A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents

Arman Cohan[†] **Franck Dernoncourt**[⋆] **Doo Soon Kim**[⋆] **Trung Bui**[⋆]
Seokhwan Kim[⋆] **Walter Chang**[⋆] **Nazli Goharian**[†]

[†]Department of Computer Science, Georgetown University, Washington, DC
{arman,nazli}@ir.cs.georgetown.edu

[⋆]Adobe Research, San Jose, CA
{dernonco,dkim,bui,seokim,wachang}@adobe.com

Abstract

Neural abstractive summarization models have led to promising results in summarizing relatively short documents. We propose the first model for abstractive summarization of single, longer-form documents (e.g., research papers). Our approach consists of a new hierarchical encoder that models the discourse structure of a document, and an attentive discourse-aware decoder to generate the summary. Empirical results on two large-scale datasets of scientific papers show that our model significantly outperforms state-of-the-art models.

1 Introduction

Existing large-scale summarization datasets consist of relatively short documents. For example, articles in the CNN/Daily Mail dataset (Hermann et al., 2015) are on average about 600 words long. Similarly, existing neural summarization models have focused on summarizing sentences and short documents. In this work, we propose a model for effective abstractive summarization of longer documents. Scientific papers are an example of documents that are significantly longer than news articles (see Table 1). They also follow a standard discourse structure describing the problem, methodology, experiments/results, and finally conclusions (Suppe, 1998).

Most summarization works in the literature focus on extractive summarization. Examples of prominent approaches include frequency-based methods (Vanderwende et al., 2007), graph-based methods (Erkan and Radev, 2004), topic modeling (Steinberger and Jezek, 2004), and neural models (Nallapati et al., 2017). Abstractive summarization is an alternative approach where the generated summary may contain novel words and phrases and is more similar to how humans summarize documents (Jing, 2002). Recently, neural methods have led to encouraging results in abstractive summarization (Nallapati et al., 2016; See et al., 2017; Paulus et al., 2017; Li et al., 2017). These approaches employ a general framework of sequence-to-sequence (seq2seq) models (Sutskever et al., 2014) where the document is fed to an encoder network and another (recurrent) network learns to decode the summary. While promising, these methods focus on summarizing news articles which are relatively short. Many other document types, however, are longer and structured. Seq2seq models tend to struggle with longer sequences because at each decoding step, the decoder needs to learn to construct a context vector capturing relevant information from all the tokens in the source sequence (Shao et al., 2017).

Our main contribution is an abstractive model for summarizing scientific papers which are an example of long-form structured document types. Our model includes a hierarchical encoder, capturing the discourse structure of the document and a discourse-aware decoder that generates the summary. Our decoder attends to different discourse sections and allows the model to more accurately represent important information from the source resulting in a better context vector. We also introduce two large-scale datasets of long and structured scientific papers obtained from arXiv and PubMed to support both training and evaluating models on the task of long document summarization. Evaluation results show that our method outperforms state-of-the-art summarization models[1].

2 Background

In the seq2seq framework for abstractive summarization, an input document $\mathbf{x}$ is encoded using a Recurrent Neural Network (RNN) with $\mathbf{h}_i^{(e)}$ being the hidden state of the encoder at timestep i. The last step of the encoder is fed as input to another RNN which decodes the output one token

[1] Data/code: https://github.com/acohan/long-summarization

Proceedings of NAACL-HLT 2018, pages 615–621
New Orleans, Louisiana, June 1 - 6, 2018.

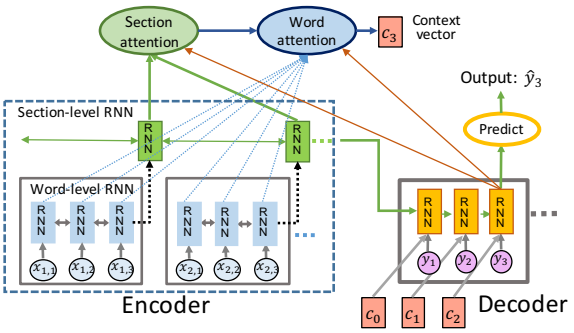

Figure 1: Overview of our model. The word-level RNN is shown in blue and section-level RNN is shown in green. The decoder also consists of an RNN (orange) and a "predict" network for generating the summary. At each decoding time step t (here t=3 is shown), the decoder forms a context vector c_t which encodes the relevant source context (c_0 is initialized as a zero vector). Then the section and word attention weights are respectively computed using the green "section attention" and the blue "word attention" blocks. The context vector is used as another input to the decoder RNN and as an input to the "predict" network which outputs the next word using a joint pointer-generator network.

at a time. Given an input document along with the corresponding ground-truth summary $\mathbf{y}$, the model is trained to output a summary $\hat{\mathbf{y}}$ that is close to $\mathbf{y}$. The output at timestep t is predicted using the decoder input $\mathbf{x}'_t$, decoder hidden state $\mathbf{h}_t^{(d)}$, and some information about the input sequence. This framework is the general seq2seq framework employed in many generation tasks including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014) and summarization (Nallapati et al., 2016; Chopra et al., 2016).

Attentive decoding The attention mechanism maps the decoder state and the encoder states to an output vector, which is a weighted sum of the encoder states and is called context vector (Bahdanau et al., 2014). Incorporating this context vector at each decoding timestep (attentive decoding) is proven effective in seq2seq models. Formally, the context vector c_t is defined as: $\mathbf{c}_t = \sum_{i=1}^{N} \alpha_i^{(t)} \mathbf{h}_i^{(e)}$ where $\alpha_i^{(t)}$ are the attention weights calculated as follows:

$$\alpha_i^{(t)} = \underset{i}{\text{softmax}}(\text{score}(\mathbf{h}_i^{(e)}, \mathbf{h}_{t-1}^{(d)})) \quad (1)$$

where $\underset{i}{\text{softmax}}$ means that the denominator's sum in the softmax function is over i. The score function can be defined in bilinear, additive, or multiplicative ways (Luong et al., 2015). We use the additive scoring function:

$$\text{score}(\mathbf{h}_i^{(e)}, \mathbf{h}_{t-1}^{(d)}) = \mathbf{v}_a^\top \tanh\left(\text{linear}(\mathbf{h}_i^{(e)}, \mathbf{h}_{t-1}^{(d)})\right) \quad (2)$$

where $\mathbf{v}_a$ is a weight vector and linear is a linear mapping function. I.e.,

$$\text{linear}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{W}_1\mathbf{x}_1 + \mathbf{W}_2\mathbf{x}_2 + \mathbf{b} \quad (3)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are weight matrices and $\mathbf{b}$ is the bias vector.

3 Model

We now describe our discourse-aware summarization model (shown in Figure 1).

Encoder Our encoder extends the RNN encoder to a hierarchical RNN that captures the document discourse structure. We first encode each discourse section and then encode the document. Formally, we encode the document as a vector $\mathbf{d}$ according to the following:

$$\mathbf{d} = \text{RNN}_{doc}\left(\{\mathbf{h}_1^{(s)}, ..., \mathbf{h}_N^{(s)}\}\right)$$

$\text{RNN}(.)$ denotes a function which is a recurrent neural network whose output is the final state of the network encoding the entire sequence. N is the number of sections in the document and $\mathbf{h}_j^{(s)}$ is representation of section j in the document consisting of a sequence of tokens.

$$\mathbf{h}_j^{(s)} = \text{RNN}_{sec}\left(\mathbf{x}_{(j,1)}, ...\mathbf{x}_{(j,M)}\}\right)$$

where $\mathbf{x}_{(j,i)}$ are dense embeddings corresponding to the tokens $w_{(j,i)}$ and M is the maximum section length. The parameters of RNN_{sec} are shared for all the discourse sections. We use a single layer bidirectional LSTM (following the LSTM formulation of Graves et al. (2013)) for both RNN_{doc} and RNN_{sec}; further extension to multilayer LSTMs is straightforward. We combine the forward and backward LSTM states to a single state using a simple feed-forward network:

$$\mathbf{h} = \text{relu}(\mathbf{W}([\overrightarrow{\mathbf{h}}, \overleftarrow{\mathbf{h}}] + \mathbf{b})$$

where $[,]$ shows the concatenation operation. Throughout, when we mention the RNN (LSTM) state, we are referring to this combined state of both forward and backward RNNs (LSTMs).

Discourse-aware decoder When humans summarize a long structured document, depending on the domain and the nature of the document, they write about important points from different discourse sections of the document. For example, scientific paper abstracts typically include the description of the problem, discussion of the methods, and finally results and conclusions (Suppe, 1998). Motivated by this observation, we propose a discourse-aware attention method. Intuitively, at each decoding timestep, in addition to the words

in the document, we also attend to the relevant discourse section (the "section attention" block in Figure 1). Then we use the discourse-related information to modify the word-level attention function. Specifically, the context vector representing the source document is:

$$\mathbf{c}_t = \sum_{j=1}^{N} \sum_{i=1}^{M} \alpha_{(j,i)}^{(t)} \mathbf{h}_{(j,i)}^{(e)} \quad (4)$$

where $\mathbf{h}_{(j,i)}^{(e)}$ shows the encoder state of word i in discourse section j and $\alpha_{(j,i)}^{(t)}$ shows the corresponding attention weight to that encoder state. The scalar weights $\alpha_{(j,i)}^{(t)}$ are obtained according to:

$$\alpha_{(j,i)}^{(t)} = \underset{(i,j)}{\text{softmax}} \left(\beta_j^{(t)} \, \text{score}(\mathbf{h}_{(j,i)}^{(e)}, \mathbf{h}_{t-1}^{(d)}) \right) \quad (5)$$

The score function is the additive attention function (Equation 2) and the weights $\beta_j^{(t)}$ are updated according to:

$$\beta_j^{(t)} = \underset{j}{\text{softmax}}(\text{score}(\mathbf{h}_j^{(s)}, \mathbf{h}_{t-1}^{(d)})) \quad (6)$$

At each timestep t, the decoder state $\mathbf{h}_t^{(d)}$ and the context vector $\mathbf{c}_t$ are used to estimate the probability distribution of next word y_t:

$$p(y_t|y_{1:t-1}) = \text{softmax}\left(\mathbf{V}^\top \text{linear}\left(\mathbf{h}_t^{(d)}, \mathbf{c}_t\right)\right) \quad (7)$$

where $\mathbf{V}$ is a vocabulary weight matrix and softmax is over the entire vocabulary.

Copying from source There has been a surge of recent works in sequence learning tasks to address the problem of *unkown* token prediction by allowing the model to occasionally copy words directly from source instead of generating a new token (Gu et al., 2016; See et al., 2017; Paulus et al., 2017; Wiseman et al., 2017). Following these works, we add an additional binary variable z_t to the decoder, indicating generating a word from vocabulary (z_t=0) or copying a word from the source (z_t=1). The probability is learnt during training according to the following equation:

$$p(z_t{=}1|y_{1:t-1}) = \sigma(\text{linear}(\mathbf{h}_t^{(d)}, \mathbf{c}_t, \mathbf{x}'_t)) \quad (8)$$

Then the next word y_t is generated according to:

$$p(y_t|y_{1:t-1}) = \sum_z p(y_t, z_t{=}z|y_{1:t-1}); z = \{0, 1\}$$

The joint probability is decomposed as:

$$p(y_t, z_t{=}z) = \begin{cases} p_c(y_t|y_{1:t-1})\; p(z_t{=}z|y_{1:t-1}), & z{=}1 \\ p_g(y_t|y_{1:t-1})\; p(z_t{=}z|y_{1:t-1}), & z{=}0 \end{cases}$$

p_g is the probability of generating a word from the vocabulary and is defined according to Equation 7. p_c is the probability of copying a word from the source vector $\mathbf{x}$ and is defined as the sum of the word's attention weights. Specifically, the probability of copying a word x_ℓ is defined as:

$$p_c(y_t = x_\ell|y_{1:t-1}) = \sum_{(j,i):x_{(j,i)}=x_\ell} \alpha_{(j,i)}^{(t)} \quad (9)$$

Decoder coverage In long sequences, the neural generation models tend to repeat phrases where the softmax layer predicts the same phrase multiple times over multiple timesteps. To address this issue, following See et al. (2017), we track attention coverage to avoid repeatedly attending to the same steps. This is done with a coverage vector $\mathbf{cov}^{(t)}$, the sum of attention weight vectors at previous timesteps: $\text{cov}_{(j,i)}^{(t)} = \sum_{k=0}^{t-1} \alpha_{(j,i)}^{(k)}$

The coverage implicitly includes information about the attended document discourse sections. We incorporate the decoder coverage as an additional input to the attention function:

$$\alpha_{(j,i)}^{(t)} = \underset{(i,j)}{\text{softmax}} \left(\beta_j^{(t)} \, \text{score}(\mathbf{h}_{(j,i)}^{(e)}, \text{cov}_{(j,i)}^{(t)}, \mathbf{h}_{t-1}^{(d)}) \right)$$

4 Related work

Neural abstractive summarization models have been studied in the past (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016) and later extended by source copying (Miao and Blunsom, 2016; See et al., 2017), reinformcement learning (Paulus et al., 2017), and sentence salience information (Li et al., 2017). One model variant of Nallapati et al. (2016) is related to our model in using sentence-level information in attention. However, our model is different as it contains a hierarchical encoder, uses discourse sections in the decoding step, and has a coverage mechanism. Similarly, Ling and Rush (2017) proposed a coarse-to-fine attention model that uses hard attention to find the text chunks of importance and then only attend to words in that chunk. In contrast, we consider all the discourse sections using soft attention. The closest model to ours is that of See et al. (2017) and Paulus et al. (2017) who used a joint pointer-generator network for summarization. However, our model extends theirs by *(i)* a hierarchical encoder for modeling long documents and *(ii)* a discourse-aware decoder that captures the information flow from all discourse sections of the document. Finally, in a recent work, Liu et al. (2018) proposed a model based on the transformer network (Vaswani et al., 2017) for abstractive generation of Wikipedia articles. However, their focus

Datasets	# docs	avg. doc. length (words)	avg. summary length (words)
CNN	92K	656	43
Daily Mail	219K	693	52
NY Times	655K	530	38
PubMed (this work)	133K	3016	203
arXiv (this work)	215K	4938	220

Table 1: Statistics of our arXiv and PubMed datasets compared with existing large-scale summarization corpora, CNN and Daily Mail (Nallapati et al., 2016) and NY Times (Paulus et al., 2017).

is on multi-document summarization.

Our datasets are obtained from scientific papers. Scientific document summarization has been recently received extended attention (Qazvinian et al., 2013; Cohan and Goharian, 2015, 2017b,a). In contrast to ours, existing approaches are extractive and rely on external information such as citations, which may not be available for all papers.

5 Data

Seq2seq models typically have a large number of parameters and thus they require large training data with ground truth summaries. Researchers have constructed such training data from news articles (e.g., CNN, Daily Mail and New York Times articles), where the abstracts or highlights of news articles are considered as ground truth summaries (Nallapati et al., 2016; Paulus et al., 2017). However, news articles are relatively short and not suitable for the task of long-from document summarization. Following these works, we take scientific papers as an example of long documents with discourse information, where their abstracts can be used as ground-truth summaries. We introduce two datasets collected from scientific repositories, arXiv.org and PubMed.com.

The choice of scientific papers for our dataset is motivated by the fact that scientific papers are examples of long documents that follow a standard discourse structure and they already come with ground truth summaries, making it possible to train supervised neural models. We follow existing work in constructing large-scale summarization datasets that take news article abstracts as ground truth.

We remove the documents that are excessively long (e.g., theses) or too short (e.g., tutorial announcements), or do not have an abstract or discourse structure. We use the level-1 section headings as the discourse information. For arXiv, we use the LaTeX files and convert them to plain text using Pandoc (https://pandoc.org) to preserve the discourse section information. We remove figures and tables using regular expressions to only preserve the textual information. We also normalize math formulas and citation markers with special tokens. We analyze the document section names and identify the most common concluding sections names (e.g. *conclusion*, *concluding remarks*, *summary*, etc). We only keep the sections up to the conclusion section of the document and we remove sections after the conclusion.

The statistics of our datasets are shown in Table 1. In our datasets, both document and summary lengths are significantly larger than the existing large-scale summarization datasets. We retain about 3% (5%) of PubMed (ArXiv) as validation data and about another 3% (5%) for test; the rest is used for training.

6 Experiments

Setup Similar to the majority of published research in the summarization literature (Chopra et al., 2016; Nallapati et al., 2016; See et al., 2017), evaluation was done using the ROUGE automatic summarization evaluation metric (Lin, 2004) with full-length F-1 ROUGE scores. We lowercase all tokens and perform sentence and word tokenization using spaCy (Honnibal and Johnson, 2015).

Implementation details We use Tensorflow 1.4 for implementing our models. We use the hyperparameters suggested by See et al. (2017). In particular, we use two bidirectional LSTMs with cell size of 256 and embedding dimensions of 128. Embeddings are trained from scratch and we did not find any gain using pre-trained embeddings. The vocabulary size is constrained to 50,000; using larger vocabulary size did not result in any improvement. We use mini-batches of size 16 and we limit the document length to 2000 and section length to 500 tokens, and number of sections to 4. We use batch-padding and dynamic unrolling to handle variable sequence lengths in LSTMs. Training was done using Adagrad optimizer with learning rate 0.15 and an initial accumulator value of 0.1. The maximum decoder size was 210 tokens which is in line with average abstract length in our datasets. We first train the model without coverage and added it at the last two epochs to help the model converge faster. We train the models on NVIDIA Titan X Pascal GPUs. Training is performed for about 10 epochs and each training step takes about 3.2 seconds. We used beam search at

	Summarizer	RG-1	RG-2	RG-3	RG-L
Extractive	SumBasic	29.47	6.95	2.36	26.30
	LexRank	33.85	10.73	**4.54**	28.99
	LSA	29.91	7.42	3.12	25.67
Abstractive	Attn-Seq2Seq	29.30	6.00	1.77	25.56
	Pntr-Gen-Seq2Seq	32.06	9.04	2.15	25.16
	This work	†‡**35.80**	†**11.05**	†3.62	†‡**31.80**

Table 2: Results on the arXiv dataset, RG: ROUGE. For our method † (‡) shows statistically significant improvement with $p<0.05$ over other abstractive methods (all other methods).

	Summarizer	RG-1	RG-2	RG-3	RG-L
Extractive	SumBasic	37.15	11.36	5.42	33.43
	LexRank	**39.19**	13.89	7.27	34.59
	LSA	33.89	9.93	5.04	29.70
Abstractive	Attn-Seq2Seq	31.55	8.52	7.05	27.38
	Pntr-Gen-Seq2Seq	35.86	10.22	7.60	29.69
	This work	†38.93	†‡**15.37**	†‡**9.97**	†‡**35.21**

Table 3: Results on PubMed dataset, RG:ROUGE. For our method, † (‡) shows statistically significant improvement with $p<0.05$ over abstractive methods (all other methods).

decoding time with beam size of 4. We train the abstractive baselines for about 250K iterations as suggested by their authors.

Comparison We compare our method with several well-known extractive baselines as well as state-of-the-art abstractive models using their open-sourced implementations, when available; we follow the same training setup described in the corresponding papers. The compared methods are: *LexRank* (Erkan and Radev, 2004), *SumBasic* (Vanderwende et al., 2007), *LSA* (Steinberger and Jezek, 2004), *Attn-Seq2Seq* (Nallapati et al., 2016; Chopra et al., 2016), *Pntr-Gen-Seq2Seq* (See et al., 2017). The first three are extractive models and last two are abstractive. *Pntr-Gen-Seq2Seq* extends *Attn-Seq2Seq* by using a joint pointer network during decoding. For *Pntr-Gen-Seq2Seq* we use their reported hyperparameters to ensure that the result differences are not due to hyperparameter tuning.

Results Our main results are shown in Tables 2 and 3. Our model significantly outperforms the state-of-the-art abstractive methods, showing its effectiveness on both datasets. We observe that in our ROUGE-1 score is respectively about 4 and 3 points higher than the abstractive model *Pntr-Gen-Seq2Seq* for the arXiv and PubMed datasets, providing a significant improvement. Our method also outperforms most of the extractive methods except for *LexRank* in one of the ROUGE scores. We note that since extractive methods copy salient sentences from the document, it is usually easier for them to achieve higher ROUGE scores.

Abstract: in this paper , the author proposes a series of multilevel double hashing schemes called cascade hash tables . they use several levels of hash tables . in each table , we use the common double hashing scheme . higher level hash tables work as fail - safes of lower level hash tables . by this strategy , it could effectively reduce collisions in hash insertion . thus it gains a constant worst case lookup time with a relatively high load factor (@xmath0) in random experiments . different parameters of cascade hash tables are tested .

Pntr-Gen-Seq2Seq: hash table is a common data structure used in large set of data storage and retrieval . it has an o(1) lookup time on average , but the worst case lookup time can be as bad as . is the size of the hash table . we present a set of hash table schemes called cascade hash tables . hash table data structures which consist of several of hash tables with different size .

Our method: cascade hash tables are a common data structure used in large set of data storage and retrieval . such a time variation is essentially caused by possibly many collisions during keys hashing . in this paper , we present a set of hash schemes called cascade hash tables which consist of several levels (@xmath2) of hash tables with different size . after constant probes , if an item ca 'nt find a free slot in limited probes in any hash table , it will try to find a cell in the second level , or subsequent lower levels . with this simple strategy , these hash tables will have descendant load factors , therefore lower collision probabilities .

Figure 2: Example of a generated summary

Figure 2 illustrates the effectiveness of our model extensions in capturing various discourse information from the papers. It can be observed that the state-of-the-art *Pntr-Gen-Seq2Seq* model generates a summary that mostly focuses on introducing the problem, whereas our model generates a summary that includes more information about the methodology and impacts of the target paper. This indicates that the context vector in our model compared with *Pntr-Gen-Seq2Seq* is better able to capture important information from the source by attending to various discourse sections.

7 Conclusions and future work

This work was the first attempt at addressing neural abstractive summarization of single, long documents. We presented a neural sequence-to-sequence model that is able to effectively summarize long and structured documents such as scientific papers. While our results are encouraging, there is still much room for improvement for this challenging task; our new datasets can help the community to further explore this problem.

We note that following the convention in the summarization research, our quantitative evaluation is performed by ROUGE automatic metric. While ROUGE is an effective evaluation framework, nuances in the coherence or coverage of the summaries are not captured with it. It is non-trivial to evaluate such qualities especially for long document summarization; future work can design expert human evaluations to explore these nuances.

Acknowledgements

We thank the three anonymous reviewers for their comments and suggestions.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Sumit Chopra, Michael Auli, Alexander M Rush, and SEAS Harvard. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *HLT-NAACL*. pages 93–98.

Arman Cohan and Nazli Goharian. 2015. Scientific article summarization using citation-context and article's discourse structure. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 390–400. http://aclweb.org/anthology/D15-1045.

Arman Cohan and Nazli Goharian. 2017a. Contextualizing citations for scientific summarization using word embeddings and domain knowledge. *arXiv preprint arXiv:1705.08063* .

Arman Cohan and Nazli Goharian. 2017b. Scientific document summarization via citation contextualization and scientific discourse. *International Journal on Digital Libraries* pages 1–17. https://doi.org/10.1007/s00799-017-0216-8.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research* 22:457–479.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, pages 6645–6649.

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)*. Association for Computational Linguistics, Berlin, Germany, pages 1631–1640. http://www.aclweb.org/anthology/P16-1154.

Karl Moritz Hermann, Tomas Kocisky, 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*. pages 1693–1701.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1373–1378. https://aclweb.org/anthology/D/D15/D15-1162.

Hongyan Jing. 2002. Using hidden markov modeling to decompose human-written summaries. *Computational linguistics* 28(4):527–543.

Piji Li, Wai Lam, Lidong Bing, Weiwei Guo, and Hang Li. 2017. Cascaded attention based unsupervised information distillation for compressive summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2071–2080.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, volume 8.

Jeffrey Ling and Alexander Rush. 2017. Coarse-to-fine attention models for document summarization. In *Proceedings of the Workshop on New Frontiers in Summarization*. Association for Computational Linguistics, Copenhagen, Denmark, pages 33–42. http://www.aclweb.org/anthology/W17-4505.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hyg0vbWC-.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* .

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317* .

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *AAAI* 1:1.

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* .

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* .

Vahed Qazvinian, Dragomir R Radev, Saif M Mohammad, Bonnie Dorr, David Zajic, Michael Whidby, and Taesun Moon. 2013. Generating extractive summaries of scientific paradigms. *Journal of Artificial Intelligence Research* 46:165–201.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* .

Abigail See, Christopher Manning, and Peter Liu. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics*. https://arxiv.org/abs/1704.04368.

Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. pages 2210–2219.

Josef Steinberger and Karel Jezek. 2004. Using latent semantic analysis in text summarization and summary evaluation. In *Proc. ISIM04*. pages 93–100.

Frederick Suppe. 1998. The structure of a scientific paper. *Philosophy of Science* 65(3):381–405.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. pages 3104–3112.

Lucy Vanderwende, Hisami Suzuki, Chris Brockett, and Ani Nenkova. 2007. Beyond sumbasic: Task-focused summarization with sentence simplification and lexical expansion. *Information Processing & Management* 43(6):1606–1618.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, Curran Associates, Inc., pages 6000–6010. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052* .