Encoding Position Improves Recurrent Neural Text Summarizers

Apostolos Karanikolos and Ioannis Refanidis
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
{a.karanikolos, yrefanid}@uom.edu.gr

Abstract

Modern text summarizers are big neural networks (recurrent, convolutional, or transformers) trained end-to-end under an encoder-decoder framework. These networks equipped with an attention mechanism, that maintains a memory of their source hidden states, are able to generalize well to long text sequences. In this paper, we explore how the different modules involved in an encoder-decoder structure affect the produced summary quality as measured by ROUGE score in the widely used CNN/Daily Mail and Gigaword summarization datasets. We find that encoding the position of the text tokens before feeding them to a recurrent text summarizer gives a significant, in terms of ROUGE, gain to its performance on the former but not the latter dataset.

1 Introduction

Within NLP a number of tasks involve generating text conditioned on some input information (machine translation, image caption generation, headline generation, single and multi-document summarization).

To accomplish the task of text summarization, a system needs the ability to capture the semantic content of the source text and then predict its grammatical, faithful and coherent summary. Since the structure of the system summary has to be closely related to the structure of the input text a central challenge to this task is the problem of alignment, i.e. the problem of how to relate sub-elements of the input to sub-elements of the output (Cho et al., 2015).

Similar to a human-produced summary that intuitively is as good as the clarity of her thoughts and goals, a machine-generated summary depends heavily on the quality of its internal information. For neural network summarizers that is equivalent to strong representations of the source document and of the summary generated so far, both kept as vectors, respectively, in their encoder and decoder hidden states.

Deep learning methods, employing end-to-end trained neural network models, have recently achieved significant, although not robust, ability in generating reasonable multi-sentence abstractive summaries of long news articles. Extending the sequence-to-sequence framework, already adapted in other sequence transduction tasks, these models mostly consist of three cooperating modules, whose parameters are learned jointly through gradient descent or reinforcement learning techniques.

First, an encoder mechanism that produces hidden representations of the source document; second, an attention network that selects its salient information; and third, a decoder module that produces the model summary. This decoder module is often an autoregressive[1] network that splits high dimensional data into a sequence of small pieces and then predicts each piece from those before.

For most languages, these neural models perform summarization in a left to right manner, one word at a time, until a special stop token is generated, which ends the summary. This information processing pipeline can be seen as a four step process "embed – encode – attend – predict". In the "embed" step lexical tokens are converted from indices in a vocabulary to dense vectors, encoding distributional semantics. Then, in the "encode" step information is passed through hidden neu-

[1] Autoregressive model is one in which the prediction for every one sample is influenced by all previous ones.

ral connections (either recurrent, convolutional or feed-forward cells) building the source document matrix representation. Each row of this matrix encodes the "meaning" of each token in the context of its surrounding tokens. Next, in the "attend" step the previous matrix is reduced to a vector while ensuring this reduction comes with minimal information loss, reflecting the goal of the attention mechanism to select the most important element from each time step. The final "predict" step reduces this vector to a prediction of the next token in the summary.

Recently, convolutional (Gehring et al., 2017) and self-attentive purely feed-forward (Vaswani et al., 2017) networks have proven able to match the performance of recurrent neural networks (Chopra et al., 2016; Tan et al., 2017) in the role of encoder and decoder modules, replacing them on several sequence generation tasks (Xie, 2017). Used as summarizers, these models can produce not only general but also topic-aware (Wang et al., 2018), query-based (Hasselqvist et al., 2017), or user-controllable (Fan et al., 2018) summaries. However, in this work we choose to only focus on general summaries.

Creating summaries from documents, seen as a sequential decision making problem for the decoder-agent, is also amenable to reinforcement learning techniques. In this setting, the model at each step learns to make a decision of the next token to generate while optimizing a sequence–level objective, the full sequence ROUGE score (Lin, 2004). Here, arises the issue of the exploration-exploitation tradeoff, a problem but also an opportunity for the agent to generate a more diverse, hence more abstract and human-like summary (Chen and Bansal, 2018).

In the standard supervised setting, the model needs labeled summaries in the training phase to provide the appropriate learning signal. In an unsupervised setting, a model could potentially learn to summarize documents without having access to ground-truth summaries in the learning phase (Chu and Liu, 2019).

The quality of the produced system summaries can be rated both by automatic metrics (ROUGE, Meteor) and by human raters. Intuitively, a high quality summary should be a concise text that captures the salient and rejects the secondary information of the source document. It would use grammatical language structures and include a significant amount of novel words and phrases not found in the source text.

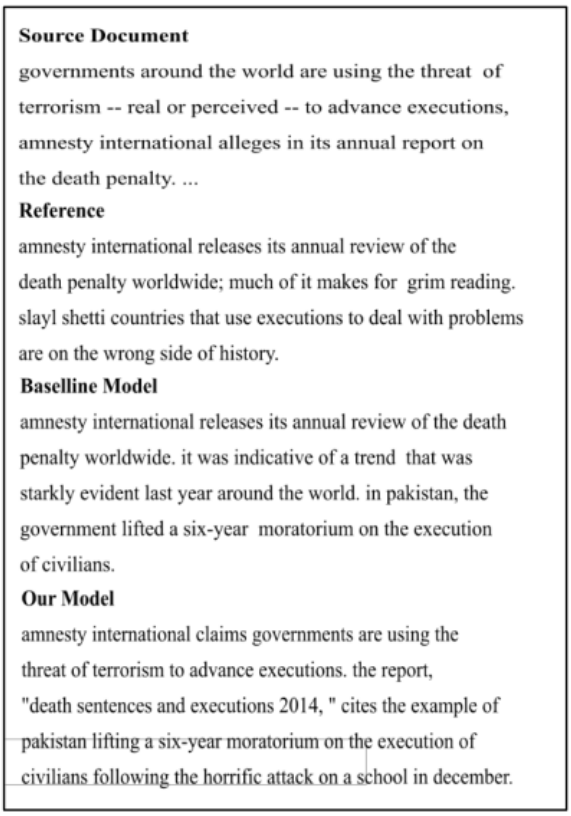

Figure 1: Example of different model generated two-sentence summaries of the same input text (source document). Reference denotes the ground-truth summary. With position encoding (our model) we see more abstractive ability, while without position encoding (baseline model) we see less paraphrasing and more copying from input text.

The key contribution of this work is the novel use of the token-position information in a recurrent neural text summarizer. We show that our neural network approach, while requiring fewer learnable parameters than a transformer model, outperforms it on the CNN/Daily Mail dataset (Hermann et al., 2015) and performs on par with it on the Gigaword corpus (Rush et al., 2017). These results suggest we do not need the computation-heavy self-attention processing of the transformer architecture in neural text summarizers.

2 Background

We describe the standard approach for supervised abstractive summarization learning based on the attentive sequence-to-sequence framework, and the challenges it faces in text representation and generation. The goal of a model under this frame-

work is to maximize the probability of generating correct target sequences.

2.1 Sequence-to-Sequence Framework

The sequence-to-sequence framework consists of two parts: a neural network for the encoder and another network for the decoder. The source text, reference summary data is tokenized and fed to the encoder and decoder networks respectively during training. The encoder network reads the source text and transforms it into a potentially useful vector representation which then passes to the decoder network to help in the prediction of the summary sequence on a token per token basis.

Encoder Mechanism: The encoder mechanism uses a deep neural network to convert a sequence of source words into a sequence of vectors representing its contextual meaning. This encoding is done using recurrent, convolutional or transformer neural networks. Word and positional embeddings can be used before feeding the source sequence to the deep neural encoder network.

Decoder Mechanism: The decoder network uses the vector representation coming out of the encoder network and its own internal state information to represent the state of the sequence generated so far. Essentially, the decoder mechanism combines specific vectorial knowledge about the relevant context with general knowledge about language generation in order to produce the output sequence. Analogous to the encoder, it can also use word and positional embeddings to the tokens it already generated, before feeding each new token to the deep neural decoder network.

2.2 Attention Mechanism

A mapping of the decoder state at each time step with all the encoder states into an attention vector, helps produce a context vector which is a weighted sum of the encoder states. Incorporating this context vector at each decoding time step helps improve text generation (Bahdanau et al., 2014).

Necessity for Attention: From a cognitive science perspective, attention, defined as the ability to focus on one thing and ignore others, allows for picking out salient information from noisy data and to remember one event rather than all events. Thus, attention is selective and appears to be as useful for deep learning as it is for people. From a sequence-to-sequence standpoint, attention is the action of focusing on specific parts of the input sequence. It can be stochastic and trained with reinforcement learning (hard attention) or differentiable and trained with back-propagation (soft attention). We note that attention changes over time. As the model generates each word, its attention changes to reflect the relevant parts of the input.

Self-Attention: When a sequence-to-sequence model is trying to generate the next word in the summary, this word is usually describing only a part of the input text. Using the whole representation of the input text (h) to condition the generation of each word cannot efficiently produce different words for different parts of the input. But, if we first divide the input into n parts, we can compute representations of each part $(h_1, ..., h_n)$. Then, when the model is generating a new word, its attention mechanism can focus on the relevant part of the input sequence, so that the model can only use specific parts of the input. This is the idea of self-attention.

2.3 Text generation

Greedy decoding: When using greedy decoding, the model at any time step has only one single hypothesis. Since a text sequence can be the most probable despite including tokens that are not the most probable at each time step, greedy decoding is seldom used in practice.

Beam decoding: When using beam search decoding the model iteratively expands each hypothesis one token at a time and in the end of each iteration it only keeps the beam-size best ones. Small beam sizes are able to yield good results in terms of ROUGE score while larger beam sizes can yield worse results. To make decoding efficient the decoder expands only hypotheses that look promising. Bad hypotheses should be pruned early to avoid wasting time on them, but pruning compromises optimality.

Challenges in text generation: In neural summary generation, a model error occurs when the summary with the highest score under the model is not a good summary, while a search error occurs when the decoder network cannot find the summary with the highest score under the model.

Other challenges include the generation of truncated or repetitive outputs, the production of blank or generic text, or ungrammatical gibberish. Rare or out of vocabulary (OOV) word generation, that naturally arises for languages with very large vocabularies can be mitigated in practice by the use

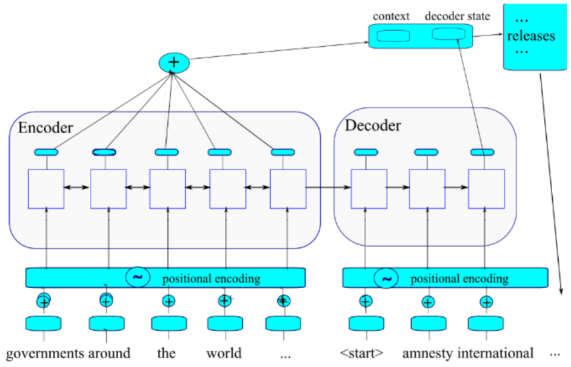

Figure 2: Model architecture. Encoder is a bi-directional 4-layer LSTM and the decoder is a uni-directional 4-layer LSTM. The vector containing encoder context and decoder state is used to compute the distribution over the output vocabulary.

of the copy mechanism, introduced later.

Another issue is the early summary termination. During the beam search procedure, hypotheses terminate with the "eos" token. The decoder should learn to place very low probability to this token until the summary is fully generated. However, sometimes "eos" does not have sufficiently low probability resulting in production of short or truncated system summaries. Length normalization, or the coverage penalty (Koehn and Knowles, 2017) technique that re-ranks these early terminating hypotheses can successfully address this issue.

Other problems for existing sequence-to-sequence neural summarizers include generation of factually incorrect summaries, and, importantly, vulnerability to adversarial information suggesting a crucial lack of semantic understanding. Finally, abstractive summaries can still be largely extractive or contain redundant information ((See et al., 2017)).

3 Model

Our model learns to generate an abstractive summary from a given source document. Based on and extending the sequence-to-sequence framework, we compute copy and coverage vectors to address redundant and repetitive generation, and positional encodings to achieve good source text representation. Figure 1 shows an example of our model generated two-sentence summary of a news article. The overall structure of our model is illustrated in Figure 2.

3.1 Copy Mechanism

As some tokens that occur in the source document are out-of-vocabulary words, a mechanism is needed to enable their generation. We use the copy mechanism, initially introduced in (Gu et al., 2016), to allow copying words from the source text thus enabling our model to produce OOV words and not be restricted to a pre-set fixed vocabulary.

The final probability distribution from which the model makes predictions is a weighted sum of the probability of generating words from the pre-set vocabulary and the probability of copying words from the source text using the attention distribution.

To calculate the attention distribution a over the source text at decoder time step t we use the bilinear dot product of the last layer decoder output s_t and encoder output h_j as follows:

$$u_t = s_t W_c h_j$$
$$a_t^j = \exp u_t^j / \sum_k \exp u_t^k$$

Then we calculate the copy probability $g_t \in [0, 1]$ which we use to adjust the model selection between copying from the source and generating from the vocabulary.

$$g_t = sigmoid(W_g[s_t, h_j] + b_g)$$

where Wc, Wg, bg are learnable parameters. So, the final probability distribution P from which the model predicts the summary token w to generate or copy at each time step t is calculated as follows:

$$p_t(w) = (1 - g_t)P_t^{vocab}(w) + g_t \sum_i^{w_i=w} a_t^i$$

3.2 Coverage Mechanism

We compute a vector to discourage repetition in our model-generated summaries. We follow ((See et al., 2017)) and maintain a coverage vector c_t as the sum of attention distributions over all previous decoder time steps:

$$c_t = \sum_{t'=0}^{t'=t-1} a^{t'}$$

Then, we use the coverage vector c_t as an extra input to the attention mechanism to help it remember its previous decisions and avoid repeated attention to the same locations in the source text.

3.3 Positional Encodings

Sinusoidal positional encodings were developed for non-recurrent neural networks, initially for the transformer model for machine translation (Vaswani et al., 2017). We are the first to make use of this feature in a recurrent neural model. We compute positional encodings and add them to the initial word representations as seen in Figure 2. The position computation of embedding size 512 uses sine and cosine functions of different frequencies as follows:

$$PosEnc_{(pos,2i)} = sin(pos/10000^{2i/512})$$
$$PosEnc_{(pos,2i+1)} = cos(pos/10000^{2i//512})$$

with each dimension i of the encoding corresponding to a sinusoid.

3.4 Learning Objective

We use a token level learning objective. During model training, the decoder is fed the ground-truth summary and the model parameters θ are optimized maximizing the likelihood of the training data, which is achieved by minimizing the cross entropy loss L:

$$L(\theta) = -\sum_{\tau=1}^{T} logp(yt|X, y < t; \theta)$$

In this method, also known as teacher forcing, ground truth tokens are shown to the model just before the decoder makes its next step prediction. A more time consuming approach would be to use a sequence level objective which incorporates policy gradient learning or a minimum risk training strategy to maximize the ROUGE score of generated summaries as in (Carbonell and Goldstein, 1998). A mixed objective that combines word and sequence level objectives with a fixed hyperparameter value was used in ((Paulus et al., 2017)).

4 Experiments

4.1 Datasets

We perform experiments on the CNN/Daily-Mail news articles summarization dataset ((Hermann et al., 2015)) and the Gigaword sentence summarization/headline generation corpus (Rush et al., 2017), which are both standard corpora for long and short document summarization. For the CNN/Daily-Mail train and validation splits, we truncate source text to 400 tokens and target summaries to 100 tokens, following standard practice. We limit both input and output vocabulary to the 50000 most frequent words, and replace the rest with UNK tokens. For training on the Gigaword dataset we follow the pre-processing steps of (Rush et al., 2017), replacing all digit characters with # and tokens seen less than five times with UNK. Table 1 shows the main statistics for both corpora.

Dataset	Train	Valid	Test	DL	SL
CNN/DM	287226	13368	11490	781	56
Gigaword	3803957	189651	1951	31.4	8.3

Table 1: Dataset statistics. DL and SL denote average number of tokens in source document and summary, respectively.

4.2 Training details

We train our models with the Adam optimizer (Kingma and Ba, 2014) with $\beta 1$ = 0.9 and $\beta 2$ = 0.998. We increase (warm up) the learning rate linearly for the first 8000 steps and then decrease it exponentially, following the noam decay scheme. We randomly initialize and learn during training word embeddings of size 512, and apply positional encoding before feeding them to a four layer LSTM stack with 512 hidden units per layer. To regularize, we use dropout (with probability 0.2) between the stacked LSTM hidden states. At test time, we use a beam size of 3 and, for CNN/Daily Mail, set the minimum length of the generated summary to 35. We do not use the trigram repetition avoidance heuristic defined in ((Paulus et al., 2017)), because we find it results in decreased performance on both datasets. We implemented our models using PyTorch on the OpenNMT system (Klein et al., 2017). We ran the experiments on a 12GB Titan Xp GPU.

4.3 Models

Baselines: We consider two strong baseline models that do not use positional encodings, (1) a four-layer transformer model with 80,68 million parameters and (2) a four-layer recurrent model with 67,80 million parameters, with a bidirectional LSTM encoder and unidirectional LSTM decoder.

Our model: We form our model by simply including fixed, sinusoidal positional encodings to our recurrent baseline, thus keeping the same architecture settings and parameters.

Model	R-1	R-2	R-L	R-AVG
LSTM 4l	37.99	16.73	35.04	29.92
Transformer	37.88	16.48	34.94	29.77
Our method	**38.60**	**17.50**	**35.81**	**30.64**
Celikyilmaz et al., (2018)	41.69	19.47	37.92	33.02

Table 2: Rouge scores on the CNN/DM test set.

Model	R-1	R-2	R-L	R-AVG
LSTM 4l	33	16.31	31.11	26.80
Transformer	33.49	16.85	31.67	27.47
Our method	**33.09**	**16.36**	**31.24**	**26.90**
Cao et al., (2018a)	37.04	19.03	34.46	30.17

Table 3: Rouge scores on the Gigaword test set.

4.4 Evaluation Metrics

For both CNN/DM and Gigaword datasets, we report the full length F-1 scores of the ROUGE-1, ROUGE-2 and ROUGE-L metrics and their average (R-AVG).

5 Results

The main results of our neural text summarizers for the CNN/DM corpus are listed in Table 2. The two baseline models are shown in the top two lines followed by our proposed model. We observe that our position aware LSTM summarizer scores better than our two baselines, without requiring any additional model parameters or fine-tuning.

On the other hand, our small, cross-entropy trained recurrent model did not match the performance of the large recurrent model of (Celikyilmaz et al., 2018) which uses multiple communicating encoders connected to a single decoder, is trained using reinforcement learning and sets the state-of-the art performance in this dataset.

Table 3 shows experiments with the same three models trained and evaluated on the Gigaword corpus and Figure 4 shows the corresponding model summaries. We can see that the positional-encoding improvement compared to the baselines did not carry over to this dataset. Here, our proposed recurrent model, although marginally better than the recurrent baseline, does not outperform the transformer summarizer. We hypothesize this result could be due to the better language modeling ability of the transformer model compared to the LSTM models in this dataset, as shown in Figure 3 from their corresponding perplexity values.

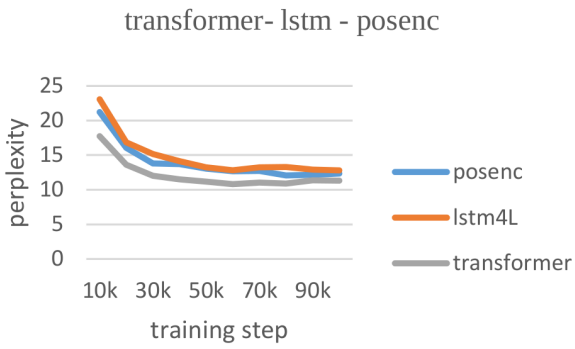

Figure 3: Perplexity scores (lower is better) of different models trained on Gigaword. Transformer scores better than both lstm and posenc (best seen in color).

We note that the state-of-the-art neural model (Cao et al., 2018a) in this dataset, is significantly more complex and memory demanding than our model. When generating its summaries it utilizes an information-retrieval platform to implement a template-based summarization approach and does not simply depend on the source text.

6 Related work

Early approaches to text summarization were based in first finding and then reordering (re-ranking) the most important sentences in a document based on their word frequency or some sentence-similarity metric. Then, a simple extraction of the top k highest scoring sentences from the source document could produce a grammatical correct, albeit incoherent, summary.

The need for more human-like, abstractive summary creation led to the modern sequence-to-sequence models with attention. These neural networks are able to generate any word from their vocabulary, even novel words and phrases unseen in the source document, but can also copy from it when generating an out of vocabulary word is called for.

However, problems like repetitive, generic, or ungrammatical summary generation, with limited abstraction and easily fooled by irrelevant information remained intact for the standard neural network summarizers. Several extensions to their basic encoder-decoder architecture or their end-to-end learning strategy developed accordingly.

In (Lin et al., 2018) the authors use a convolutional gated unit to help control the information flow between the encoder and decoder networks aiming to filter the secondary and preserve only the core information, while Zhou et al. (2017) de-

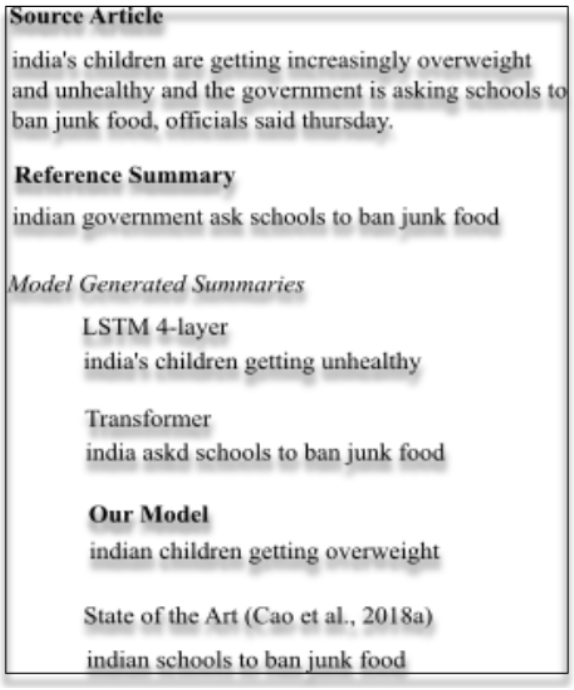

Figure 4: Source article, reference and model generated summaries from the Gigaword test set.

sign a selective gate network with the same goal. In order to avoid generating fake facts in a summary, Cao et al. (2018b) extract actual factual descriptions from the source text leveraging information retrieval techniques. A task-agnostic diverse beam search procedure is proposed in (Vijayakumar et al., 2018) that modifies the standard beam search algorithm in the direction of more diverse text generation.

Other works explore abstractive sentence compression with paraphrasing (Nayeem et al., 2019), different network training regimes (Ayana et al., 2016) or architectures that jointly learn summarization and semantic parsing (Fan et al., 2018). The authors in (Guo et al., 2018) propose a multi-task model with parallel training of three tasks: summary generation, question generation, and entailment generation and find it provides useful guidance for summarization. While we share their motivation to make the model input richer, our work presents a much simpler approach. Another recent attempt to produce rich pre-trained encoder representations for many downstream tasks, including summarization, is BERT (Dev(Lin et al., 2018)).

7 Conclusion

The application of encoder-decoder structures has attracted growing attention in the area of longer text summarization research. Neural networks with recurrences, convolutions and transformers were developed for the task of single-document summarization. We began this work aiming to explore the causal factors with the greatest impact in final model output. In the process, we found that position aware recurrent networks can be a simpler, better performing approach than transformers in abstractive single document summarization.

Recent advances in word contextual representations hold the promise of richer, more abstractive summary generation. In this paper, we show that explicitly representing and using the positional information of source text tokens in a recurrent sequence to sequence summarizer helps improve its performance.

Relative position representations, which encode the distance between sequence elements rather than their absolute position, could also help further improve performance. This effect could take place through enabling better optimization of the information selection process in later processing steps, an hypothesis we aim to explore in future work.

Acknowledgments

This research is funded by the University of Macedonia Research Committee as part of the "Principal Research 2019" funding program. We thank the anonymous reviewers for helpful comments. The Titan Xp used for this work was donated by the NVIDIA Corporation.

References

Shiqi Shen Ayana, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. arXiv preprint arXiv:1604.01904.

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

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018a. Retrieve, rerank and rewrite: Soft template based neural summarization. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 152–161.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018b. Faithful to the original: Fact aware neural abstractive summarization. In Thirty-Second AAAI Conference on Artificial Intelligence.

Jaime G Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In SIGIR, volume 98, pages 335–336.

Asli Celikyilmaz, Antoine Bosselut, Xiaodong He, and Yejin Choi. 2018. Deep communicating agents for abstractive summarization. arXiv preprint arXiv:1803.10357.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. arXiv preprint arXiv:1805.11080.

Kyunghyun Cho, Aaron Courville, and Yoshua Bengio. 2015. Describing multimedia content using attention-based encoder-decoder networks. IEEE Transactions on Multimedia, 17(11):1875–1886.

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.

Eric Chu and Peter Liu. 2019. Meansum: a neural model for unsupervised multi-document abstractive summarization. In International Conference on Machine Learning, pages 1223–1232.

Lisa Fan, Dong Yu, and Lu Wang. 2018. Robust neural abstractive summarization systems and evaluation against adversarial information. arXiv preprint arXiv:1810.06065.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1243–1252. JMLR. org.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. arXiv preprint arXiv:1603.06393.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Soft layer-specific multi-task summarization with entailment and question generation. arXiv preprint arXiv:1805.11004.

Johan Hasselqvist, Niklas Helmertz, and Mikael Kågebäck. 2017. Query-based abstractive summarization using neural networks. arXiv preprint arXiv:1712.06100.

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.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. arXiv preprint arXiv:1701.02810.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. arXiv preprint arXiv:1706.03872.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Text summarization branches out, pages 74–81.

Junyang Lin, Xu Sun, Shuming Ma, and Qi Su. 2018. Global encoding for abstractive summarization. arXiv preprint arXiv:1805.03989.

Mir Tafseer Nayeem, Tanvir Ahmed Fuad, and Yllias Chali. 2019. Neural diverse abstractive sentence compression generation. In European Conference on Information Retrieval, pages 109–116. Springer.

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

Alexander M Rush, SEAS Harvard, Sumit Chopra, and Jason Weston. 2017. A neural attention model for sentence summarization. In ACLWeb. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. arXiv preprint arXiv:1704.04368.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1171–1181.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In Thirty-Second AAAI Conference on Artificial Intelligence.

Li Wang, Junlin Yao, Yunzhe Tao, Li Zhong, Wei Liu, and Qiang Du. 2018. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. arXiv preprint arXiv:1805.03616.

Ziang Xie. 2017. Neural text generation: A practical guide. arXiv preprint arXiv:1711.09534.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. arXiv preprint arXiv:1704.07073.