Fast Abstractive Summarization with Reinforce-Selected Sentence Rewriting

Yen-Chun Chen and **Mohit Bansal**
UNC Chapel Hill
{yenchun, mbansal}@cs.unc.edu

Abstract

Inspired by how humans summarize long documents, we propose an accurate and fast summarization model that first selects salient sentences and then rewrites them abstractively (i.e., compresses and paraphrases) to generate a concise overall summary. We use a novel sentence-level policy gradient method to bridge the non-differentiable computation between these two neural networks in a hierarchical way, while maintaining language fluency. Empirically, we achieve the new state-of-the-art on all metrics (including human evaluation) on the CNN/Daily Mail dataset, as well as significantly higher abstractiveness scores. Moreover, by first operating at the sentence-level and then the word-level, we enable *parallel decoding* of our neural generative model that results in substantially faster (10-20x) inference speed as well as 4x faster training convergence than previous long-paragraph encoder-decoder models. We also demonstrate the generalization of our model on the test-only DUC-2002 dataset, where we achieve higher scores than a state-of-the-art model.

1 Introduction

The task of document summarization has two main paradigms: extractive and abstractive. The former method directly chooses and outputs the salient sentences (or phrases) in the original document (Jing and McKeown, 2000; Knight and Marcu, 2000; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011). The latter abstractive approach involves rewriting the summary (Banko et al., 2000; Zajic et al., 2004), and has seen substantial recent gains due to neural sequence-to-sequence models (Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017; Paulus et al., 2018). Abstractive models can be more concise by performing generation from scratch, but they suffer from slow and inaccurate encoding of very long documents, with the attention model being required to look at all encoded words (in long paragraphs) for decoding each generated summary word (slow, one by one sequentially). Abstractive models also suffer from redundancy (repetitions), especially when generating multi-sentence summary.

To address both these issues and combine the advantages of both paradigms, we propose a hybrid extractive-abstractive architecture, with policy-based reinforcement learning (RL) to bridge together the two networks. Similar to how humans summarize long documents, our model first uses an *extractor agent* to select salient sentences or highlights, and then employs an *abstractor network* to rewrite (i.e., compress and paraphrase) each of these extracted sentences. To overcome the non-differentiable behavior of our extractor and train on available document-summary pairs without saliency label, we next use actor-critic policy gradient with sentence-level metric rewards to connect these two neural networks and to learn sentence saliency. We also avoid common language fluency issues (Paulus et al., 2018) by preventing the policy gradients from affecting the abstractive summarizer's word-level training, which is supported by our human evaluation study. Our sentence-level reinforcement learning takes into account the word-sentence hierarchy, which better models the language structure and makes parallelization possible. Our extractor combines reinforcement learning and pointer networks, which is inspired by Bello et al. (2017)'s attempt to solve the Traveling Salesman Problem. Our abstractor is a simple encoder-aligner-decoder

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pages 675–686
Melbourne, Australia, July 15 - 20, 2018.

model (with copying) and is trained on pseudo document-summary sentence pairs obtained via simple automatic matching criteria.

Thus, our method incorporates the abstractive paradigm's advantages of concisely rewriting sentences and generating novel words from the full vocabulary, yet it adopts intermediate extractive behavior to improve the overall model's quality, speed, and stability. Instead of encoding and attending to every word in the long input document sequentially, our model adopts a human-inspired *coarse-to-fine* approach that first extracts all the salient sentences and then decodes (rewrites) them (*in parallel*). This also avoids almost all redundancy issues because the model has already chosen non-redundant salient sentences to abstractively summarize (but adding an optional final reranker component does give additional gains by removing the fewer across-sentence repetitions).

Empirically, our approach is the new state-of-the-art on all ROUGE metrics (Lin, 2004) as well as on METEOR (Denkowski and Lavie, 2014) of the CNN/Daily Mail dataset, achieving statistically significant improvements over previous models that use complex long-encoder, copy, and coverage mechanisms (See et al., 2017). The test-only DUC-2002 improvement also shows our model's better generalization than this strong abstractive system. In addition, we surpass the popular lead-3 baseline on all ROUGE scores with an abstractive model. Moreover, our sentence-level abstractive rewriting module also produces substantially more (3x) novel N-grams that are not seen in the input document, as compared to the strong flat-structured model of See et al. (2017). This empirically justifies that our RL-guided extractor has learned sentence saliency, rather than benefiting from simply copying longer sentences. We also show that our model maintains the same level of fluency as a conventional RNN-based model because the reward does not leak to our abstractor's word-level training. Finally, our model's training is 4x and inference is more than 20x faster than the previous state-of-the-art. The optional final reranker gives further improvements while maintaining a 7x speedup.

Overall, our contribution is three fold: First we propose a novel sentence-level RL technique for the well-known task of abstractive summarization, effectively utilizing the word-then-sentence hierarchical structure without annotated matching sentence-pairs between the document and ground truth summary. Next, our model achieves the new state-of-the-art on all metrics of multiple versions of a popular summarization dataset (as well as a test-only dataset) both extractively and abstractively, without loss in language fluency (also demonstrated via human evaluation and abstractiveness scores). Finally, our parallel decoding results in a significant 10-20x speed-up over the previous best neural abstractive summarization system with even better accuracy.[1]

2 Model

In this work, we consider the task of summarizing a given long text document into several (ordered) highlights, which are then combined to form a multi-sentence summary. Formally, given a training set of document-summary pairs $\{x_i, y_i\}_{i=1}^N$, our goal is to approximate the function $h : X \rightarrow Y, X = \{x_i\}_{i=1}^N, Y = \{y_i\}_{i=1}^N$ such that $h(x_i) = y_i, 1 \leq i \leq N$. Furthermore, we assume there exists an abstracting function g defined as: $\forall s \in S_i, \exists d \in D_i$ such that $g(d) = s, 1 \leq i \leq N$, where S_i is the set of summary sentences in x_i and D_i the set of document sentences in y_i. i.e., in any given pair of document and summary, every summary sentence can be produced from some document sentence. For simplicity, we omit subscript i in the remainder of the paper. Under this assumption, we can further define another latent function $f : X \rightarrow D^n$ that satisfies $f(x) = \{d_t\}_{j=1}^n$ and $y = h(x) = [g(d_1), g(d_2), \ldots, g(d_n)]$, where $[,]$ denotes sentence concatenation. This latent function f can be seen as an extractor that chooses the salient (ordered) sentences in a given document for the abstracting function g to rewrite. Our overall model consists of these two submodules, the *extractor agent* and the *abstractor network*, to approximate the above-mentioned f and g, respectively.

2.1 Extractor Agent

The extractor agent is designed to model f, which can be thought of as extracting salient sentences from the document. We exploit a hierarchical neural model to learn the sentence representations of the document and a 'selection network' to extract sentences based on their representations.

[1] We are releasing our code, best pretrained models, as well as output summaries, to promote future research: https://github.com/ChenRocks/fast_abs_rl

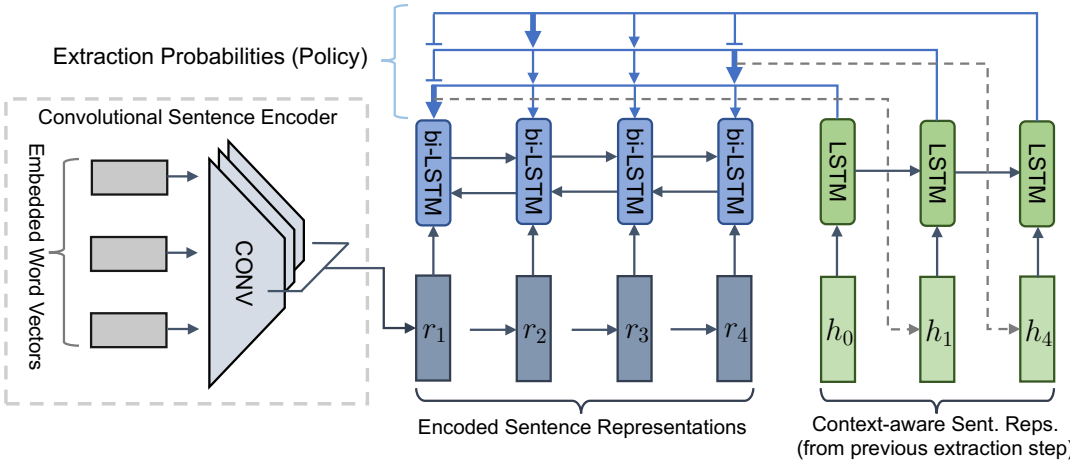

Figure 1: Our extractor agent: the convolutional encoder computes representation r_j for each sentence. The RNN encoder (blue) computes context-aware representation h_j and then the RNN decoder (green) selects sentence j_t at time step t. With j_t selected, h_{j_t} will be fed into the decoder at time $t+1$.

2.1.1 Hierarchical Sentence Representation

We use a temporal convolutional model (Kim, 2014) to compute r_j, the representation of each individual sentence in the documents (details in supplementary). To further incorporate global context of the document and capture the long-range semantic dependency between sentences, a bidirectional LSTM-RNN (Hochreiter and Schmidhuber, 1997; Schuster et al., 1997) is applied on the convolutional output. This enables learning a strong representation, denoted as h_j for the j-th sentence in the document, that takes into account the context of all previous and future sentences in the same document.

2.1.2 Sentence Selection

Next, to select the extracted sentences based on the above sentence representations, we add another LSTM-RNN to train a *Pointer Network* (Vinyals et al., 2015), to extract sentences recurrently. We calculate the extraction probability by:

$$u_j^t = \begin{cases} v_p^\top \tanh(W_{p1} h_j + W_{p2} e_t) & \text{if } j_t \neq j_k \\ & \forall k < t \\ -\infty & \text{otherwise} \end{cases} \quad (1)$$

$$P(j_t | j_1, \ldots, j_{t-1}) = \text{softmax}(u^t) \quad (2)$$

where e_t's are the output of the *glimpse* operation (Vinyals et al., 2016):

$$a_j^t = v_g^\top \tanh(W_{g1} h_j + W_{g2} z_t) \quad (3)$$

$$\alpha^t = \text{softmax}(a^t) \quad (4)$$

$$e_t = \sum_j \alpha_j^t W_{g1} h_j \quad (5)$$

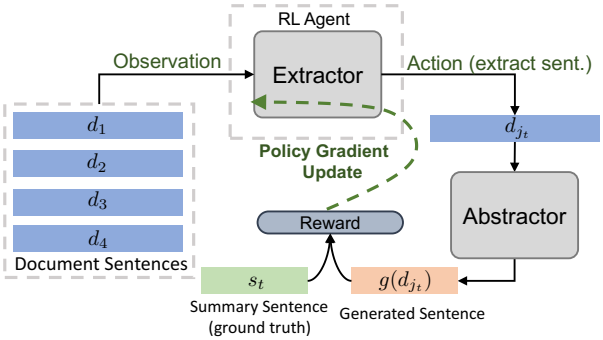

Figure 2: Reinforced training of the extractor (for one extraction step) and its interaction with the abstractor. For simplicity, the critic network is not shown. Note that all d's and s_t are raw sentences, *not* vector representations.

In Eqn. 3, z_t is the output of the added LSTM-RNN (shown in green in Fig. 1) which is referred to as the *decoder*. All the W's and v's are trainable parameters. At each time step t, the decoder performs a 2-hop attention mechanism: It first attends to h_j's to get a context vector e_t and then attends to h_j's again for the extraction probabilities.[2] This model is essentially classifying all sentences of the document at each extraction step. An illustration of the whole extractor is shown in Fig. 1.

2.2 Abstractor Network

The abstractor network approximates g, which compresses and paraphrases an extracted document sentence to a concise summary sentence. We

[2]Note that we force-zero the extraction prob. of already extracted sentences so as to prevent the model from using repeating document sentences and suffering from redundancy. This is non-differentiable and hence only done in RL training.

use the standard encoder-aligner-decoder (Bahdanau et al., 2015; Luong et al., 2015). We add the copy mechanism[3] to help directly copy some out-of-vocabulary (OOV) words (See et al., 2017). For more details, please refer to the supplementary.

3 Learning

Given that our extractor performs a non-differentiable *hard* extraction, we apply standard policy gradient methods to bridge the back-propagation and form an end-to-end trainable (stochastic) computation graph. However, simply starting from a randomly initialized network to train the whole model in an end-to-end fashion is infeasible. When randomly initialized, the extractor would often select sentences that are not relevant, so it would be difficult for the abstractor to learn to abstractively rewrite. On the other hand, without a well-trained abstractor the extractor would get noisy reward, which leads to a bad estimate of the policy gradient and a sub-optimal policy. We hence propose optimizing each sub-module separately using maximum-likelihood (ML) objectives: train the extractor to select salient sentences (fit f) and the abstractor to generate shortened summary (fit g). Finally, RL is applied to train the full model end-to-end (fit h).

3.1 Maximum-Likelihood Training for Submodules

Extractor Training: In Sec. 2.1.2, we have formulated our sentence selection as classification. However, most of the summarization datasets are end-to-end document-summary pairs without extraction (saliency) labels for each sentence. Hence, we propose a simple similarity method to provide a 'proxy' target label for the extractor. Similar to the extractive model of Nallapati et al. (2017), for each ground-truth summary sentence, we find the most similar document sentence d_{j_t} by:[4]

$$j_t = \text{argmax}_i(\text{ROUGE-L}_{recall}(d_i, s_t)) \quad (6)$$

Given these proxy training labels, the extractor is then trained to minimize the cross-entropy loss.

Abstractor Training: For the abstractor training, we create training pairs by taking each summary sentence and pairing it with its extracted document sentence (based on Eqn. 6). The network is trained as an usual sequence-to-sequence model to minimize the cross-entropy loss $L(\theta_{abs}) = -\frac{1}{M}\sum_{m=1}^{M} \log P_{\theta_{abs}}(w_m|w_{1:m-1})$ of the decoder language model at each generation step, where θ_{abs} is the set of trainable parameters of the abstractor and w_m the m^{th} generated word.

3.2 Reinforce-Guided Extraction

Here we explain how policy gradient techniques are applied to optimize the whole model. To make the extractor an RL agent, we can formulate a Markov Decision Process (MDP)[5]: at each extraction step t, the agent observes the current state $c_t = (D, d_{j_{t-1}})$, samples an action $j_t \sim \pi_{\theta_{a,\omega}}(c_t, j) = P(j)$ from Eqn. 2 to extract a document sentence and receive a reward[6]

$$r(t+1) = \text{ROUGE-L}_{F_1}(g(d_{j_t}), s_t) \quad (7)$$

after the abstractor summarizes the extracted sentence d_{j_t}. We denote the trainable parameters of the extractor agent by $\theta = \{\theta_a, \omega\}$ for the decoder and hierarchical encoder respectively. We can then train the extractor with policy-based RL. We illustrate this process in Fig. 2.

The vanilla policy gradient algorithm, REINFORCE (Williams, 1992), is known for high variance. To mitigate this problem, we add a critic network with trainable parameters θ_c to predict the state-value function $V^{\pi_{\theta_{a,\omega}}}(c)$. The predicted value of critic $b_{\theta_{c,\omega}}(c)$ is called the 'baseline', which is then used to estimate the *advantage function*: $A^{\pi_\theta}(c, j) = Q^{\pi_{\theta_{a,\omega}}}(c, j) - V^{\pi_{\theta_{a,\omega}}}(c)$ because the total return R_t is an estimate of action-value function $Q(c_t, j_t)$. Instead of maximizing $Q(c_t, j_t)$ as done in REINFORCE, we maximize $A^{\pi_\theta}(c, j)$ with the following policy gradient:

$$\nabla_{\theta_{a,\omega}} J(\theta_a, \omega) = \mathbb{E}[\nabla_{\theta_{a,\omega}} \log \pi_\theta(c, j) A^{\pi_\theta}(c, j)] \quad (8)$$

And the critic is trained to minimize the square loss: $L_c(\theta_c, \omega) = (b_{\theta_{c,\omega}}(c_t) - R_t)^2$. This is

[3] We use the terminology of *copy mechanism* (originally named *pointer-generator*) in order to avoid confusion with the *pointer network* (Vinyals et al., 2015).

[4] Nallapati et al. (2017) selected sentences greedily to maximize the global summary-level ROUGE, whereas we match exactly 1 document sentence for each GT summary sentence based on the *individual* sentence-level score.

[5] Strictly speaking, this is a *Partially Observable Markov Decision Process* (POMDP). We approximate it as an MDP by assuming that the RNN hidden state contains all past info.

[6] In Eqn. 6, we use ROUGE-recall because we want the extracted sentence to contain as much information as possible for rewriting. Nevertheless, for Eqn. 7, ROUGE-F_1 is more suitable because the abstractor g is supposed to rewrite the extracted sentence d to be as *concise* as the ground truth s.

known as the *Advantage Actor-Critic* (A2C), a synchronous variant of A3C (Mnih et al., 2016). For more A2C details, please refer to the supp.

Intuitively, our RL training works as follow: If the extractor chooses a good sentence, after the abstractor rewrites it the ROUGE match would be high and thus the action is encouraged. If a bad sentence is chosen, though the abstractor still produces a compressed version of it, the summary would not match the ground truth and the low ROUGE score discourages this action. Our RL with a sentence-level agent is a novel attempt in neural summarization. We use RL as a saliency guide without altering the abstractor's language model, while previous work applied RL on the word-level, which could be prone to gaming the metric at the cost of language fluency.[7]

Learning how many sentences to extract: In a typical RL setting like game playing, an episode is usually terminated by the environment. On the other hand, in text summarization, the agent does not know in advance how many summary sentence to produce for a given article (since the desired length varies for different downstream applications). We make an important yet simple, intuitive adaptation to solve this: by adding a 'stop' action to the policy action space. In the RL training phase, we add another set of trainable parameters v_{EOE} (EOE stands for 'End-Of-Extraction') with the same dimension as the sentence representation. The pointer-network decoder treats v_{EOE} as one of the extraction candidates and hence naturally results in a stop action in the stochastic policy. We set the reward for the agent performing EOE to ROUGE-1$_{F_1}([\{g(d_{j_t})\}_t], [\{s_t\}_t])$; whereas for any extraneous, unwanted extraction step, the agent receives zero reward. The model is therefore encouraged to extract when there are still remaining ground-truth summary sentences (to accumulate intermediate reward), and learn to stop by optimizing a global ROUGE and avoiding extra extraction.[8] Overall, this modification allows dynamic decisions of number-of-sentences based on the input document, eliminates the need for tuning a fixed number of steps, and enables a data-driven adaptation for any specific dataset/application.

3.3 Repetition-Avoiding Reranking

Existing abstractive summarization systems on long documents suffer from generating repeating and redundant words and phrases. To mitigate this issue, See et al. (2017) propose the coverage mechanism and Paulus et al. (2018) incorporate tri-gram avoidance during beam-search at test-time. Our model without these already performs well because the summary sentences are generated from mutually exclusive document sentences, which naturally avoids redundancy. However, we do get a small further boost to the summary quality by removing a few 'across-sentence' repetitions, via a simple reranking strategy: At sentence-level, we apply the same beam-search tri-gram avoidance (Paulus et al., 2018). We keep all k sentence candidates generated by beam search, where k is the size of the beam. Next, we then rerank all k^n combinations of the n generated summary sentence beams. The summaries are reranked by the number of repeated N-grams, the smaller the better. We also apply the diverse decoding algorithm described in Li et al. (2016) (which has almost no computation overhead) so as to get the above approach to produce useful diverse reranking lists. We show how much the redundancy affects the summarization task in Sec. 6.2.

4 Related Work

Early summarization works mostly focused on extractive and compression based methods (Jing and McKeown, 2000; Knight and Marcu, 2000; Clarke and Lapata, 2010; Berg-Kirkpatrick et al., 2011; Filippova et al., 2015). Recent large-sized corpora attracted neural methods for abstractive summarization (Rush et al., 2015; Chopra et al., 2016). Some of the recent success in neural abstractive models include hierarchical attention (Nallapati et al., 2016), coverage (Suzuki and Nagata, 2016; Chen et al., 2016; See et al., 2017), RL based metric optimization (Paulus et al., 2018), graph-based attention (Tan et al., 2017), and the copy mechanism (Miao and Blunsom, 2016; Gu et al., 2016; See et al., 2017).

[7] During this RL training of the extractor, we keep the abstractor parameters fixed. Because the input sentences for the abstractor are extracted by an intermediate stochastic policy of the extractor, it is impossible to find the correct target summary for the abstractor to fit g with ML objective. Though it is possible to optimize the abstractor with RL, in out preliminary experiments we found that this does not improve the overall ROUGE, most likely because this RL optimizes at a sentence-level and can add across-sentence redundancy. We achieve SotA results without this abstractor-level RL.

[8] We use ROUGE-1 for terminal reward because it is a better measure of bag-of-words information (i.e., has all the important information been generated); while ROUGE-L is used as intermediate rewards since it is known for better measurement of language fluency within a local sentence.

Our model shares some high-level intuition with extract-then-compress methods. Earlier attempts in this paradigm used Hidden Markov Models and rule-based systems (Jing and McKeown, 2000), statistical models based on parse trees (Knight and Marcu, 2000), and integer linear programming based methods (Martins and Smith, 2009; Gillick and Favre, 2009; Clarke and Lapata, 2010; Berg-Kirkpatrick et al., 2011). Recent approaches investigated discourse structures (Louis et al., 2010; Hirao et al., 2013; Kikuchi et al., 2014; Wang et al., 2015), graph cuts (Qian and Liu, 2013), and parse trees (Li et al., 2014; Bing et al., 2015). For neural models, Cheng and Lapata (2016) used a second neural net to select words from an extractor's output. Our abstractor does not merely 'compress' the sentences but generatively produce novel words. Moreover, our RL bridges the extractor and the abstractor for end-to-end training.

Reinforcement learning has been used to optimize the non-differential metrics of language generation and to mitigate exposure bias (Ranzato et al., 2016; Bahdanau et al., 2017). Henß et al. (2015) use Q-learning based RL for extractive summarization. Paulus et al. (2018) use RL policy gradient methods for abstractive summarization, utilizing sequence-level metric rewards with curriculum learning (Ranzato et al., 2016) or weighted ML+RL mixed loss (Paulus et al., 2018) for stability and language fluency. We use sentence-level rewards to optimize the extractor while keeping our ML trained abstractor decoder fixed, so as to achieve the best of both worlds.

Training a neural network to use another fixed network has been investigated in machine translation for better decoding (Gu et al., 2017a) and real-time translation (Gu et al., 2017b). They used a fixed pretrained *translator* and applied policy gradient techniques to train another task-specific network. In question answering (QA), Choi et al. (2017) extract one sentence and then generate the answer from the sentence's vector representation with RL bridging. Another recent work attempted a new coarse-to-fine attention approach on summarization (Ling and Rush, 2017) and found desired sharp focus properties for scaling to larger inputs (though without metric improvements). Very recently (concurrently), Narayan et al. (2018) use RL for ranking sentences in pure extraction-based summarization and Çelikyilmaz et al. (2018) investigate multiple communicating encoder agents to enhance the copying abstractive summarizer.

Finally, there are some loosely-related recent works: Zhou et al. (2017) proposed *selective gate* to improve the attention in abstractive summarization. Tan et al. (2018) used an *extract-then-synthesis* approach on QA, where an extraction model predicts the important spans in the passage and then another synthesis model generates the final answer. Swayamdipta et al. (2017) attempted cascaded non-recurrent small networks on extractive QA, resulting a scalable, parallelizable model. Fan et al. (2017) added controlling parameters to adapt the summary to length, style, and entity preferences. However, none of these used RL to bridge the non-differentiability of neural models.

5 Experimental Setup

Please refer to the supplementary for full training details (all hyperparameter tuning was performed on the validation set). We use the CNN/Daily Mail dataset (Hermann et al., 2015) modified for summarization (Nallapati et al., 2016). Because there are two versions of the dataset, original text and entity anonymized, we show results on both versions of the dataset for a fair comparison to prior works. The experiment runs training and evaluation for each version separately. Despite the fact that the 2 versions have been considered separately by the summarization community as 2 different datasets, we use same hyper-parameter values for both dataset versions to show the generalization of our model. We also show improvements on the DUC-2002 dataset in a *test-only* setup.

5.1 Evaluation Metrics

For all the datasets, we evaluate standard ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) on full-length F_1 (with stemming) following previous works (Nallapati et al., 2017; See et al., 2017; Paulus et al., 2018). Following See et al. (2017), we also evaluate on METEOR (Denkowski and Lavie, 2014) for a more thorough analysis.

5.2 Modular Extractive vs. Abstractive

Our hybrid approach is capable of both extractive and abstractive (i.e., rewriting every sentence) summarization. The extractor alone performs extractive summarization. To investigate the effect of the recurrent extractor (rnn-ext), we implement a feed-forward extractive baseline ff-ext (details in supplementary). It is also possible to apply RL

Models	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
Extractive Results				
lead-3 (See et al., 2017)	40.34	17.70	36.57	22.21
Narayan et al. (2018)	40.0	18.2	36.6	-
ff-ext	40.63	18.35	36.82	**22.91**
rnn-ext	40.17	18.11	36.41	22.81
rnn-ext + RL	**41.47**	**18.72**	**37.76**	22.35
Abstractive Results				
See et al. (2017) (w/o coverage)	36.44	15.66	33.42	16.65
See et al. (2017)	39.53	17.28	36.38	18.72
Fan et al. (2017) (controlled)	39.75	17.29	36.54	-
ff-ext + abs	39.30	17.02	36.93	20.05
rnn-ext + abs	38.38	16.12	36.04	19.39
rnn-ext + abs + RL	40.04	17.61	37.59	**21.00**
rnn-ext + abs + RL + rerank	**40.88**	**17.80**	**38.54**	20.38

Table 1: Results on the original, non-anonymized CNN/Daily Mail dataset. Adding RL gives statistically significant improvements for all metrics over non-RL rnn-ext models (and over the state-of-the-art See et al. (2017)) in both extractive and abstractive settings with $p < 0.01$. Adding the extra reranking stage yields statistically significant better results in terms of all ROUGE metrics with $p < 0.01$.

to extractor without using the abstractor (rnn-ext + RL).[9] Benefiting from the high modularity of our model, we can make our summarization system abstractive by simply applying the abstractor on the extracted sentences. Our abstractor rewrites each sentence and generates novel words from a large vocabulary, and hence every word in our overall summary is generated from scratch; making our full model categorized into the abstractive paradigm.[10] We run experiments on separately trained extractor/abstractor (ff-ext + abs, rnn-ext + abs) and the reinforced full model (rnn-ext + abs + RL) as well as the final reranking version (rnn-ext + abs + RL + rerank).

6 Results

For easier comparison, we show separate tables for the original-text vs. anonymized versions – Table 1 and Table 2, respectively. Overall, our model achieves strong improvements and the new state-of-the-art on both extractive and abstractive settings for both versions of the CNN/DM dataset (with some comparable results on the anonymized version). Moreover, Table 3 shows the generalization of our abstractive system to an out-of-domain test-only setup (DUC-2002), where our model achieves better scores than See et al. (2017).

6.1 Extractive Summarization

In the extractive paradigm, we compare our model with the extractive model from Nallapati et al. (2017) and a strong lead-3 baseline. For producing our summary, we simply concatenate the extracted sentences from the extractors. From Table 1 and Table 2, we can see that our feed-forward extractor out-performs the lead-3 baseline, empirically showing that our hierarchical sentence encoding model is capable of extracting salient sentences.[11] The reinforced extractor performs the best, because of the ability to get the summary-level reward and the reduced train-test mismatch of feeding the previous extraction decision. The improvement over lead-3 is consistent across both tables. In Table 2, it outperforms the previous best neural extractive model (Nallapati et al., 2017). In Table 1, our model also outperforms a recent, con-

Models	R-1	R-2	R-L
Extractive Results			
lead-3 (Nallapati et al., 2017)	39.2	15.7	35.5
Nallapati et al. (2017)	39.6	16.2	35.3
ff-ext	39.51	**16.85**	35.80
rnn-ext	38.97	16.65	35.32
rnn-ext + RL	**40.13**	16.58	**36.47**
Abstractive Results			
Nallapati et al. (2016)	35.46	13.30	32.65
Fan et al. (2017) (controlled)	38.68	15.40	35.47
Paulus et al. (2018) (ML)	38.30	14.81	35.49
Paulus et al. (2018) (RL+ML)	**39.87**	15.82	36.90
ff-ext + abs	38.73	15.70	36.33
rnn-ext + abs	37.58	14.68	35.24
rnn-ext + abs + RL	38.80	15.66	36.37
rnn-ext + abs + RL + rerank	39.66	**15.85**	**37.34**

Table 2: ROUGE for anonymized CNN/DM.

[9] In this case the abstractor function $g(d) = d$.

[10] Note that the abstractive CNN/DM dataset does *not* include any human-annotated extraction label, and hence our models do not receive any direct extractive supervision.

[11] The ff-ext model outperforms rnn-ext possibly because it does not predict sentence ordering; thus is easier to optimize and the n-gram based metrics do not consider sentence ordering. Also note that in our MDP formulation, we cannot apply RL on ff-ext due to its historyless nature. Even if applied naively, there is no mean for the feed-forward model to learn the EOE described in Sec. 3.2.

Models	R-1	R-2	R-L
See et al. (2017)	37.22	15.78	33.90
rnn-ext + abs + RL	39.46	17.34	36.72

Table 3: Generalization to DUC-2002 (F1).

current work by Narayan et al. (2018), showing that our pointer-network extractor and reward formulations are very effective when combined with A2C RL.

6.2 Abstractive Summarization

After applying the abstractor, the ff-ext based model still out-performs the rnn-ext model. Both combined models exceed the pointer-generator model (See et al., 2017) without coverage by a large margin for all metrics, showing the effectiveness of our 2-step hierarchical approach: our method naturally avoids repetition by extracting multiple sentences with different keypoints.[12]

Moreover, after applying reinforcement learning, our model performs better than the best model of See et al. (2017) and the best ML trained model of Paulus et al. (2018). Our reinforced model outperforms the ML trained rnn-ext + abs baseline with statistical significance of $p < 0.01$ on all metrics for both version of the dataset, indicating the effectiveness of the RL training. Also, rnn-ext + abs + RL is statistically significant better than See et al. (2017) for all metrics with $p < 0.01$.[13] In the supplementary, we show the learning curve of our RL training, where the average reward goes up quickly after the extractor learns the End-of-Extract action and then stabilizes. For all the above models, we use standard greedy decoding and find that it performs well.

Reranking and Redundancy Although the extract-then-abstract approach inherently will not generate repeating sentences like other neural-decoders do, there might still be across-sentence redundancy because the abstractor is not aware of other extracted sentences when decoding one. Hence, we incorporate an optional reranking strategy described in Sec. 3.3. The improved ROUGE scores indicate that this successfully removes some remaining redundancies and hence produces more concise summaries. Our best abstractive model (rnn-ext + abs + RL + rerank) is clearly superior than the one of See et al. (2017). We are comparable on R-1 and R-2 but a 0.4 point improvement on R-L w.r.t. Paulus et al. (2018).[14] We also outperform the results of Fan et al. (2017) on both original and anonymized dataset versions. Several previous works have pointed out that extractive baselines are very difficult to beat (in terms of ROUGE) by an abstractive system (See et al., 2017; Nallapati et al., 2017). Note that our best model is one of the first abstractive models to outperform the lead-3 baseline on the original-text CNN/DM dataset. Our extractive experiment serves as a complementary analysis of the effect of RL with extractive systems.

	Relevance	Readability	Total
See et al. (2017)	120	128	248
rnn-ext + abs + RL + rerank	**137**	**133**	**270**
Equally good/bad	43	39	82

Table 4: Human Evaluation: pairwise comparison between our final model and See et al. (2017).

6.3 Human Evaluation

We also conduct human evaluation to ensure robustness of our training procedure. We measure *relevance* and *readability* of the summaries. Relevance is based on the summary containing important, salient information from the input article, being correct by avoiding contradictory/unrelated information, and avoiding repeated/redundant information. Readability is based on the summarys fluency, grammaticality, and coherence. To evaluate both these criteria, we design the following Amazon MTurk experiment: we randomly select 100 samples from the CNN/DM test set and ask the human testers (3 for each sample) to rank between summaries (for relevance and readability) produced by our model and that of See et al. (2017) (the models were anonymized and randomly shuffled), i.e. A is better, B is better, both are equally good/bad. Following previous work, the input article and ground truth summaries are also shown to the human participants in addition to the two model summaries.[15] From the results shown in Table 4, we can see that our model is better in both relevance and readability w.r.t. See et al. (2017).

[12] A trivial *lead-3 + abs* baseline obtains ROUGE of (37.37, 15.59, 34.82), which again confirms the importance of our reinforce-based sentence selection.

[13] We calculate statistical significance based on the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1994) with 100K samples. Output of Paulus et al. (2018) is not available so we couldn't test for statistical significance there.

[14] We do not list the scores of their pure RL model because they discussed its bad readability.

[15] We selected human annotators that were located in the US, had an approval rate greater than 95%, and had at least 10,000 approved HITs on record.

	Speed	
Models	total time (hr)	words / sec
(See et al., 2017)	12.9	14.8
rnn-ext + abs + RL	0.68	361.3
rnn-ext + abs + RL + rerank	2.00 (1.46 +0.54)	109.8

Table 5: Speed comparison with See et al. (2017).

	Novel N-gram (%)			
Models	1-gm	2-gm	3-gm	4-gm
See et al. (2017)	0.1	2.2	6.0	9.7
rnn-ext + abs + RL + rerank	0.3	10.0	21.7	31.6
reference summaries	10.8	47.5	68.2	78.2

Table 6: Abstractiveness: novel n-gram counts.

6.4 Speed Comparison

Our two-stage extractive-abstractive hybrid model is not only the SotA on summary quality metrics, but more importantly also gives a significant speed-up in both train and test time over a strong neural abstractive system (See et al., 2017).[16]

Our full model is composed of a extremely fast extractor and a parallelizable abstractor, where the computation bottleneck is on the abstractor, which has to generate summaries with a large vocabulary from scratch.[17] The main advantage of our abstractor at decoding time is that we can first compute all the extracted sentences for the document, and then abstract every sentence concurrently (*in parallel*) to generate the overall summary. In Table 5, we show the substantial test-time speed-up of our model compared to See et al. (2017).[18] We calculate the total decoding time for producing all summaries for the test set.[19] Due to the fact that the main test-time speed bottleneck of RNN language generation model is that the model is constrained to generate one word at a time, the total decoding time is dependent on the number of total words generated; we hence also report the decoded words per second for a fair comparison. Our model without reranking is extremely fast. From Table 5 we can see that we achieve a speed up of 18x in time and 24x in word generation rate. Even after adding the (optional) reranker, we still maintain a 6-7x speed-up (and hence a user can choose to use the reranking component depending on their downstream application's speed requirements).[20]

[16]The only publicly available code with a pretrained model for neural summarization which we can test the speed.

[17]The time needed for extractor is negligible w.r.t. the abstractor because it does not require large matrix multiplication for generating every word. Moreover, with convolutional encoder at word-level made parallelizable by the hierarchical rnn-ext, our model is scalable for very long documents.

[18]For details of training speed-up, please see the supp.

[19]We time the model of See et al. (2017) using beam size of 4 (used for their best-reported scores). Without beam-search, it gets significantly worse ROUGE of (36.62, 15.12, 34.08), so we do not compare speed-ups w.r.t. that version.

[20]Most of the recent neural abstractive summarization systems are of similar algorithmic complexity to that of See et al. (2017). The main differences such as the training objective (ML vs. RL) and copying (soft/hard) has negligible test runtime compared to the slowest component: the long-summary attentional-decoder's sequential generation; and this is the component that we substantially speed up via our parallel sentence decoding with sentence-selection RL.

7 Analysis

7.1 Abstractiveness

We compute an abstractiveness score (See et al., 2017) as the ratio of novel n-grams in the generated summary that are not present in the input document. The results are shown in Table 6: our model rewrites substantially more abstractive summaries than previous work. A potential reason for this is that when trained with individual sentence-pairs, the abstractor learns to drop more document words so as to write individual summary sentences as concise as human-written ones; thus the improvement in multi-gram novelty.

7.2 Qualitative Analysis on Output Examples

We show examples of how our best model selects sentences and then rewrites them. In the supplementary Figure 2 and Figure 3, we can see how the abstractor rewrites the extracted sentences concisely while keeping the mentioned facts. Adding the reranker makes the output more compact globally. We observe that when rewriting longer text, the abstractor would have many facts to choose from (Figure 3 sentence 2) and this is where the reranker helps avoid redundancy across sentences.

8 Conclusion

We propose a novel sentence-level RL model for abstractive summarization, which makes the model aware of the word-sentence hierarchy. Our model achieves the new state-of-the-art on both CNN/DM versions as well a better generalization on test-only DUC-2002, along with a significant speed-up in training and decoding.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported by a Google Faculty Research Award, a Bloomberg Data Science Research Grant, an IBM Faculty Award, and NVidia GPU awards.

References

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron C. Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 318–325, Stroudsburg, PA, USA. Association for Computational Linguistics.

Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. 2017. Neural combinatorial optimization with reinforcement learning. *arXiv preprint 1611.09940*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 481–490, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J. Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. In *ACL*.

Asli Çelikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. *NAACL-HLT*.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *IJCAI*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 484–494, Berlin, Germany. Association for Computational Linguistics.

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 209–220. Association for Computational Linguistics.

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. Association for Computational Linguistics.

James Clarke and Mirella Lapata. 2010. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

Angela Fan, David Grangier, and Michael Auli. 2017. Controllable abstractive summarization. *arXiv preprint*, abs/1711.05217.

Katja Filippova, Enrique Alfonseca, Carlos Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP'15)*.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiatao Gu, Kyunghyun Cho, and Victor O. K. Li. 2017a. Trainable greedy decoding for neural machine translation. In *EMNLP*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

Jiatao Gu, Graham Neubig, Kyunghyun Cho, and Victor O. K. Li. 2017b. Learning to translate in real-time with neural machine translation. In *EACL*.

Sebastian Henß, Margot Mieskes, and Iryna Gurevych. 2015. A reinforcement learning approach for adaptive single- and multi-document summarization. In *International Conference of the German Society for Computational Linguistics and Language Technology (GSCL-2015)*, pages 3–12.

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

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-document summarization as a tree knapsack problem. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1515–1520, Seattle, Washington, USA. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(9):1735–1780.

Hongyan Jing and Kathleen R. McKeown. 2000. Cut and paste based text summarization. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, NAACL 2000, pages 178–185, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 315–320, Baltimore, Maryland. Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710. AAAI Press.

Chen Li, Yang Liu, Fei Liu, Lin Zhao, and Fuliang Weng. 2014. Improving multi-documents summarization by sentence compression based on expanded constituent parse trees. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 691–701. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint*, abs/1611.08562.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*, pages 33–42. Association for Computational Linguistics.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse indicators for content selection in summarization. In *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '10, pages 147–156, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

André F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.

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

Ramesh Nallapati, Bowen Zhou, Cicero Nogueira dos santos, Caglar Gulcehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *CoNLL*.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Ranking sentences for extractive summarization with reinforcement learning. *NAACL-HLT*.

Eric W Noreen. 1989. *Computer-intensive methods for testing hypotheses*. Wiley New York.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *ICLR*.

Xian Qian and Yang Liu. 2013. Fast joint compression and summarization via graph cuts. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1492–1502, Seattle, Washington, USA. Association for Computational Linguistics.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

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*, pages 379–389, Lisbon, Portugal. Association for Computational Linguistics.

Mike Schuster, Kuldip K. Paliwal, and A. General. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083. Association for Computational Linguistics.

Jun Suzuki and Masaaki Nagata. 2016. Rnn-based encoder-decoder approach with word frequency estimation. In *EACL*.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. 2017. Multi-mention learning for reading comprehension with neural cascades. *arXiv preprint*, abs/1711.00894.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2018. S-net: From answer extraction to answer generation for machine reading comprehension. In *AAAI*.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *ACL*.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. 2016. Order matters: Sequence to sequence for sets. In *International Conference on Learning Representations (ICLR)*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc.

Xun Wang, Yasuhisa Yoshida, Tsutomu Hirao, Katsuhito Sudoh, and Masaaki Nagata. 2015. Summarization based on task-oriented discourse parsing. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 23(8):1358–1367.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256.

David Zajic, Bonnie Dorr, and Richard Schwartz. 2004. Bbn/umd at duc-2004: Topiary. In *HLT-NAACL 2004 Document Understanding Workshop*, pages 112–119, Boston, Massachusetts.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1095–1104. Association for Computational Linguistics.