Copy or Rewrite: Hybrid Summarization with Hierarchical Reinforcement Learning

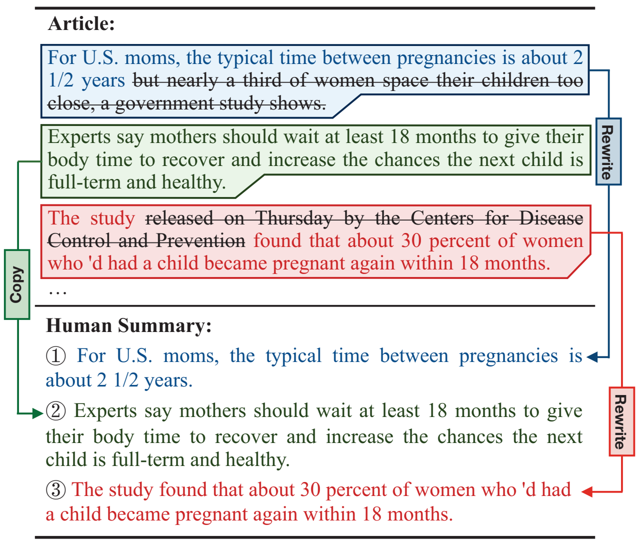
Liqiang Xiao,[1] Lu Wang,[2] Hao He,[1,3] Yaohui Jin[1,3]
[1]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University
[2]Khoury College of Computer Sciences, Northeastern University, Boston, MA 02115
[3]State Key Lab of Advanced Optical Communication System and Network, Shanghai Jiao Tong University
{xiaoliqiang, hehao, jinyh}@sjtu.edu.cn, luwang@ccs.neu.edu

Abstract

Jointly using the extractive and abstractive summarization methods can combine their complementary advantages, generating both informative and concise summary. Existing methods that adopt an extract-then-abstract strategy have achieved impressive results, yet they suffer from the information loss in the abstraction step because they compress all the selected sentences without distinguish. Especially when the whole sentence is summary-worthy, salient content would be lost by compression. To address this problem, we propose HySum, a hybrid framework for summarization that can flexibly switch between copying sentence and rewriting sentence according to the degree of redundancy. In this way, our approach can effectively combine the advantages of two branches of summarization, juggling informativity and conciseness. Moreover, we based on Hierarchical Reinforcement Learning, propose an end-to-end reinforcing method to bridge together the extraction module and rewriting module, which can enhance the cooperation between them. Automatic evaluation shows that our approach significantly outperforms the state-of-the-arts on the CNN/DailyMail corpus. Human evaluation also demonstrates that our generated summaries are more informative and concise than popular models.

1 Introduction

The target of summarization is to rewrite a long article into a short and fluent version while maintaining the most salient content. With the successful application of neural network on natural language processing (NLP) tasks, two data-driven branches, *extractive* and *abstractive* summarization (See, Liu, and Manning, 2017; Vinyals, Fortunato, and Jaitly, 2015; Nallapati, Zhai, and Zhou, 2017), stand out from the various approaches (Jing and McKeown, 2000; Knight and Marcu, 2000). Extractive methods (Dong et al., 2018; Nallapati, Zhai, and Zhou, 2017) generally select the most salient sentences from the source article as the summary, which are precise at content selection making the result informative but suffer from high redundancy since it does not edit the sentences. In the opposite, abstractive methods (Paulus, Xiong, and Socher, 2017; Çelikyilmaz et al., 2018) can generate more concise summary via compressing and paraphrasing, while current models are weak at content selection and easy to lose crucial information. We can see that these two branches are complementary to each other, which motivates us to combine their advantages, generating both informative and concise summary.



Article:

For U.S. moms, the typical time between pregnancies is about 2 1/2 years ~~but nearly a third of women space their children too close, a government study shows.~~

Experts say mothers should wait at least 18 months to give their body time to recover and increase the chances the next child is full-term and healthy.

The study ~~released on Thursday by the Centers for Disease Control and Prevention~~ found that about 30 percent of women who 'd had a child became pregnant again within 18 months.

…

Human Summary:

① For U.S. moms, the typical time between pregnancies is about 2 1/2 years.

② Experts say mothers should wait at least 18 months to give their body time to recover and increase the chances the next child is full-term and healthy.

③ The study found that about 30 percent of women who 'd had a child became pregnant again within 18 months.

Figure 1: Sample summary of an article from the test set of CNN/DailyMail corpus (Hermann et al., 2015). The words used in summary are colored. In this sample, one sentence (green) is directly copied from article and two are rewritten to be concise.

Some effort has been made to combine these two branches. Most existing works use the extract-then-abstract framework that first extracts the summary-worthy sentences and then abstracts each of them (Dong et al., 2018; Mendes et al., 2019; Chen and Bansal, 2018). However, they suffer from an information loss in abstract stage, since all the sentence is compressed and pruned without a distinguish. When the whole sentence is crucial, some important content would be mistakenly deleted, causing a severe information loss. In addition, their training method is not end-to-end since there lacks an effective reinforcement learning framework to bridge together two modules.

In this paper, we introduce **HySum**, a novel hybrid framework that can flexibly switch between copy (original) and rewriting according to the degree of sentence redundancy. By distinguishingly copying the succinct sentences and rewriting the redundant ones, the information loss would be reduced. In this way, our method can effectively combine the strengths of both extractive and abstractive summarization methods, generating informative and concise summaries. Our strategy can improve the performance because it is more aligned with how human summarize a long article. Figure 1 shows the human behavior in summarization that some redundancy sentences are rewritten but others are kept unchanged.

Concretely, we construct our framework with a two-step approach. As shown in Figure 2, we first extract the salience sentences from the input article, with a **copy-or-write mechanism** to distinguish the sentences according to the redundancy. Then, the final summary is generated by correspondingly copying or rewriting the selected sentences. Moreover, we based on **Hierarchical Reinforcement Learning**, propose an *end-to-end* training method to bridge two independent steps, which can enhance the cooperation of them, dynamically adapt them to each other during training.

We use both automatic and human evaluation to test our framework on popular CNN/DailyMail corpus. Experimental results indicate that our methods significantly outperform the state-of-the-art summarization models. Human judges also prefer the summaries generated by our method and think they are more informative and concise.

The contributions of this paper are threefold:

- A novel framework is proposed, which can switch between copying and rewriting to effectively combine the advantages of extractive and abstractive summarization. To the best of our knowledge, we are the first to mix extracted and abstracted sentences in summary.
- We design an end-to-end reinforcement learning method for the two-step systems, which can enhance the cooperation of two modules.
- Our method achieves a new state-of-the-art on CNN/DailyMail corpus, and human evaluation also verifies the informativeness and conciseness of our results.

2 Summarization Framework

As shown in Figure 2, HySum is comprised of two steps: (1) *extraction*, that first encodes each sentence into a vector with *hierarchical BERT representation*, and then determines to copy or rewrite which sentences. (2) *sentence abstraction*, that further distills the content when simplification or paraphrase is necessary to output the final summary. These two steps are independent of parameter, and they learn to cooperate with each other via reinforcement learning.

Notation We denote each article $A = \{s_i\}$ as a sequence of sentences, and a sentence $s_i = \{w_m\}$ as a sequence of words. We embed each word w_m into a vector $\mathbf{w}_m^e$ and then represent the whole sentence into a n-dimensional vector $\mathbf{h}_i$. Extractor $E(\cdot)$ calculates the possibility $p_{ext}(s_i)$ of each sentence to be chosen and generate a subsequence $\{s_{j_t}\}$ that contains the most important sentences. Part of the chosen sentences are further edited by rewriter $W(\cdot)$ to generate the final summary $\tilde{Y} = \{\tilde{y}_t\}$ as a prediction for the human summary $Y = \{y_t\}$.

2.1 Hierarchical BERT Representation

Context and position information have been proven very important for the sentence selection. BERT (Bidirectional Encoder Representation from Transformers) (Devlin et al., 2019) has obtained a widely success on many downstream tasks with the strength of representation ability. BERT as an improvement for transformer, can generates bidirectional representation by jointly conditioning on both left and right context with long-distance dependencies. Also its position embedding also effectively injects the position information into the vectors. Therefore, we employ BERT and propose a **Hierarchical BERT Representation** method for sentence encoding, which is constructed by two stacked BERT networks to embed more context and position information in both word- and sentence- level. This design is also compatible for LSTM, so in experiment we replace BERT with and 1-layer BiLSTM as the baseline for BERT.

Word Context Embedding Unlike previous work that encode each sentence separately (Chen and Bansal, 2018; Zhang et al., 2018; Dong et al., 2018), we first feed the entire article into a pretrained BERT[1], which gives each word a wide range of context helping more precisely represent the meaning. Then the word representation is obtained by concatenating the last four layers and passing a MLP layer. This step injects context and also the word position in the whole article into the word vectors.

Sentence Context Embedding We obtain the preliminary representation for each sentence s_i by a mean pooling operation over the word vectors $\mathbf{h}_i = \frac{1}{|s_i|}\sum_{\mathbf{w}_m^e \in s_i} \mathbf{w}_m^e$, where $|\cdot|$ denotes the length. Then, to embed sentence positions information and sentence-level context into the representations, we further feed them into a one-layer BERT and obtain the final sentence vectors $\mathbf{h}_i$.

2.2 Extraction with Copy-or-Rewrite Mechanism

In this section we first introduce the basic extraction network for sentence selection, and then propose a copy-or-rewrite mechanism to determine whether to copy or rewrite each sentence.

Given the sentence vectors, we employ a pointer network (Vinyals, Fortunato, and Jaitly, 2015) to extract the article sentences that cover the most important information for the summary. At each time step t, we first calculate a sentence

[1]We use 'base-bert-cased' version for pretrained BERT provided by https://github.com/huggingface.

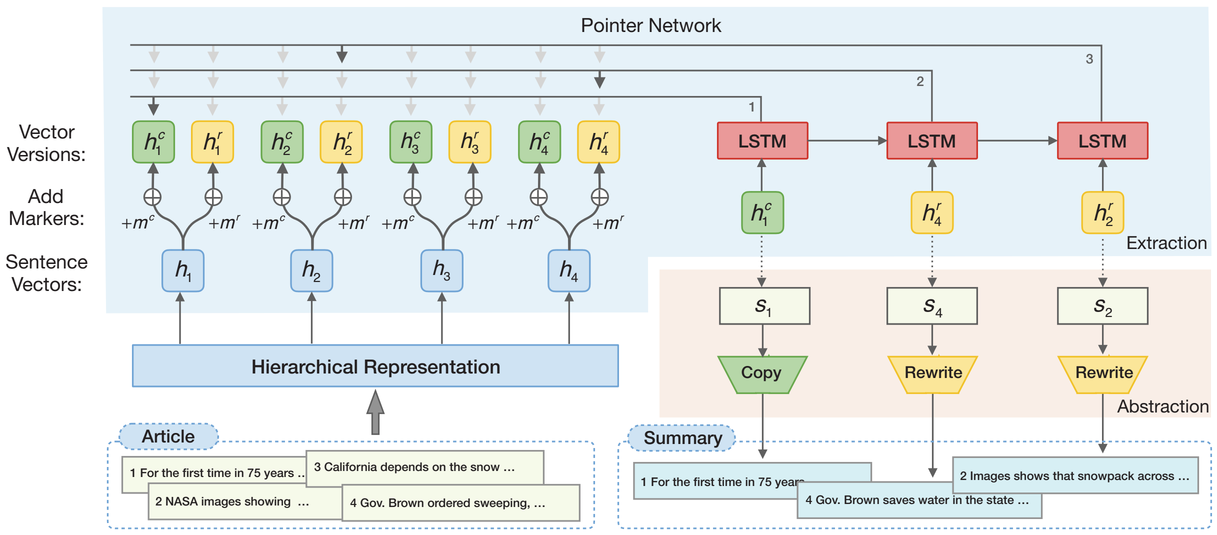

Pointer Network
Vector Versions:
Add Markers:
Sentence Vectors:
Hierarchical Representation
LSTM
Extraction
Abstraction
Copy
Rewrite
Article
1 For the first time in 75 years ...
2 NASA images showing ...
3 California depends on the snow ...
4 Gov. Brown ordered sweeping, ...
Summary
1 For the first time in 75 years
2 Images shows that snowpack across ...
4 Gov. Brown saves water in the state ...

Figure 2: The overview of the HYSUM framework. Hierarchical representation module first encodes the article sentences s_i into vectors $\mathbf{h}_j$. Then each sentence vector becomes two versions by adding with two different markers $\mathbf{m}^c, \mathbf{m}^r$. When the pointer network (arrows denote attention and darker color represents higher weights) selects the copy version $\mathbf{h}_i^c$ of a sentence, it will be copied. Otherwise when the rewriting version $\mathbf{h}_i^r$ is selected, the sentence will be rewritten to reduce redundancy.

context vector $\mathbf{e}_t$ based on the glimpse operation.

$$a_i^t = \mathbf{v}_s^\top \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_z \mathbf{z}_t) \quad (1)$$

$$\alpha^t = \text{softmax}(\mathbf{a}_t) \quad (2)$$

$$\mathbf{e}_t = \sum_t \alpha_i^t \mathbf{W}_h \mathbf{h}_i \quad (3)$$

where α^t are attention weights, $\mathbf{z}_t$ is the hidden state of LSTM and $\mathbf{W}_h, \mathbf{W}_z, \mathbf{v}_s$ are trainable parameters. Then extraction possibility for each sentence is calculated by:

$$u_i^t = \mathbf{v}_p^\top \tanh(\mathbf{W}_h' \mathbf{h}_i + \mathbf{W}_e \mathbf{e}_t) \quad (4)$$

$$P_{ext}(s_i^t) = \text{softmax}(\mathbf{u}^t) \quad (5)$$

where the sentence s_i^t with the highest possibility is selected. The extraction proceeds until the model selects the end-of-extraction token.

For the pretraining stage, the loss is calculated by cross entropy to maximum the log-likelihood: $\mathcal{L}_{ext} = -\frac{1}{T}\sum_t^T \log P_{ext}(s_{j_t}^* | s_{j_1}^*, \cdots, s_{j_{t-1}}^*, A)$, where $s_{j_t}^*$ is the t-th sentence in the ground-truth extraction sequence $(s_{j_1}^*, \cdots, s_{j_T}*)$ that is built with Equation (19).

Copy-or-Rewrite Mechanism It has been proven that rewriting all the selected sentences without distinguish hurts the performance (Chen and Bansal, 2018)[2]. So we endow our extraction module with the ability to distinguish what sentences should be copied and what should be rewritten. There are two difficulties to overcome. First, it is hard to build a ground-truth to supervise the decision prediction, because we cannot know whether rewriting or copy leads to the better overall ROUGE scores. Hence, we let the model learn the decision pattern by exploration in reinforcement learning. Second, reinforcement learning algorithm based on MDPs (Markov Decision Processes) supports only one action space, but our framework requires two, sentence selection and copy/rewrite decision making. To overcome this, we fuse two action spaces together with a novel copy-or-rewrite mechanism, taking two kinds of actions simultaneously.

As illustrated in Figure 2, after the sentence representation, we duplicate the sentences vectors $\{\mathbf{h}_i\}$ and add two different markers $(\mathbf{m}^c, \mathbf{m}^r)$ to them respectively:

$$\mathbf{h}_i^c = \mathbf{h}_i + \mathbf{m}^c \quad (6)$$

$$\mathbf{h}_i^r = \mathbf{h}_i + \mathbf{m}^r \quad (7)$$

in which the markers are trainable parameters to help the model distinguish two different operations for each sentence. Now each sentence has two different versions of vector. When the pointer network selects the copy version h_i^c for a sentence, it will be directly added to summary without any edition. In the opposite, if the rewriting version h_i^r is selected, the sentence would be rewritten (compress or paraphrase) to reduce the redundancy. In this way, we successfully merge two action spaces into one, making it suitable for current reinforcement learning.

Because most of the sentences need to be rewritten, at the start of training copy marker may lack of update. We find it is very easy for the model to generate a bias, tending to rewrite all the sentences. Therefore we propose two different approaches to make a balance between copy and rewriting.

Unbalanced Marker The first way is to use the relative scale of the makers to effect the choose. Since the markers are not pretrained, they can be regarded as a kind of noise for the representation vectors, which generates a slightly negative effect at the start of RL training. So we scale up the

[2]Their model with sentence rewriting is inferior to the their pure extraction version.

relative magnitude of rewriting marker $|\mathbf{m}^r| = \gamma \cdot |\mathbf{m}^c|$ to offset its starting advantage. Different magnitude is searched in the range of $[10^0, 10^5]$ and optimal setting is 10^2.

Copy-or-Rewrite History See, Liu, and Manning (2017) proposes a coverage mechanism for sentence generation, which tells model the attention history to help model attend on the non-selected items. Inspired by this, we also try to feed the attention history into network to let the model know how much sentence has been copied or rewritten. The attention accumulation of copy and rewriting before time step t on all I sentences is calculated by:

$$\varphi_t^c = \sum_{i=0}^{I} \sum_{t'=0}^{t-1} P_{ext}(\mathbf{h}_i^c) \tag{8}$$

$$\varphi_t^r = \sum_{i=0}^{I} \sum_{t'=0}^{t-1} P_{ext}(\mathbf{h}_i^r) \tag{9}$$

where φ_j^c, φ_j^r are the attention accumulation of copy and rewriting respectively. We concatenate them up $\varphi_t = \varphi_t^c \oplus \varphi_t^r$ and let them participate in the computation of attention score, and Equation (1)(4) can be revised to

$$a_i^t = \mathbf{v}_s^\top \tanh(\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_z \mathbf{z}_t + \mathbf{W}_\varphi \varphi_t) \tag{10}$$

$$u_i^t = \mathbf{v}_p^\top \tanh(\mathbf{W}'_h \mathbf{h}_i + \mathbf{W}_e \mathbf{e}_t + \mathbf{W}'_\varphi \varphi_t) \tag{11}$$

where $\mathbf{W}_\varphi, \mathbf{W}'_\varphi$ are trainable parameters.

2.3 Sentence Abstraction

This step follows the extraction decisions to copy or rewrite the corresponding sentences, generating the final summary. They copy operation aims to preserve all the information when the extracted sentence is already concise enough, and rewriting operation is employed to simplify or paraphrase the sentence that is redundant. To implement the latter, we use a vanilla encoder-aligner-decoder network (Bahdanau, Cho, and Bengio, 2014) with the copy mechanism (See, Liu, and Manning, 2017) to predict the out-of-vocabulary words.

In the pretraining stage of rewriter network, the loss for rewriting each ground-truth extracted sentence $s_{j_t}^*$ is calculated with cross-entropy:

$$\mathcal{L}_{abs} = -\frac{1}{M} \sum_{i=1}^{M} \log P_{abs}(y_t^{i*} | y_t^{1*}, \cdots, y_t^{i-1*}, s_{j_t}^*) \tag{12}$$

where the y_t^{i*} denotes the i-th words in sentence of human summary and M denotes the length of sentence.

3 Hierarchical Reinforcement Learning

Widely-used reinforcement learning (RL) methods such as REINFORCE (Williams, 1992) are not suitable for the two-step models that have two different agents. Hence, previous extract-then-abstract based models (Chen and Bansal, 2018; Zhang et al., 2018) only reinforce their extractor and fix the abstractor. This kind of training method is not end-to-end, so that the networks are not globally optimized due to the lack of RL training for abstractor.

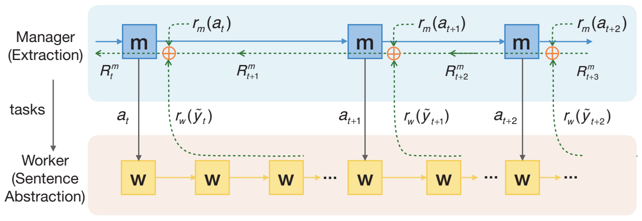


Figure 3: Architecture of our hierarchical reinforcement learning and reward composition (green lines) of extraction.

Therefore we based on Hierarchical Reinforcement Learning (HRL) framework (Vezhnevets et al., 2017; Dayan and Hinton, 1992), propose a simple end-to-end RL training method for the two-step systems. HRL typically takes a two-level architecture, which consists of (1) a *manager* at higher-level that assigns larger-grained tasks, (2) a *worker* at lower-level that selects specific actions to complete the given tasks. (3) an internal critic to distinguish whether the worker completes current task.

In our HRL method, we regard the extraction module as the manager that operates on sentence-level, and the sentence abstraction module as the worker that runs on word-level. The tasks are the selected sentences and copy-or-rewrite decision. The difference of our HRL methods with previous works is that it simplifies the training process by removing the internal critic and letting the worker itself to determine when the task is accomplished. Besides, we consider the reward of worker when estimating the reward of the manager, which more precisely describes the effect of manager actions.

3.1 Extraction Training

We analogize the extraction process with Markov Decision Processes (MDPs). At each step t, the manager agent π_{θ_m} takes an action a_t with its policy $\pi_{\theta_m}(a_t, c_t)$ according to the current state $c_t = \{A, a_{t-1}\}$. Then, the worker receives the selected sentence and copies/rewrites it to generate a summary sentence $\tilde{y}_t$. The objective of manager is to minimize the negative discounted return R_t^m, in formula

$$\triangledown_{\theta_m} \mathcal{L}(\theta_m) = - \triangledown \log \pi_{\theta_m}(a_t | c_t) R_t^m(a_t) \tag{13}$$

where return $R_t^m(a_t) = r_m(a_t) + \lambda R_{t+1}^m(a_{t+1})$ is the sum of reward after time t with discount λ. Because the action of manager also affects the performance of worker, we also consider the worker's reward $r_w(\tilde{y}_t)$ to more precisely describe the effect of manager's action. Then the return turns to $R_t^m(a_t) = r_m(a_t) + \lambda R_{t+1}^m(a_{t+1}) + \beta r_w(\tilde{y}_t)$, and Equation (13) can be revised to

$$\triangledown_{\theta_m} \mathcal{L}(\theta_m) = - \triangledown \log \pi_{\theta_m}(a_t | c_t)[r_m(a_t) + \lambda R_{t+1}^m(a_{t+1}) + \beta r_w(\tilde{y}_t)] \tag{14}$$

Usually, directly maximizing the return would suffer from a high variance, so we reduce the variance by adding a baseline as follows:

$$\triangledown_{\theta_m} \mathcal{L}(\theta_m) = - \triangledown \log \pi_{\theta_m}(a_t | c_t)[r_m(a_t) + \lambda R_t^m(a_{t+1}) + \beta r_w(\tilde{y}_t) - b_t^m] \tag{15}$$

where b_t^m is an arbitrary baseline function independent from the action. Baseline is an approximation for the state-value function $V^{\pi_\theta}(c_t)$. In order to predict the baseline, we follow the *A2C* (Advantage Actor-Critic) algorithm, constructing a critic network to predict it and optimizing its with a mean square error (MSE) loss $\mathcal{L}_b = (b_t^m - R_t^m)^2$.

Marginal Reward Directly using ROUGE score as the reward for each extracted sentence is not global optimized, because it cannot accurately evaluate the contribution of each step to the overall summary. So we use *margin reward* to compute a incremental score for each extracted sentence, which ensures that each sentence contribute novel information to the whole summary.

$$r_m(a_t) = \text{ROUGE}(a_{1:t}, Y) - \text{ROUGE}(a_{1:t-1}, Y) \quad (16)$$

where $a_{1:t}$ denotes the concatenation of selected sentences from time step 1 to t and $Y = \{y_t\}$ denotes the overall human summary. For the score function ROUGE, we find that using only ROUGE-L as (Paulus, Xiong, and Socher, 2017) will depress the ROUGE-2 score. So we combine multiple metrics to make a balance [3]:

$$\begin{aligned}\text{ROUGE}(x, y) =& \frac{1}{3}\text{ROUGE-1}_{F_1}(x, y) + \text{ROUGE-2}_{F_1}(x, y) \\ &+ \text{ROUGE-L}_{F_1}(x, y) \quad (17)\end{aligned}$$

3.2 Sentence Abstraction Training

When training the worker, we assume manager has Oracle policy and update worker's parameter with policy gradient described in Eq. (13). For baseline, following (Paulus, Xiong, and Socher, 2017) we employ self-critic training algorithm to use greedily-decoded sentence as baseline.

At each step t for rewriting, two output sequences $\hat{y}_t, \tilde{y}_t$ are generated, where the $\hat{y}_t$ is sampled from the probability distributions and $\tilde{y}_t$ is greedily generated by argmax at each time step to be the baseline. The sampled actions that get more reward $r_w(\cdot)$ are encouraged via the following loss function.

$$\mathcal{L}_{abs} = -(r_w(\hat{y}_t) - r_w(\tilde{y}_t)) \sum_{i=1}^{N} \log p_{\theta_w}(\hat{y}_t^i) \quad (18)$$

where $\hat{y}_t^i$ denotes the i-th word in sampled sequence, and the reward is defined as $r_w(\hat{y}_t) = \text{ROUGE}(\hat{y}_t, y_t)$.

4 Experiment

In this section, we verify our methods on the most popular dataset CNN/DailyMail, so that we can make a fully comparison with the state-of-the-art works.

4.1 Dataset and Ground-Truth Construction

We evaluate our approach on the summarization corpus CNN/Daily Mail, which is comprised of news stories and is first proposed by Hermann et al. (2015). We use the non-anonymized version that does not replace the name entities, since it is the most frequently used version in recent works. We follow the processing steps in (See, Liu, and Manning, 2017), obtaining 287,188 training, 13,367 validation and 11, 490 testing samples.

To pretrain two steps of HySum as a warmup for the RL training. We align each reference sentence y_t to an article sentence by

$$s_{j_t} = \max_{s_i \in A}(\text{ROUGE-L}_{recall}(s_i, y_t)) \quad (19)$$

Selected sentences s_{j_t} are used to supervise the extraction modules and sentence pairs (s_{j_t}, y_t) are used for rewriter training.

4.2 Training Details and Evaluation Method

The word vectors (Mikolov et al., 2013) are pretrained on the whole dataset with 128 dimension. And all the LSTM networks in our framework use 256 hidden units. Adam optimizer (Kingma and Ba, 2014) is applied with a learning rate 0.001 for extractor and 0.0001 for rewriter, and the mini-batches size is set to 32. We also decrease the learning rate during training to make a better convergence. In RL training stage, we set the discount factor λ as 0.95 for return and coefficient β as 0.9 with grid search. During reference, we apply the beam search (Pasunuru and Bansal, 2018) with width 5 on rewriter to avoid trigram repetition.

We evaluate our method with standard ROUGE-1, ROUGE-2 and ROUGE-L (Hsu et al., 2018) on full-length F_1. We finally choose the models achieving the highest ROUGE-L score on validation set and report the performance on test set. We also follow See, Liu, and Manning (2017) to evaluate results on METEOR (Denkowski and Lavie, 2014) that considers synonyms, paraphrases, and stemming for a more comprehensive comparison.

4.3 Results

Automatic Evaluation We report the results of our model in Table 1. Our model significantly outperforms ($p < 0.001$) all the competitors on ROUGE-1, ROUGE-L and METEOR. HySum with HRL training (Copy/Rewrite + History + HRL) improves the ROUGE-1 and ROUGE-L scores by 0.96 and 0.65 respectively over the best previous work, which proves the effectiveness of our copy-or-rewrite framework and the new training method. Moreover, we also investigate the efficacy of pre-trained language model, which recently shows an extensive success on many downstream tasks. By equipping the model with the hierarchical BERT representation, our model obtains the highest scores on all four metrics, which creates a new state-of-the-art result for the summarization task on CNN/DailyMail corpus. This indicates that our hierarchical structure for using pretrained BERT is effective.

Ablation Study We also conduct some ablation studies in Table 1 to verify the effectiveness of each component. For **copy-or-rewrite mechanism**, we build two baseline models, Copy+EXT-RL and Rewrite+EXT-RL, which have only copy (i.e. the pure extraction model) or rewriting operation and only the extractor is reinforced. The fact that only-copy

[3]ROUGE-1 and ROUGE-L are strongly connected, so we lower the coefficient of ROUGE-1 to avoid bias.

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Extract-based Models				
SummaRuNNer (Nallapati, Zhai, and Zhou, 2017)	39.60	16.20	35.30	-
RankingRL (Narayan, Cohen, and Lapata, 2018)	40.0	18.2	36.6	-
BANDITSUM (Dong et al., 2018)	41.5	18.7	37.6	-
Abstract-based Models				
PointerGen+Coverage (See, Liu, and Manning, 2017)	39.53	17.28	36.38	18.72
DeepRL (Paulus, Xiong, and Socher, 2017)	39.87	15.82	36.90	-
Inconsistency Loss (Hsu et al., 2018)	40.68	17.97	37.15	-
Fast-RL (Chen and Bansal, 2018)	40.88	17.80	38.54	20.38
Bottom Up (Gehrmann, Deng, and Rush, 2018)	41.22	18.68	38.34	19.42
DCA† (Çelikyilmaz et al., 2018)	40.91	19.21	38.03	18.13
Our Results				
Rewrite + EXT-RL	40.73	17.85	38.26	19.05
Copy + EXT-RL	41.48	18.75	37.79	21.71
Copy/Rewrite + Unbalanced Marker + EXT-RL	42.10*	18.91	38.87	20.69*
Copy/Rewrite + History + EXT-RL	41.98*	18.96	38.83	21.91*
Copy/Rewrite + History + INDEPENDENT-RL	42.21*	18.91	38.94*	21.16*
Copy/Rewrite + History + HRL	42.46*	19.10	39.19*	21.88*
Copy/Rewrite + History + HRL + BERT	**42.92***	**19.43***	**39.35***	**22.12***

Table 1: Results on CNN/DailyMail. The best scores are in bold, and significantly better scores than the baselines are marked with $*$ ($p < 0.001$, t-test). EXT-RL denotes the RL training on extraction module, and BERT denotes the hierarchical BERT representation. †: We re-evaluate DCA output against full human summaries, and the difference on ROUGE-2 score to our HRL model is not significant. ($p = 0.517$).

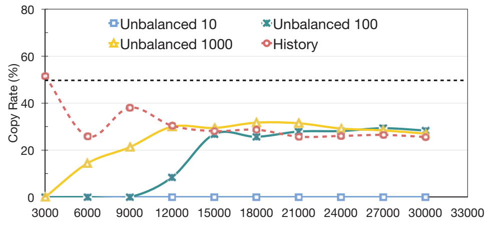


Figure 4: Copy rate learning curve of two balancing mechanisms. Unbalanced 100: $|m^r| = 10^2|m^c|$; Unbalanced 1000: $|m^r| = 10^3|m^c|$.

	Info.	Con.	Read.
Fast-RL	2.83	3.1	3.86
HySum (ours)	3.21	3.83	4.09
HySum + BERT (ours)	**3.40**	**3.85**	**4.11**

Table 2: Human evaluation on informativity (Info.), conciseness (Con.) and readability(Read.). Best results among all methods are in bold.

version outperforms only-rewrite version verifies the knowledge that rewriting all selected sentence causes information loss. By combining both operations (Copy/Rewrite + EXT-RL), there is an exciting improvement on all metrics, which proves that jointly using copy and rewriting operations can extremely preserve the summary-worthy content, avoiding information loss.

Unbalanced Marker and History mechanism obtain very similar performance. They are both capable of reducing the bias of extractor. But interestingly they learn the copy rate in totally different ways. As Fig. 4 shows, unbalanced marker mechanism learns to copy gradually from zero, while the history mechanism starts with randomly choose (50%) and converges towards the optimized rate. Finally both of them approach a copy rate of about 28%. The consistency of them proves this is best rate for the data distribution.

HRL Training's advantage is not only from reinforcing the rewriter, but also from the improvement for the cooperation between two modules. To prove this, we build a baseline that separately reinforces two networks (Copy/Rewrite + History + INDEPENDENT-RL). Unsurprisingly, it has only a slight improvement over the extraction reinforced model (Copy/Rewrite + History + EXT-RL), which indicates the importance of the end-to-end training method like HRL for a two-agent system.

Human Evalution We perform human evaluation to establish that the improvements on ROUGE are correlated with human judgments. We compare our model with HRL training to Fast-RL (Chen and Bansal, 2018), which extracts and compresses every sentence. We employ following factors as evaluation criterias for decoded summaries to show the benefits of our framework: *informativity*, *conciseness* and *readability*. We randomly select 50 samples from the CNN/DailyMail test set and hire 3 workers as judges to evaluate the generated summaries. We show the judges the original article and two model outputs and ask them to score the each summary from 1(worst) to 5(best). For each sample, summaries are shuffled to make a fair comparison. We average the scores across all the samples and judges, and the results are reported in Table 2.

Human judges significantly prefer the summaries generated by our model, and give higher scores on all three eval-

Human Summary: Nicholas dematteis, 39, flew into a rage while at restaurant bocca east during brunch on saturday afternoon. He demanded free meal around 4 pm because of slow service, police said. Witnesses said he spewed homophobic slur at a manager before grabbing him by the neck and slamming him into bar and hurling him into woman. He has been arrested and charged with assault following the incident.

Baseline Models:
Fast-RL: Nicholas dematteis flew into the rage while at restaurant bocca east. Dematteis was arrested and charged with assault following the incident. The 34-year-old manager made attempts to calm dematteis down. Nicholas dematteis, 39, flew into the rage while waiting an hour and a half. New york diner lost his cool grabbing a restaurant manager by the neck.
Bottom-up: Nicholas dematteis, 39, flew into the rage after waiting more than an hour for his omelette. Dematteis was arrested and charged with assault following the incident on saturday. Dematteis was previously arrested earlier this year in january for breaking into his girlfriend 's apartment.

HYSUM: [copy]Nicholas dematteis, 39, flew into the rage while at restaurant bocca east during brunch on saturday afternoon, and demanded a free meal around 4 pm because of slow service, police said. [rewrite]Dematteis was arrested and charged with assault following the incident. [rewrite]He grabbed his restaurant manager by the neck before slamming him against a bar and into an elderly woman.

Figure 5: Summary comparison with the baseline models. Overlapped content is colored. Our model overlaps the most with human summary and generates less redundancy.

uation dimensions. The largest advantage of our model is on the dimension of informativity and conciseness, because by copying original sentences our model can reserve all the important information when the whole sentence is crucial, avoiding information loss, and via reinforcing the rewriter, our model obtains a stronger abstraction ability, making the generated summary more concise.

Case Study Figure 5 illustrates several summaries generated on the test set. We can see that HYSUM is better on selection, which covers more content appearing in human summary. Especially for the first sentence, HYSUM chooses to copy the whole sentence, which to the most extend avoids the information loss by abstraction. In the opposite, the baseline Fast-RL, as an extract-then-abstract model, correctly selects the first sentence, but further compression discards the second half the sentence, which losses important information. Besides, the output from model Bottom-up is redundant since pure abstractive model is week in content selection.

5 Related Work

In this section, we introduce the related work from two threads: 1) the combination of extractive and abstractive summarization; 2) the usage of reinforcement learning in the summarization.

Combining the two branches of summariztion, extraction and abstraction, has become popular in summarization task and some extract-then-abstract models are proposed. They first identify the salient sentences and then abstract the selected together (Nallapati, Zhai, and Zhou, 2017; Sharma et al., 2019) or one by one (Chen and Bansal, 2018; Dong et al., 2018). These models make an improvement because they exploit the complementarity of two branches of summarization that 1) extractive models (Cheng and Lapata, 2016) excel at content selection but cannot remove the redundancy; 2) abstractive models (Nallapati et al., 2016) can simplify or paraphrase the sentence, but it is week in content selection. However, rewriting all the sentence without distinguish hurts the performance, which has been proved in (Chen and Bansal, 2018) that their rewriting model is inferior to pure extraction model. This is because some sentences are already concise enough and compressing or paraphrasing them with unperfect rewriter would lost some key information. Therefore, we propose a hybrid framework that can distinguish the sentences and allow using raw sentences in summary. To the best of our knowledge, we are the first to mix the extractive and abstractive sentences in summary.

Reinforcement learning has been introduced into sequence generation task to directly optimize the non-differential metrics and generation quality by alleviating the 'exposure bias'. Dong et al. (2018); Narayan, Cohen, and Lapata (2018); Hermann et al. (2015) use policy gradient to improve their extractive models and Paulus, Xiong, and Socher (2017); Çelikyilmaz et al. (2018) employ weighted ML+RL loss to reinforce their abstractive encoder-decoder models. Unfortunately, rare end-to-end reinforcement learning method is proposed for two-step models. Current two-step models only reinforce one network and fix the other e.g. (Gu et al., 2016) for translation, (Choi et al., 2017) for question answer and (Chen and Bansal, 2018; Zhang et al., 2018) for summarization. This is not globally optimized and lacks mutual adaption between two networks. Therefore a gap exists between the reinforcement learning and the two-step models proposed for language generation. In this work, we propose a hierarchical reinforcement learning approach, which can reinforce the whole framework end-to-end and enhance the cooperation of two networks. Our training method is for summarization but also is appliable for similar two-step systems.

6 Conclusion

In this paper, we propose a novel hybrid summarization framework, which for the first time mixes the extracted and abstracted sentences in the summary. A copy-or-rewrite mechanism is designed to distinguish the sentences that can be directly used in summary and the sentences that need to be rewritten. Moreover, we propose an end-to-end hierarchical reinforcement learning method for training two-step models, which uses extracted sentence as the task from manager to worker, extremely improving the cooperation between two networks.

Acknowledgements

This research is supported in part by National Key Research and Development Program of China under Grant 2018YFC0830400 and National Natural Science Foundation of China under Grant 61575123. Lu Wang is supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341. We also thank Sebastian Sehrmann and Asli Celikyilmaz for sharing their summaries on CNN/DM.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Çelikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep communicating agents for abstractive summarization. In *Proceedings of the 2018 Conference of the NAACL-HLT, 2018*, 1662–1675.

Chen, Y., and Bansal, M. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, 675–686.

Cheng, J., and Lapata, M. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the ACL 2016*.

Choi, E.; Hewlett, D.; Uszkoreit, J.; Polosukhin, I.; Lacoste, A.; and Berant, J. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the ACL 2017*, 209–220.

Dayan, P., and Hinton, G. E. 1992. Feudal reinforcement learning. In *Advances in Neural Information Processing Systems 5, [NIPS Conference, 1992]*, 271–278.

Denkowski, M. J., and Lavie, A. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation, ACL 2014*, 376–380.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the NAACL-HLT 2019*, 4171–4186.

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. Banditsum: Extractive summarization as a contextual bandit. In *Proceedings of the 2018 Conference of the EMNLP, 2018*, 3739–3748.

Gehrmann, S.; Deng, Y.; and Rush, A. M. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference of the EMNLP, 2018*, 4098–4109.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. K. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Hermann, K. M.; Kociský, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: NIPS 2015*.

Hsu, W. T.; Lin, C.; Lee, M.; Min, K.; Tang, J.; and Sun, M. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proceedings of the 56th Annual Meeting of the ACL 2018*, 132–141.

Jing, H., and McKeown, K. R. 2000. Cut and paste based text summarization. In *6th Applied Natural Language Processing Conference, ANLP 2000*, 178–185.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Knight, K., and Marcu, D. 2000. Statistics-based summarization - step one: Sentence compression. In *InProceedings of the 2000 AAAI Conference on Artificial Intelligence,*, 703–710.

Mendes, A.; Narayan, S.; Miranda, S.; Marinho, Z.; Martins, A. F. T.; and Cohen, S. B. 2019. Jointly extracting and compressing documents with summary state representations. In *Proceedings of the 2019 Conference of the NAACL-HLT 2019*, 3955–3966.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *27th Annual Conference of NeurlPS 2013.*, 3111–3119.

Nallapati, R.; Zhou, B.; dos Santos, C. N.; Gülçehre, Ç.; and Xiang, B. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Conference on Computational Natural Language Learning, CoNLL 2016*, 280–290.

Nallapati, R.; Zhai, F.; and Zhou, B. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017*, 3075–3081.

Narayan, S.; Cohen, S. B.; and Lapata, M. 2018. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the NAACL-HLT 2018*, 1747–1759.

Pasunuru, R., and Bansal, M. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the NAACL-HLT, 2018*, 646–653.

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *CoRR* abs/1705.04304.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*.

Sharma, E.; Huang, L.; Hu, Z.; and Wang, L. 2019. An entity-driven framework for abstractive summarization.

Vezhnevets, A. S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; and Kavukcuoglu, K. 2017. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th of the ICML 2017*, 3540–3549.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28: NIPS 2015*, 2692–2700.

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

Zhang, X.; Lapata, M.; Wei, F.; and Zhou, M. 2018. Neural latent extractive document summarization. In *Proceedings of the Conference of EMNLP, 2018*, 779–784.