure 2 shows the detailed procedure for calculating $\boldsymbol{g}$ and $\boldsymbol{r}$ in Eq. 3. For $\boldsymbol{r}$, we sum up all of the features of the input given by the encoder (Line 2) and estimate the frequency. In contrast, for $\boldsymbol{g}$, we expect Lines 5 and 6 to work as a kind of *voting* for both positive and negative directions since $\boldsymbol{g}$ needs just *occurrence* information, not *frequency*. For example, $\boldsymbol{g}$ may take large positive or negative values if a certain input word (feature) has a strong influence for occurring or not occurring specific target word(s) in the output. This idea is borrowed from the Max-pooling layer (Goodfellow et al., 2013).

3.4 Parameter estimation (Training)

Given the training data, let $\boldsymbol{a}^* \in \mathbb{P}^M$ be a vector representation of the true frequency of the target words given the input, where $\mathbb{P} = \{0, 1, \ldots, +\infty\}$. Clearly $\boldsymbol{a}^*$ can be obtained by counting the words in the corresponding output. We define loss function Ψ^{wfe} for estimating our WFE sub-model as follows:

$$\Psi^{\mathrm{wfe}}\big(\boldsymbol{X}, \boldsymbol{a}^*, \mathcal{W}\big) = \boldsymbol{d} \cdot \boldsymbol{v}(1, M) \qquad (6)$$
$$\boldsymbol{d} = c_1 \max\left(\boldsymbol{v}(0, M), \hat{\boldsymbol{a}} - \boldsymbol{a}^* - \boldsymbol{v}(\epsilon, M)\right)^b + c_2 \max\left(\boldsymbol{v}(0, M), \boldsymbol{a}^* - \hat{\boldsymbol{a}} - \boldsymbol{v}(\epsilon, M)\right)^b,$$

where $\mathcal{W}$ represents the overall parameters. The form of $\Psi^{\mathrm{wfe}}(\cdot)$ is closely related to that used in support vector regression (SVR) (Smola and Schölkopf, 2004). We allow estimation $\hat{\boldsymbol{a}}[m]$ for all m to take a value in the range of $[\boldsymbol{a}^*[m] - \epsilon, \boldsymbol{a}^*[m] + \epsilon]$ with no penalty (the loss is zero). In our case, we select $\epsilon = 0.25$ since all the elements

Source vocabulary	† 119,507
Target vocabulary	† 68,887
Dim. of embedding D	200
Dim. of hidden state H	400
Encoder RNN unit	2-layer bi-LSTM
Decoder RNN unit	2-layer LSTM with attention
Optimizer	Adam (first 5 epoch) + SGD (remaining epoch) ⋆
Initial learning rate	0.001 (Adam) / 0.01 (SGD)
Mini batch size	256 (shuffled at each epoch)
Gradient clipping	10 (Adam) / 5 (SGD)
Stopping criterion	max 15 epoch w/ early stopping based on the val. set
Other opt. options	Dropout = 0.3

Table 1: Model and optimization configurations in our experiments. †: including special BOS, EOS, and UNK symbols. ⋆: as suggested in (Wu et al., 2016)

of $\boldsymbol{a}^*$ are an integer. The remaining 0.25 for both the positive and negative sides denotes the *margin* between every integer. We select $b = 2$ to penalize larger for more distant error, and $c_1 < c_2$, *i.e.*, $c_1 = 0.2, c_2 = 1$, since we aim to obtain upper-bound estimation and to penalize the underestimation below the true frequency $\boldsymbol{a}^*$.

Finally, we minimize Eq. 6 with a standard negative log-likelihood objective function to estimate the baseline EncDec model.

4 Experiments

We investigated the effectiveness of our method on ABS experiments, which were first performed by Rush et al., (2015). The data consist of approximately 3.8 million training, 400,000 validation and 400,000 test data, respectively[2]. Generally, 1951 test data, randomly extracted from the test data section, are used for evaluation[3]. Additionally, DUC-2004 evaluation data (Over et al., 2007)[4] were also evaluated by the identical models trained on the above Gigaword data. We strictly followed the instructions of the evaluation setting used in previous studies for a fair comparison. Table 1 summarizes the model configuration and the parameter estimation setting in our experiments.

4.1 Main results: comparison with baseline

Table 2 shows the results of the baseline EncDec and our proposed EncDec+WFE. Note that the

[2]The data can be created by the data construction scripts in the author's code: https://github.com/facebook/NAMAS.

[3]As previously described (Chopra et al., 2016) we removed the ill-formed (empty) data for Gigaword.

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

G: china success at youth world championship shows preparation for #### olympics
A: china germany germany germany germany and germany at world youth championship
B: china faces germany at world youth championship

G: British and Spanish governments leave extradition of Pinochet to courts
A: spain britain seek shelter from pinochet 's pinochet case over pinochet 's
B: spain britain seek shelter over pinochet 's possible extradition from spain

G: torn UNK : plum island juniper duo now just a lone tree
A: black women black women black in black code
B: in plum island of the ancient

Figure 3: Examples of generated summary. G: reference summary, A: baseline EncDec, and B: EncDec+WFE. (underlines indicate repeating phrases and words)

DUC-2004 data was evaluated by recall-based ROUGE scores, while the Gigaword data was evaluated by F-score-based ROUGE, respectively. For a validity confirmation of our EncDec baseline, we also performed `OpenNMT` tool[5]. The results on Gigaword data with $B = 5$ were, 33.65, 16.12, and 31.37 for ROUGE-1(F), ROUGE-2(F) and ROUGE-L(F), respectively, which were almost similar results (but slightly lower) with our implementation. This supports that our baseline worked well as a strong baseline. Clearly, EncDec+WFE significantly outperformed the strong EncDec baseline by a wide margin on the ROUGE scores. Thus, we conclude that the WFE sub-model has a positive impact to gain the ABS performance since performance gains were derived only by the effect of incorporating our WFE sub-model.

4.2 Comparison to current top systems

Table 3 lists the current top system results. Our method EncDec+WFE successfully achieved the current best scores on most evaluations. This result also supports the effectiveness of incorporating our WFE sub-model.

MRT (Ayana et al., 2016) previously provided the best results. Note that its model structure is nearly identical to our baseline. On the contrary, MRT trained a model with a sequence-wise min-

[5]http://opennmt.net

Method	Beam	DUC-2004 (w/ 75-byte limit) ROUGE-1(R)	ROUGE-2(R)	ROUGE-L(R)	Gigaword (w/o length limit) ROUGE-1(F)	ROUGE-2(F)	ROUGE-L(F)
EncDec	B=1	29.23	8.71	25.27	33.99	16.06	31.63
(baseline)	B=5	29.52	9.45	25.80	†34.27	†16.68	†32.14
our impl.)	B=10	† 29.60	† 9.62	† 25.97	34.18	16.51	31.97
EncDec+WFE	B=1	31.92	9.36	27.22	36.21	16.87	33.55
(proposed)	B=5	⋆**32.28**	⋆**10.54**	⋆**27.80**	⋆**36.30**	⋆**17.31**	⋆**33.88**
	B=10	31.70	10.34	27.48	36.08	17.23	33.73
(perf. gain from † to ⋆)		+2.68	+0.92	+1.83	+2.03	+0.63	+1.78

Table 2: Results on DUC-2004 and Gigaword data: ROUGE-x(R): recall-based ROUGE-x, ROUGE-x(F): F1-based ROUGE-x, where $x \in \{1, 2, L\}$, respectively.

Method	DUC-2004 (w/ 75-byte limit) ROUGE-1(R)	ROUGE-2(R)	ROUGE-L(R)	Gigaword (w/o length limit) ROUGE-1(F)	ROUGE-2(F)	ROUGE-L(F)
ABS (Rush et al., 2015)	26.55	7.06	22.05	30.88	12.22	27.77
RAS (Chopra et al., 2016)	28.97	8.26	24.06	33.78	15.97	31.15
BWL (Nallapati et al., 2016a)[1]	28.35	9.46	24.59	32.67	15.59	30.64
(words-lvt5k-1sent†)	28.61	9.42	25.24	35.30	†16.64	32.62
MRT (Ayana et al., 2016)	†30.41	†**10.87**	†26.79	†**36.54**	16.59	†33.44
EncDec+WFE [This Paper]	**32.28**	10.54	**27.80**	36.30	**17.31**	**33.88**
(perf. gain from †)	+1.87	-0.33	+1.01	-0.24	+0.72	+0.44

Table 3: Results of current top systems: '*': previous best score for each evaluation. †: using a larger vocab for both encoder and decoder, not strictly fair configuration with other results.

True $\boldsymbol{a}^*$ \ Estimation $\hat{\boldsymbol{a}}$	0	1	2	3	4 $\geq$
1	7,014	7,064	1,784	16	4
2	51	95	60	0	0
3 $\geq$	2	4	1	0	0

Table 4: Confusion matrix of WFE on Gigaword data: only evaluated true frequency ≥ 1.

imum risk estimation, while we trained all the models in our experiments with standard (pointwise) log-likelihood maximization. MRT essentially complements our method. We expect to further improve its performance by applying MRT for its training since recent progress of NMT has suggested leveraging a sequence-wise optimization technique for improving performance (Wiseman and Rush, 2016; Shen et al., 2016). We leave this as our future work.

4.3 Generation examples

Figure 3 shows actual generation examples. Based on our motivation, we specifically selected the redundant repeating output that occurred in the baseline EncDec. It is clear that EncDec+WFE successfully reduced them. This observation offers further evidence of the effectiveness of our method in quality.

4.4 Performance of the WFE sub-model

To evaluate the WFE sub-model alone, Table 4 shows the confusion matrix of the frequency estimation. We quantized $\hat{\boldsymbol{a}}$ by $\lfloor \hat{\boldsymbol{a}}[m] + 0.5 \rfloor$ for all m, where 0.5 was derived from the margin in Ψ^{wfe}. Unfortunately, the result looks not so well. There seems to exist an enough room to improve the estimation. However, we emphasize that it already has an enough power to improve the overall quality as shown in Table 2 and Figure 3. We can expect to further gain the overall performance by improving the performance of the WFE sub-model.

5 Conclusion

This paper discussed the behavior of redundant repeating generation often observed in neural EncDec approaches. We proposed a method for reducing such redundancy by incorporating a submodel that directly estimates and manages the frequency of each target vocabulary in the output. Experiments on ABS benchmark data showed the effectiveness of our method, EncDec+WFE, for both improving automatic evaluation performance and reducing the actual redundancy. Our method is suitable for *lossy compression* tasks such as image caption generation tasks.

Acknowledgement

We thank three anonymous reviewers for their helpful comments.

References

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. *CoRR*, abs/1604.01904.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–98, San Diego, California, June. Association for Computational Linguistics.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR*, abs/1412.3555.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks. In Geoffrey J. Gordon and David B. Dunson, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, volume 15, pages 315–323. Journal of Machine Learning Research - Workshop and Conference Proceedings.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. 2013. Maxout Networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, pages 1319–1327.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 140–149, Berlin, Germany, August. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338, Austin, Texas, November. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal, September. Association for Computational Linguistics.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 955–960, Austin, Texas, November. Association for Computational Linguistics.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization. *CoRR*, abs/1602.06023.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany, August. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in context. *Information Processing and Management*, 43(6):1506–1520.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 379–389.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum risk training for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1683–1692, Berlin, Germany, August. Association for Computational Linguistics.

Alex J Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation.

In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1054–1059, Austin, Texas, November. Association for Computational Linguistics.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling Coverage for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85, Berlin, Germany, August. Association for Computational Linguistics.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1296–1306, Austin, Texas, November. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.