Enhancing a Text Summarization System with ELMo

Claudio Mastronardo
DISI - University of Bologna, Italy
claudio.mastronardo@studio.unibo.it

Fabio Tamburini
FICLIT - University of Bologna, Italy
fabio.tamburini@unibo.it

Abstract

Text summarization has gained a considerable amount of research interest due to deep learning based techniques. We leverage recent results in transfer learning for Natural Language Processing (NLP) using pre-trained deep contextualized word embeddings in a sequence-to-sequence architecture based on pointer-generator networks. We evaluate our approach on the two largest summarization datasets: *CNN/Daily Mail* and the recent *Newsroom* dataset. We show how using pre-trained contextualized embeddings on Newsroom improves significantly the state-of-the-art ROUGE-1 measure and obtains comparable scores on the other ROUGE values.

1 Introduction

The amount of human generated data is outstanding: every day we generate about 2 quintillion bytes of unstructured data and this number is expected to grow. With such a huge amount of information, swiftly accessing and comprehending large piece of textual data is becoming more and more difficult. Automatic text summarization constitutes a powerful tool which can provide a useful solution to this problem.

In recent years, automatic text summarization systems have gained a considerable amount of research interest due to deep learning powered NLP impressive results (Mikolov et al., 2013; Bahdanau et al., 2015; Yang et al., 2017; Vaswani et al., 2017; Józefowicz et al., 2016; Devlin et al., 2019). Neural network (NN) based approaches have always been considered data hungry techniques because they often require a large amount of training data, but, in the latest years, several works have made a huge contribution in this direction (Grusky et al., 2018; Nallapati et al., 2016a; Napoles et al., 2012).

Text summarization systems can be divided into two main categories: *Extractive* and *Abstractive* (Shi et al., 2018). The first generate summaries by purely copying the most representative chunks from the source text (Dorr et al., 2003; Nallapati et al., 2016b), while in the second summarization algorithms make up summaries by using novel phrases and words in order to rephrase and compress the information in the source text (Chopra et al., 2016). Some works shed light on using both approaches through hybrid neural architectures attempting to gather the best characteristics of each world (See et al., 2017; Khatri et al., 2017).

NLP has seen a tremendous amount of attention after several deep learning based important results (Lample et al., 2016; Józefowicz et al., 2016; Hermann et al., 2015). Most of them relied on the concept of distributed representation of words, defining them as real-valued vectors learned from data (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017; Joulin et al., 2017). Recent results were able to generate richer word embeddings by exploiting their linguistic context in order to model word polysemy (Peters et al., 2018; McCann et al., 2017; Peters et al., 2017).

In this paper, we build upon the work of See et al. (2017) on the Pointer-Generator Network for text summarization by integrating it with recent advances in transfer learning for NLP with deep contextualized word embeddings, namely an ELMo model (Peters et al., 2018). We show that, using pre-trained deep contextualized word embeddings, integrating them with pointer-generator networks and learning the ELMo parameters for combining the various model layers together with the text summarization model, we can improve substantially some of the ROUGE evaluation met-

rics. Our experiments were based on two datasets commonly used to evaluate this task: *CNN/Daily Mail* (Nallapati et al., 2016a) and *Newsroom* (Grusky et al., 2018).

2 Related work

One of the first neural encoder-decoder approaches to text summarization has been presented by Nallapati et al. (2016a) where they show that an off-the-shelf encoder-decoder framework, used for machine translation, already outperforms the previous systems for text summarization. They also augment input data by concatenating to classical word embeddings part-of-speech tags, named-entity tags and tf-idf statistics. They leverage the *hierarchical attention* mechanism where less important chunks of text are less attended with a chunk-level mechanism attention.

Zhou et al. (2017) propose selective encoding for text summarization by introducing a selective gate network into the encoder with the purpose of distilling salient information from source articles. Then a second layer called "distilled representation" is constructed by multiplying the selective gate to the hidden state of the first layer. Such gate network can control information flow from encoder to the decoder and select salient information, boosting the performances of the sentence summarization task.

Read-Again Encoding (Zeng et al., 2016) follows the human approach of reading several times before writing a summary by using two LSTM encoders reading the source article and a transformed version of the first LSTM output respectively. Another original approach is presented by Xia et al. (2017) where they follow another human-driven approach by first writing a draft and then polishing it looking at the global context. In an encoder-decoder framework there are two decoders, the first attends to encoder states and generates a draft while the second attends to both the encoder and first decoder outputs generating a summary by exploiting information from *two* context vectors. This approach, called *deliberation network*, boosted the performances for both text summarization and machine translation.

Another set of approaches uses reinforcement learning as in Chen and Bansal (2018), where they use two sequence-to-sequence models. The first is defined as an extractive model with the goal of extracting salient sentences from the input source. The second is an abstractive model which paraphrases and compresses the extracted sentences into a short summary. They make use of convolutional neural networks (ConvNet) to encode tokens and train the two models by using standard policy gradient methods treating them as reinforcement learning agents.

Paulus et al. (2018) presented a new abstractive summarization model achieving state-of-the-art on the New York Times dataset by introducing intra-temporal attention in both encoder and decoder. They use a new objective function by combining maximum-likelihood cross-entropy loss and rewards from policy gradient reinforcement learning in order to reduce the exposure bias and train their architectures by directly optimizing the ROUGE score.

Another research direction goes beyond RNNs to avoid their computational and memory costs by using ConvNet-based encoder-decoder models. Kalchbrenner et al. (2016) adopt one-dimensional convolutions stacking on top of the hidden representation on the encoder/decoder ConvNet. Quasi-Recurrent Neural Networks (Bradbury et al., 2017) use encoders and decoders made of convolutional layers and dynamic average pooling layers, requiring less amount of computational time when compared with LSTMs. Several other approaches attempted to use ConvNets for NLP.

It is also relevant the *transformer* model proposed in (Vaswani et al., 2017) which uses only feed-forward NN and multi-head attention.

3 Datasets

All the experiments in this work have been conducted on two datasets. The first, the CNN/Daily Mail dataset (Nallapati et al., 2016a), has been created by scraping news articles from the `cnn.com` website and concatenating news highlights in order to form a multi-sentence summary. It is composed of about 300,000 examples. The second, the recently released Newsroom dataset (Grusky et al., 2018) consists of 1.3 million article-summaries pairs. It is the largest and most diverse dataset known in literature. Compared to CNN/Daily Mail dataset, Newsroom has been created with the explicit goal of summarizing articles over two decades by using 38 major publishers as sources. Authors in (Grusky et al., 2018) also demonstrate that CNN/Daily Mail dataset is skewed towards extractive summaries, while the News-

room dataset covers a wider range of summarization styles, highly abstractive/extractive summaries and several article-summary compression ratios. For these reasons, even if we will provide the results for both datasets, we will mainly comment them only for the Newsroom dataset.

4 The Proposed Model

Our approach builds upon the work made by See et al. (2017) on pointer-generator networks applied to text summarization. The pointer-generator network is based on the architecture presented in (Nallapati et al., 2016c).

4.1 Pointer-Generator Network

It is an encoder-decoder architecture where tokens of a source text are fed one-by-one to an encoder network (a single layer LSTM) which also generates a sequence of hidden states. The decoder network (a single layer LSTM), at each step t receives the embedding of the emitted word at time $t-1$ and the current decoder's hidden state. This architecture makes use of Bahdanau attention (Bahdanau et al., 2015) using:

$$e_i^t = \mathbf{v}^T \tanh\left(\mathbf{W}_h h_i + \mathbf{W}_s s_t + \mathbf{b}_{\text{attn}}\right)$$
$$a^t = \text{softmax}\left(e^t\right)$$

where s_t represents the decoder's hidden state at step t, h_i represents the encoder's hidden state at timestep i and e_i^t represents the weight given to h_i at decoder's timestep t not yet normalized. Capital letters mark trainable parameters. The tensor *a* represents a probability distribution over encoder's hidden states and encodes how much to attend each state in order to alleviate the encoder from the responsibility of encoding all the information into a fixed vector. The tensor *a* is used to produce a weighted sum of the encoder hidden states called h^* which is concatenated to the decoder's current hidden state making up the input tensor for the LSTM cell that produces a distribution of probability over the vocabulary.

Pointer-generator networks extend this architecture by leveraging ideas from pointer networks (Vinyals et al., 2015): it is a special kind of architecture being able to point to a specific input token and copy it from the source text to the output sequence. At each time-step t the network produces a *generation probability* value $p_{gen} \in [0,1]$ calculated from the context vector h^*, the decoder's state s_t and the decoder's input x_t:

$$p_{\text{gen}} = \sigma\left(\mathbf{w}_{h^*}^T h_t^* + \mathbf{w}_s^T s_t + \mathbf{w}_x^T x_t + \mathbf{b}_{\text{ptr}}\right)$$

again capital letters represent learnable parameters and σ indicates the sigmoid function. p_{gen} is used as a soft switch to choose whether to generate a word from the network's vocabulary or copy a word from the source text. So, given p_{gen}, the probability of outputting a word *w* is:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t$$

where P_{vocab} represents the probability value for the word *w* at the output layer of the LSTM decoder, $\sum_{i:w_i=w} a_i^t$ is the sum of the attention values given to the hidden states at time t whose input word was the specific word w. In the case of an extremely low p_{gen}, the decoder gives a higher probability value to the input words which produced hidden states who had been attended the most.

At a given time-step t the loss value is computed as the negative log-likelihood of the ground truth word w_t^* for that time-step

$$\text{loss}_t = -\log P\left(w_t^*\right)$$

and for a given sequence the loss value is computed by averaging the losses for each word.

In order to cope with the common repetition problem (Mi et al., 2016; Tu et al., 2016; Sankaran et al., 2016), the *coverage loss* (Tu et al., 2016) is used to penalize source-document words attended too much. It is implemented by maintaining a *coverage vector* c^t: $c^t = \sum_{t'=0}^{t-1} a^{t'}$ which tracks the degree of coverage that words have received from the attention mechanism so far. This leads to the augmented version of the attention mechanism including the coverage loss

$$e_i^t = \mathbf{v}^T \tanh\left(\mathbf{W}_h h_i + \mathbf{W}_s s_t + \mathbf{W}_c c_i^t + \mathbf{b}_{\text{attn}}\right)$$

with $\mathbf{W}_c$ as learnable parameter. Hence, coverage loss is computed by:

$$\text{covloss}_t = \sum_i \min\left(a_i^t, c_i^t\right)$$

in order to prevent repeated attention.

4.2 Deep Contextualized Word Embeddings

The original pointer-generator network does not use pre-trained word embeddings, but it learns 128-dimensional word embeddings from scratch during training. Even though learning specialized word embeddings for the summarization task might seem a reasonable approach, we think that using pre-trained word embeddings could improve the overall network performance.

Following Peters et al. (2018) we adopt a transfer learning approach by leveraging the power of

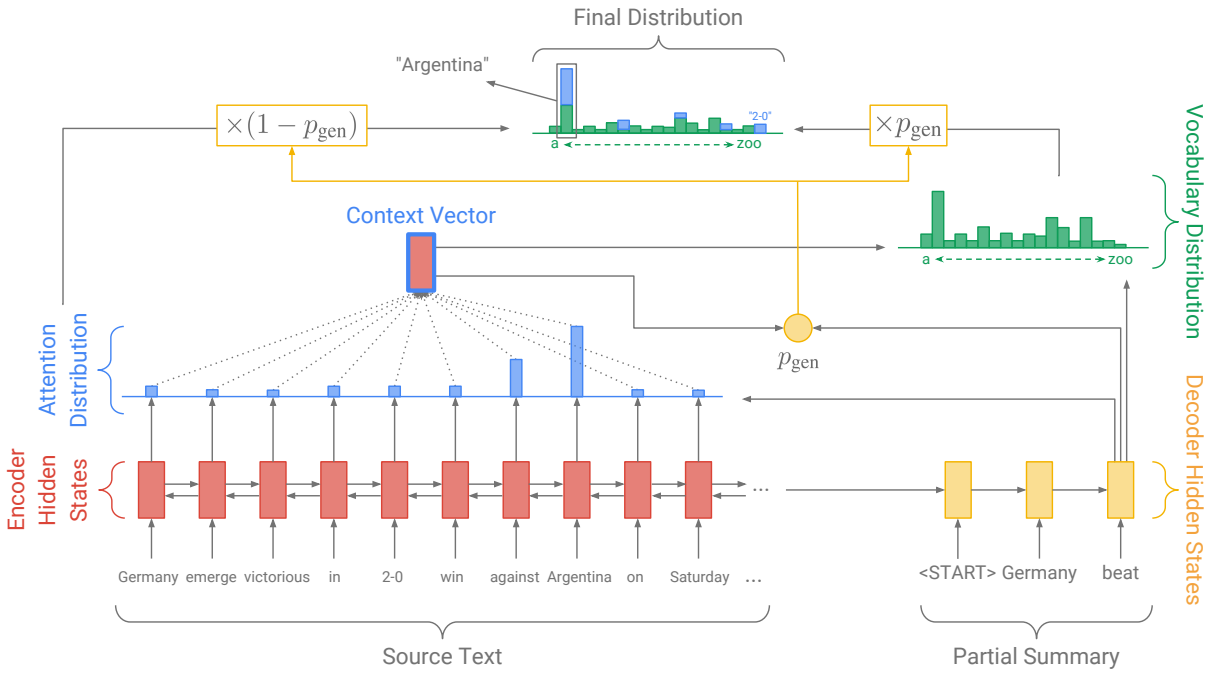

Figure 1: The pointer-generator model. At each time-step the encoder reads a word and outputs an hidden state. The decoder attends to encoder hidden states and generates the attention distribution. After generating p_{gen}, it weights and adds the attention distribution and the vocabulary distribution leading to the final word distribution. Picture courtesy of See et al. (2017).

pre-trained deep contextualized word embeddings. Embedding from Language Model (ELMo) is a particular type of embedding where word representation is a function of the entire input sequence. ELMo trains a bidirectional language modeling architecture inspired by Józefowicz et al. (2016) and Kim et al. (2016), on a large corpus. In order to compute the probability for the token t_k, the language model architecture computes a context-independent token representation via a ConvNet over characters and passes the output to a L-layer bidirectional LSTM. An ELMo representation is the result of a weighted combination of the hidden states of the language modeling architecture. For each token t_k, this architecture computes a set of $2L + 1$ representations: $R_k = \left\{\mathbf{h}_{k,j}^{LM} | j = 0, \ldots, L\right\}$ where $\mathbf{h}_{k,0}^{LM}$ is the output of the ConvNet token layer and $\mathbf{h}_{k,j}^{LM} = \left[\overrightarrow{\mathbf{h}}_{k,j}^{LM}; \overleftarrow{\mathbf{h}}_{k,j}^{LM}\right]$ $j > 0$, for each bi-LSTM layer.

More generally, in order to use ELMo for a specific downstream task, word representations are computed by a weighted sum of each intermediate network representation:

$$\text{ELMo}_k^{\text{task}} = \gamma^{\text{task}} \sum_{j=0}^{L} s_j^{\text{task}} \mathbf{h}_{k,j}^{LM}$$

where s^{task} are softmax-normalized learnable weights and γ^{task} allows to scale the entire produced vector with respect to the downstream task.

Our method feeds ELMo embeddings into a pointer-generator model: as the encoder reads the source text, a pre-trained ELMo model generates contextualized word embeddings. Pointer-generator encoder has two main sources to keep track of what has been read: its own memory and the inner information about past and following words injected into the current word embedding. We learn the s^{task} and γ^{task} weights during training.

We used the "Original (5.5B)" ELMo embeddings[1]. The encoder gets 1024 dimensional embeddings which are fed into an LSTM cell of 512 neurons followed by a linear layer. Between the encoder and the decoder there is a neural network called *reduce state* with the aim of reducing the dimensionality of the passed tensors. The decoder is a bidirectional LSTM with size 256 followed by two linear layers of 256 neurons. We use an attention network with Bahdanau's formula and the coverage mechanism. Decoder's vocabulary size is set to the first most common 50,000 tokens in the training set. Freezing the model from learning embeddings from scratch reduces the number

[1] https://allennlp.org/elmo

Paper	R-1	R-2	R-L
(See et al., 2017)	39.53	17.28	36.38
(Paulus et al., 2018)	41.16	15.75	39.08
(Gehrmann et al., 2018)	41.22	18.68	38.64
(Liu, 2019)	**43.25**	**20.24**	**39.63**
This work	38.96	16.25	34.32

Table 1: ROUGE metrics on CNN/Daily Mail test set.

Paper	R-1	R-2	R-L
(Grusky et al., 2018) (Pointer-generator)	26.04	13.24	22.45
(Shi et al., 2018)	39.36	**27.86**	**36.35**
This work	**40.49**	27.15	34.11

Table 2: ROUGE metrics on the Newsroom test set.

of parameters of 2,150,011. We trained our architecture on both CNN/Daily Mail and Newsroom datasets using Adagrad as the optimization algorithm (Duchi et al., 2011) with an initial learning rate of 0.15 and the initial accumulator set to 0.1. During training the batch size has been fixed to 8 and we run the decoder for at least 35 steps. As pre-processing step we just lowercased and tokenized texts using the *nltk* python package. The loss function remained unchanged since we used the negative log-likelihood for the ground truth word with coverage loss.

5 Experimental Results

We trained our model for 455,000 iterations on CNN/Daily Mail and for 520,000 iterations on Newsroom. The best performing models have been tested on both CNN/Daily Mail and Newsroom test sets and the ROUGE metrics are reported in Table 1 and 2 respectively.

The proposed approach achieves state-of-the-art ROUGE-1 value for the Newsroom dataset and competitive values for ROUGE-2 and ROUGE-L. ELMo addition causes an increase of $+14.45$, $+13.91$ and $+11.66$ for the three metrics with respect to basic pointer-generator from Grusky et al. (2018). ELMo s^{task} learned weights are, respectively, 0.4140, 0.4690, 0.1169 and $\gamma^{task} = 0.35$. This shows that the model favours syntactic information (captured at lower LSTM layers) instead of semantic information when generating text embeddings. From a qualitatively point of view we report some network generated summaries as supplementary material[2]. As we can see the model can generate fairly reasonable summaries, which can differ from the ground truth but still represent valid alternatives. This can explain the high value for ROUGE-1, meaning that summaries' words have been covered anyway but in a different order (causing a lower ROUGE-L).

6 Discussion and Conclusions

In this work we leveraged recent results in transfer learning for NLP with deep contextualized word embeddings in conjunction with pointer-generator NN for automatic abstractive text summarization. We noticed a considerable increase of model's performance in terms of the ROUGE score, achieving state-of-the-art on the Newsroom dataset for the ROUGE-1 metric. This is a dataset designed for testing abstractive systems while the other dataset (CNN/Daily Mail) contains summaries formed by sentences extracted from the original texts and it is more suitable for testing extractive systems. Then, it is reasonable that we got improvements only when using the Newsroom dataset.

Intrinsic, corpus-based metrics based on string overlap, string distance, or content overlap, such as BLEU and ROUGE, suffer from the need to have a reference output provided by the gold standard corpus in order to evaluate the system outputs. That seems very problematic (e.g. see Gatt and Krahmer 2018) because the reference summary is only one of the possible summaries that humans can produce. By looking at the supplementary material regarding some examples of our system output, one can immediately recognize that, even if very different from the reference one, the summaries produced by the proposed system are in most cases acceptable.

The definition of proper metrics capturing in the right way the correctness of system outputs remains, in our opinion, a critical open issue. As discussed also in the recent review by Chatzikoumi (2019) about Machine Translation (MT) metrics, "When reference translations are used [...] MT outputs that are very similar to the reference translation are boosted and not similar MT outputs are penalised even if they are good", the so-called "reference bias". The same metrics are currently used also in text summarization leading to similar problems.

[2] https://bit.ly/2XUJvbd

Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

References

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

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural networks. In *Proc. of ICLR 2017*, Toulon, France.

Eirini Chatzikoumi. 2019. How to evaluate machine translation: A review of automated and human metrics. *Natural Language Engineering*, pages 1–25.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. of ACL 2018*, pages 675–686, Melbourne, Australia.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL 2019*, pages 4171–4186, Minneapolis, Minnesota.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 Text Summarization Workshop*, pages 1–8, Edmonton, Canada.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Int. Res.*, 61(1):65–170.

Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. 2018. Bottom-up abstractive summarization. In *Proc. of EMNLP 2018*, pages 4098–4109, Brussels, Belgium.

Max Grusky, Mor Naaman, and Yoav Artzi. 2018. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In *Proc. of NAACL2018*, pages 708–719, New Orleans, Louisiana.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proc. of NIPS 2015*, pages 1693–1701, Montreal, Canada.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of tricks for efficient text classification. In *Proc. of EACL 2017*, pages 427–431, Valencia, Spain.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *CoRR*, abs/1610.10099.

Chandra Khatri, Gyanit Singh, and Nish Parikh. 2017. Abstractive and extractive text summarization using document context vector and recurrent neural networks. In *Proc. of CoNLL 2016*, pages 280–290, Berlin, Germany.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proc. of AAAI 2016*, pages 2741–2749, Phoenix, Arizona.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL-HLT 2016*, pages 260–270, San Diego, California.

Yang Liu. 2019. Fine-tune BERT for extractive summarization. *CoRR*, abs/1903.10318.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *Proc.*

of NIPS 2017, pages 6297–6308, Long Beach, CA.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Proc. of EMNLP 2016*, pages 955–960, Austin, Texas.

Tomas Mikolov, G.s Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proc. of ICLR 2013*, Scottsdale, Arizona.

Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016a. Sequence-to-sequence rnns for text summarization. In *Proc. of Workshop track - ICLR 2016*, San Juan, Puerto Rico.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2016b. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. In *Proc. of AAAI 2016*, Phoenix, Arizona.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. 2016c. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100, Montreal, Canada.

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

Jeffrey Pennington, Richard Socher, and Christoper Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP 2014*, pages 1532–1543, Doha, Qatar.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *Proc. of ACL 2017*, pages 1756–1765, Vancouver, Canada.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL-HLT 2018*, pages 2227–2237, New Orleans, Louisiana.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *CoRR*, abs/1608.02927.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proc. of ACL 2017*, pages 1073–1083, Vancouver, Canada.

Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *CoRR*, abs/1812.02303.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proc. ACL 2016*, pages 76–85, Berlin, Germany.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. of NIPS 2017*, Long Beach, CA.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Proc. of NIPS 2015*, pages 2692–2700, Montreal, Canada.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. In *Proc. NIPS 2017*, pages 1784–1794, Long Beach, CA.

Zichao Yang, Zhiting Hu, Yuntian Deng, Chris Dyer, and Alex Smola. 2017. Neural machine translation with recurrent attention modeling. In *Proc. of EACL 2017*, pages 383–387, Valencia, Spain.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *CoRR*, abs/1611.03382.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2017. Selective encoding for abstractive sentence summarization. In *Proc. of ACL 2017*, pages 1095–1104, Vancouver, Canada.