Jointly Extracting and Compressing Documents with Summary State Representations

Afonso Mendes[♠] **Shashi Narayan**[◇*] **Sebastião Miranda**[♠]
Zita Marinho[♡♠] **André F. T. Martins**[†♣] **Shay B. Cohen**[◇]

[♠]Priberam Labs, Alameda D. Afonso Henriques, 41, 2º, 1000-123 Lisboa, Portugal
[◇]School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK
[♡] Instituto de Sistemas e Robótica, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[†]Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
[♣]Unbabel Lda, Rua Visconde de Santarém, 67-B, 1000-286 Lisboa, Portugal

amm@priberam.com, shashi.narayan@gmail.com, ssm@priberam.com,
zam@priberam.com, andre.martins@unbabel.com, scohen@inf.ed.ac.uk

Abstract

We present a new neural model for text summarization that first extracts sentences from a document and then compresses them. The proposed model offers a balance that sidesteps the difficulties in abstractive methods while generating more concise summaries than extractive methods. In addition, our model dynamically determines the length of the output summary based on the gold summaries it observes during training, and does not require length constraints typical to extractive summarization. The model achieves state-of-the-art results on the CNN/DailyMail and Newsroom datasets, improving over current extractive and abstractive methods. Human evaluations demonstrate that our model generates concise and informative summaries. We also make available a new dataset of oracle compressive summaries derived automatically from the CNN/DailyMail reference summaries.[1]

1 Introduction

Text summarization is an important NLP problem with a wide range of applications in data-driven industries (e.g., news, health, and defense). Single document summarization—the task of generating a short summary of a document preserving its informative content (Spärck Jones, 2007)—has been a highly studied research topic in recent years (Nallapati et al., 2016b; See et al., 2017; Fan et al., 2018; Pasunuru and Bansal, 2018).

Modern approaches to single document summarization using neural network architectures have primarily focused on two strategies: *extractive* and *abstractive*. The former select a subset of the sentences to assemble a summary (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018a,c). The latter generates sentences that do not appear in the original document (See et al., 2017; Narayan et al., 2018b; Paulus et al., 2018). Both methods suffer from significant drawbacks: extractive systems are wasteful since they cannot trim the original sentences to fit into the summary, and they lack a mechanism to ensure overall coherence. In contrast, abstractive systems require natural language generation and semantic representation, problems that are inherently harder to solve than just extracting sentences from the original document.

In this paper, we present a novel architecture that attempts to mitigate the problems above via a middle ground, **compressive summarization** (Martins and Smith, 2009). Our model selects a set of sentences from the input document, and

(ExConSumm Extractive) • (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

(ExConSumm Compressive) • ~~(CNN)~~ A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen ~~, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war~~.
• Ibrahim al-Rubaish died ~~Monday night~~ in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

Figure 1: Summaries produced by our model. For illustration, the compressive summary shows the removed spans strike-through.

[1]Our dataset and code is available at https://github.com/Priberam/exconsumm.

[*] Now at Google London.

Proceedings of NAACL-HLT 2019, pages 3955–3966
Minneapolis, Minnesota, June 2 - June 7, 2019.

compresses them by removing unnecessary words, while keeping the summaries informative, concise and grammatical. We achieve this by dynamically modeling the generated summary using a Long Short Term Memory (LSTM; Hochreiter and Schmidhuber, 1997) to produce **summary state representations**. This state provides crucial information to iteratively increment summaries based on previously extracted information. It also facilitates the generation of variable length summaries as opposed to fixed lengths, in previous extractive systems (Cheng and Lapata, 2016; Nallapati et al., 2017; Narayan et al., 2018c; Zhang et al., 2018). Our model can be trained in both extractive (labeling sentences for extraction) or compressive (labeling words for extraction) settings. Figure 1 shows a summary example generated by our model.

Our contributions in this paper are three-fold:

- we present the first end-to-end neural architecture for EXtractive and COmpressive Neural SUMMarization (dubbed EXCONSUMM, see §3),
- we validate this architecture on the CNN/DailyMail and the Newsroom datasets (Hermann et al., 2015; Grusky et al., 2018), showing that our model generates variable-length summaries which correlate well with gold summaries in length and are concise and informative (see §5), and
- we provide a new CNN/DailyMail dataset annotated with automatic compressions for each sentence, and a set of compressed oracle summaries (see §4).

Experimental results show that when evaluated automatically, both the extractive and compressive variants of our model provide state-of-the-art results. Human evaluation further shows that our model is better than previous state-of-the-art systems at generating informative and concise summaries.

2 Related Work

Recent work on neural summarization has mainly focused on sequence-to-sequence (seq2seq) architectures (Sutskever et al., 2014), a formulation particularly suited and initially employed for abstractive summarization (Rush et al., 2015). However, state-of-the-art results have been achieved by RNN-based methods which are extractive. They select sentences based on an LSTM classifier that predicts a binary label for each sentence (Cheng and Lapata, 2016), based on ranking using reinforcement learning (Narayan et al., 2018c), or even by training an extractive latent model (Zhang et al., 2018). Other methods rely on an abstractive approach with strongly conditioned generation on the source document (See et al., 2017). In fact, the best results for abstractive summarization have been achieved with models that are more extractive in nature than abstractive, since most of the words in the summary are copied from the document (Gehrmann et al., 2018).

Due to the lack of training corpora, there is almost no work on neural architectures for compressive summarization. Most compressive summarization work has been applied to smaller datasets (Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Almeida and Martins, 2013). Other non-neural summarization systems apply this idea to select and compress the summary. Dorr et al. (2003) introduced a method to first extract the first sentence of a news article and then use linguistically-motivated heuristics to iteratively trim parts of it. Durrett et al. (2016) also learns a system that selects textual units to include in the summary and compresses them by deleting word spans guided by anaphoric constraints to improve coherence. Recently, Zhang et al. (2018) trained an abstractive sentence compression model using attention-based sequence-to-sequence architecture (Rush et al., 2015) to map a sentence in the document selected by the extractive model to a sentence in the summary. However, as the sentences in the document and in the summary are not aligned for compression, their compression model is significantly inferior to the extractive model.

In this paper, we propose a novel seq2seq architecture for compressive summarization and demonstrate that it avoids the over-extraction of existing extractive approaches (Cheng and Lapata, 2016; Dlikman and Last, 2016; Nallapati et al., 2016a).

Our model builds on recent approaches to neural extractive summarization as a sequence labeling problem, where sentences in the document are labeled to specify whether or not they should be included in the summary (Cheng and Lapata, 2016; Narayan et al., 2018a). These models often condition their labeling decisions on the document representation only. Nallapati et al. (2017) tries to model the summary as the average representation

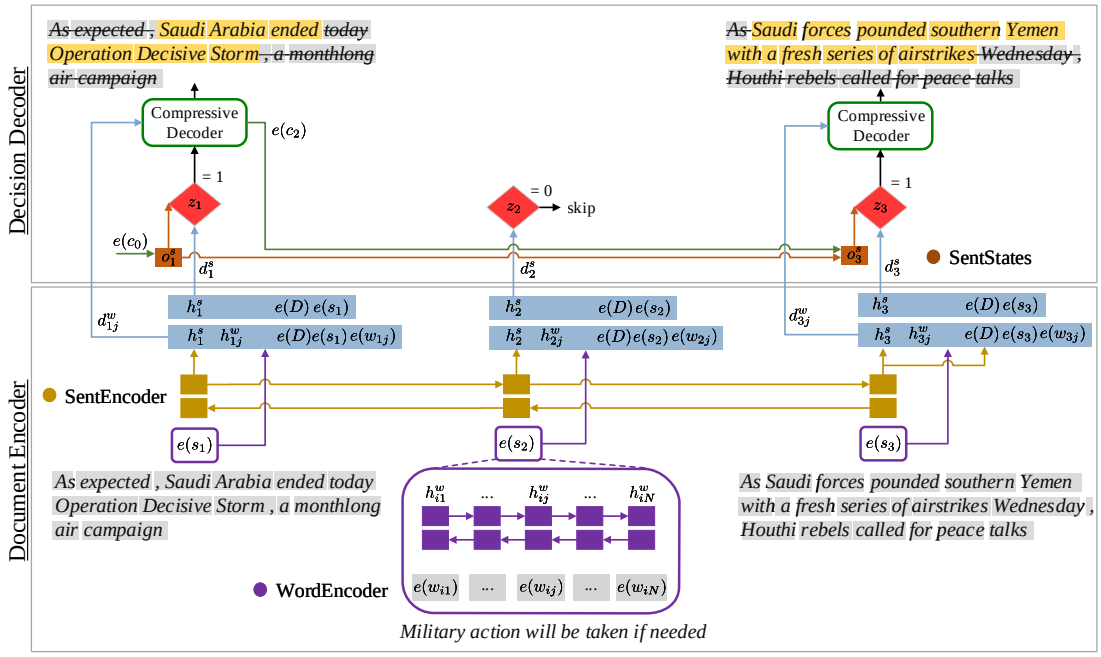

Figure 2: Illustration of our summarization system. The model extracts the most relevant sentences from the document by taking into account the WordEncoder representation of the current sentence $\boldsymbol{e}(s_i)$, the SentEncoder representation of the previous sentence $\boldsymbol{h}_i^s$, the current summary state representation $\boldsymbol{o}_i^s$, and the representation of the document $\boldsymbol{e}(D)$. If a sentence is selected ($z_i = 1$), its representation is fed to SentStates, and we move to the next sentence. Here, sentences s_1 and s_3 were selected. If the model is also compressing, the compressive layer selects words for the final summary (Compressive Decoder). See Figure 3 for details on the decoders.

of the positively labeled sentences. However, as we show later, this strategy is not the most adequate to ensure summary coherence, as it does not take the order of the selected sentences into account. Our approach addresses this problem by maintaining an LSTM cell to dynamically model the generated summary. To the best of our knowledge, our work is the first to use a model that keeps a state of already generated summary to effectively model variable-length summaries in an extractive setting, and the first to learn a compressive summarizer with an end-to end approach.

3 Summarization with Summary State Representation

Our model extracts sentences from a given document and further compresses these sentences by deleting words. More formally, we denote a document $D = (s_1, \ldots, s_M)$ as a sequence of M sentences, and a sentence $s_i = (w_{i1}, \ldots, w_{iN})$ as a sequence of N words. We denote by $\boldsymbol{e}(w_{ij})$, $\boldsymbol{e}(s_i)$ and $\boldsymbol{e}(D)$ the embedding of words, sentences and document in a continuous space. We model document summarization as a sequence labeling problem where the labeler transitions between internal states. Each state is dynamically computed based on the context, and it combines an extractive summarizer followed by a compressive one. First, we encode a document in a multi-level approach, to extract the embeddings of words and sentences ("Document Encoder"). Second, we decode these embeddings using a hierarchical "Decision Decoder." The extractive summarizer labels each sentence s_i with a label $z_i \in \{0, 1\}$ where 1 indicates that the sentence should be included in the final summary and 0 otherwise. An extractive summary is then assembled by selecting all sentences with the label 1. Analogously, the compressive summarizer labels each word w_{ij} with a label $y_{ij} \in \{0, 1\}$, denoting whether the word j in sentence i is included in the summary or not. The final summary is then assembled as the sequence of words w_{ij} for each $z_i = 1$ and $y_{ij} = 1$. See Figures 2 and 3 for an overview of our model. We next describe each of its components in more detail.

3.1 Document Encoder

The document encoder is a two layer biLSTM, one layer encoding each sentence, and the second layer encoding the document. The first layer takes as input the word embeddings $\boldsymbol{e}(w_{ij})$ for each word j in sentence s_i, and outputs the hidden representa-

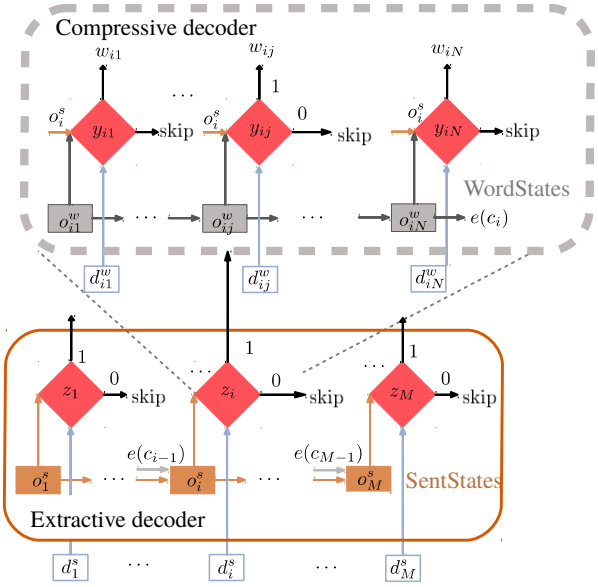

Figure 3: Decision decoder architecture. Decoder contains an extractive level for sentences (orange box) and a compressive level for words (dashed gray box), using an LSTM to model the summary state. Red diamond shapes represent decision variables $z_i = 1$ if $p(z_i \mid \boldsymbol{p}_i) > 0.5$ for selecting the sentence s_i, and $z_i = 0$ if $p(z_i \mid \boldsymbol{p}_i) \leq 0.5$ for skipping this sentence. The same for y_{ij} and $p(y_{ij} \mid \boldsymbol{q}_{ij}) > 0.5$ for deciding over words w_{ij} to keep in the summary.

tion of each word $\boldsymbol{h}^w_{ij}$. The hidden representation consist of the concatenation of a forward $\overrightarrow{\boldsymbol{h}}^w_{ij}$ and a backward $\overleftarrow{\boldsymbol{h}}^w_{ij}$ LSTM (WordEncoder in Figure 2). This layer eventually outputs a representation for each sentence $\boldsymbol{e}(s_i) = [\overrightarrow{\boldsymbol{h}}^w_{iN}, \overleftarrow{\boldsymbol{h}}^w_{i1}]$ that corresponds to the concatenation of the last forward and first backward LSTMs. The second layer encodes information about the document and is also a biLSTM that runs at the sentence-level. This biLSTM takes as input the sentence representation from the previous layer $\boldsymbol{e}(s_i)$ and outputs the hidden representation for each sentence s_i in the document as $\boldsymbol{h}^s_i$ (SentEncoder in Figure 2). We consider the output of the last forward LSTM over M sentences and first backward LSTM to be the final representation of the document $\boldsymbol{e}(D) = [\overrightarrow{\boldsymbol{h}}^s_M, \overleftarrow{\boldsymbol{h}}^s_1]$.

The encoder returns two output vectors, $\boldsymbol{d}^s_i = [\boldsymbol{e}(D), \boldsymbol{e}(s_i), \boldsymbol{h}^s_i]$ associated with each sentence s_i, and $\boldsymbol{d}^w_{ij} = [\boldsymbol{e}(D), \boldsymbol{e}(s_i), \boldsymbol{e}(w_{ij}), \boldsymbol{h}^s_i, \boldsymbol{h}^w_{ij}]$ for each word j at the specific state of the encoder i.

3.2 Decision Decoder

Given that our model operates both at the sentence-level and at the word-level, the decision decoder maintains two state LSTMs denoted by SentStates and WordStates as in Figure 3. For the sentence-level decoder sentences are selected and the state of the summary gets updated by SentStates. For the word-level, all compressed word representations in a sentence are pushed to the word-level layer. In the compressive decoder, words that get selected are pushed onto the WordStates, and once the decoder has reached the end of the sentence, it pushes the output representation of the last state onto the sentence-level layer for the next sentence.

Extractive Decoder The extractive decoder selects the sentences that should go to the summary. For each sentence s_i at time step i, the decoder takes a decision based on the encoder representation $\boldsymbol{d}^s_i$ and the state of the summary $\boldsymbol{o}^s_i$, computed as follows:

$$\boldsymbol{o}^s_i = \text{SentStates}(\{\boldsymbol{e}(c_k)\}_{k<i, z_k=1}).$$

where the $\boldsymbol{o}^s_i$ is modeled by an LSTM taking as input the already selected and compressed sentences comprising the summary so far $\{\boldsymbol{e}(c_k)\}_{k<i,z_k=1}$. This way, at each point in time, we have a representation of the summary given by the SentStates LSTM that encodes the state of summary generated so far, based on the past sentences already processed by the compressive decoder $\boldsymbol{e}(c_{i-1})$ (in WordStates).[2] The summary representation at step i ($\boldsymbol{o}^s_i$) is then used to determine whether to keep or not the current sentence in the summary ($z_i = 1$ or 0 respectively). The summarizer state subsumes information about the document, sentence and summary as:

$$\boldsymbol{p}_i = \tanh(W_E[\boldsymbol{d}^s_i; \boldsymbol{o}^s_i] + \boldsymbol{b}^s),$$

where W_E is a model parameter, $\boldsymbol{o}^s_i$ is the dynamic LSTM state, and $\boldsymbol{b}^s$ is a bias term.

This modeling decision is crucial in order to generate variable length summaries. It captures information about the sentences or words already present in the summary, helping in better understanding the "true" length of the summary given the document.

Finally, the summarizer state $\boldsymbol{p}_i$ is used to compute the probability of the action at time i as:

$$p(z_i \mid \boldsymbol{p}_i) = \frac{\exp\left(W_{z_i}\boldsymbol{p}_i + \boldsymbol{x}_{z_i}\right)}{\sum_{z' \in \{0,1\}} \exp\left(W_{z'}\boldsymbol{p}_i + \boldsymbol{x}_{z'}\right)},$$

[2] When using only the extractive model the summary state $\boldsymbol{o}^s_i$ is generated from an LSTM whose inputs correspond to the sentence encoded embeddings $\{\boldsymbol{e}(s_k)\}_{k<i,z_k=1}$ instead of the previously generated compressed representations $\{\boldsymbol{e}(c_k)\}_{k<i,z_k=1}$.

where W_z is a model parameter and $\boldsymbol{x}_z$ is a bias term for the summarizer action z.

We minimize the negative log-likelihood of the observed labels at training time (Dimitroff et al., 2013), where λ_0^s and λ_1^s represent the distribution of each class for the given sentences:[3]

$$L(\theta^s) = -\sum_{c\in\{0,1\}} \frac{\lambda_c^s}{\sum_{i=1}^{M} \mathbb{1}_{z_i=c}} \sum_{i,z_i=0} \log p(z_i|\boldsymbol{p}_i),$$

where $\mathbb{1}_{z_i=c}$ is the indicator function of class c and θ^s represents all the training parameters of the sentence encode/decoder. At test time, the model emits probability $p(z_i \mid \boldsymbol{p}_i)$, which is used as the soft prediction sequentially extracting the sentence i. We admit sentences when $p(z_i = 1 \mid \boldsymbol{p}_i) > 0.5$.

Compressive Decoder Our compressive decoder shares its architecture with the extractive decoder. The compressive layer is triggered every time a sentence is selected in the summary and is responsible for selecting the words within each selected sentence. In practice, WordStates LSTM (see Figure 3) is applied hierarchically after the sentence-level decoder, using as input the collected word embeddings so far:

$$\boldsymbol{o}_{ij}^w = \text{WordStates}(\{\boldsymbol{e}(w_{ik})\}_{k\le j, y_{ik}=1}).$$

After making the selection decision for all words pertaining to a sentence, the final state of the WordStates, $\boldsymbol{e}(c_i) = \boldsymbol{o}_{iN}^w$ is fed back to SentStates of the extractive level decoder for the consecutive sentence, as depicted in Figure 3.

The word-level summarizer state representation depends on the encoding of words, document and sentence $\boldsymbol{d}_{ij}^w$, on the dynamic LSTM encoding for the summary based on the selected words (WordStates) $\boldsymbol{o}_{ij}^w$ and sentences (SentStates) $\boldsymbol{o}_i^s$:

$$\boldsymbol{q}_{ij} = \tanh(W_C[\boldsymbol{d}_{ij}^w; \boldsymbol{o}_i^s; \boldsymbol{o}_{ij}^w] + \boldsymbol{b}^w),$$

where W_C is a model parameter and $\boldsymbol{b}^w$ is a bias term. Each action at time step j is computed by

$$p(y_{ij} \mid \boldsymbol{q}_{ij}) = \frac{\exp\left(W_{y_{ij}}\boldsymbol{q}_{ij} + \boldsymbol{x}_{y_{ij}}\right)}{\sum_{y'\in\{0,1\}} \exp\left(W_{y'}\boldsymbol{q}_{ij} + \boldsymbol{x}_{y'}\right)},$$

with parameter $W_{y_{ij}}$ and bias $\boldsymbol{x}_{y_{ij}}$. The final loss for the compressive layer is

$$L(\theta^w) = \sum_{i=1}^{M} z_i \phi(i \mid \theta^w),$$

where θ^w represents the set of all the training parameters of the word-level encoder/decoder, $\phi(i)$ is the compressive layer loss over N words:

$$\phi(i \mid \theta^w) = -\sum_{c\in\{0,1\}} \frac{\lambda_c^w}{\sum_{i=1}^{M} \mathbb{1}_{y_{ij}=c}} \sum_{i,z_i=0} \log p(y_{ij}|\boldsymbol{q}_{ij}).$$

The total final loss is then given by the sum of the extractive and compressive counterparts, $L(\theta) = L(\theta^s) + L(\theta^w)$.

4 Experimental Setup

We mainly used the CNN/DailyMail corpus (Hermann et al., 2015) to evaluate our models. We used the standard splits of Hermann et al. (2015) for training, validation, and testing (90,266/1,220/1,093 documents for CNN and 196,961/12,148/10,397 for DailyMail). To evaluate the flexibility of our model, we also evaluated our models on the Newsroom dataset (Grusky et al., 2018), which includes articles form a diverse collection of sources (38 publishers) with different summary style subsets: extractive (Ext.), mixed (Mixed) and abstractive (Abs.). We used the standard splits of Grusky et al. (2018) for training, validation, and testing (331,778/36,332/36,122 documents for Ext., 328,634/35,879/36,006 for Mixed and 332,554/36,380/36,522 for Abs.). We did not anonymize entities or lower case tokens.

4.1 Estimating Oracles

Datasets for training extractive summarization systems do not naturally contain sentence/word-level labels. Instead, they are typically accompanied by abstractive summaries from which extraction labels are extrapolated. We create extractive and compressive summaries prior to training using two types of *oracles*.

We used an *extractive oracle* to identify the set of sentences which collectively gives the highest ROUGE (Lin and Hovy, 2003) with respect to the gold summary (Narayan et al., 2018c).

To build a *compressive oracle*, we trained a supervised sentence labeling classifier, adapted from

[3] If $M - \sum_{i=1}^{M} z_i{=}0$ or $\sum_{i=1}^{M} z_i{=}0$, we simply consider the whole term to be zero. Here M represents the number of sentences in the document.

Oracle	R1	R2	RL
Extractive Oracle	54.67	30.37	50.81
Compressive Oracle	**57.12**	**32.59**	**53.27**

Table 1: Oracle scores obtained for the CNN and DailyMail testsets. We report ROUGE-1 (R1), ROUGE-2 (R2) and ROUGE-L (RL) F1 scores.

the Transition-Based Chunking Model (Lample et al., 2016), to annotate spans in every sentence that can be dropped in the final summary. We used the publicly released set of 10,000 sentence-compression pairs from the Google sentence compression dataset (Filippova and Altun, 2013; Filippova et al., 2015) for training. After tagging all sentences in the CNN and DailyMail corpora using this compression model, we generated oracle compressive summaries based on the best average of ROUGE-1 (R1) and ROUGE-2 (R2) F_1 scores from the combination of all possible sentences and all removals of the marked compression chunks.

To verify the adequacy of our proposed oracles, we show in Table 1 a comparison of their scores. Our compressive oracle achieves much better scores than the extractive oracle, because of its capability to make summaries concise. Moreover, the linguistic quality of these oracles was preserved due to the tagging of the entire span by the sentence compressor trained on the sentence compression dataset.[4] We believe that our dataset with oracle compression labels will be of significant interest to the sentence compression and summarization community.

4.2 Training Parameters

The parameters for the loss at the sentence-level were λ_0^s=2 and λ_1^s=1 and at the word-level, λ_0^w=1 and λ_1^w=0.5. We used LSTMs with d = 512 for all hidden layers. We performed mini-batch negative log-likelihood training with a batch size of 2 documents for 5 training epochs.We observed the convergence of the model between the 2nd and the 3rd epochs. It took around 12 hrs on a single GTX 1080 GPU to train. We evaluated our model on the validation set after every 5,000 batches. We trained with Adam (Kingma and Ba, 2015) with an initial learning rate of 0.001. Our system was implemented using DyNet (Neubig et al., 2017).

4.3 Model Evaluation

We evaluated summarization quality using F_1 ROUGE (Lin and Hovy, 2003). We report results in terms of unigram and bigram overlap (R1) and (R2) as a means of assessing informativeness, and the longest common subsequence (RL) as a means of assessing fluency.[5] In addition to ROUGE, which can be misleading when used as the only means to assess summaries (Schluter, 2017), we also conducted a question-answering based human evaluation to assess the informativeness of our summaries in their ability to preserve key information from the document (Narayan et al., 2018c).[6] First, questions are written using the gold summary, we then examined how many questions participants were able to answer by reading system summaries alone, without access to the article.[7] Figure 5 shows a set of candidate summaries along with questions used for this evaluation.

4.4 Model and Baselines

We evaluated our model EXCONSUMM in two settings: Extractive (selects sentences to assemble the summary) and Compressive (selects sentences and compresses them by removing unnecessary spans of words). We compared our models against a baseline (LEAD) that selects the first m leading sentences from each document,[8] three neural extractive models, and various abstractive models. For the extractive models, we used SUMMARUNNER (Nallapati et al., 2017), since it shares some similarity to our model, REFRESH (Narayan et al., 2018c) trained with reinforcement learning and LATENT (Zhang et al., 2018) a neural architecture that makes use of latent variable to avoid creating oracle summaries. We further compare against LATENT+COMPRESS (Zhang et al., 2018), an extension of the LATENT model that learns to map extracted sentences to final summaries using an attention-based seq2seq model (Rush et al., 2015). All models, unlike ours, extract a fixed number of sentences to assemble their summaries. For abstractive models, we compare against the state-of-the art models of POINTER+COVERAGE (See et al., 2017), ML+RL (Paulus et al., 2018), and Tan et al. (2017) among others.

[4] We show examples of both oracles in Appendix §A.1.

[5] We used `pyrouge` to compute the ROUGE scores. The parameters we used were "-a -c 95 -m -n 4 -w 1.2."

[6] We used the CNN/DailyMail QA test set of Narayan et al. (2018c) for evaluation. It includes 20 documents with a total of 71 manually written question-answer pairs.

[7] See Appendix §A.2 for more details.

[8] We follow Narayan et al. (2018c) and set m = 3 for CNN and 4 for DailyMail. We follow Grusky et al. (2018) and set m = 2 for Newsroom.

Models	CNN			DailyMail			Newsroom Ext.			Newsroom Mixed			Newsroom Abs.		
	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL	R1	R2	RL
LEAD	29.1	11.1	25.9	40.7	18.3	37.2	53.1	49.0	52.4	—	—	—	13.7	2.4	11.2
REFRESH	30.0	11.7	26.9	41.0	18.8	37.7	—	—	—	—	—	—	—	—	—
EXCONSUMM Extractive	**32.5**	12.6	28.5	**42.8**	**19.3**	**38.9**	**69.4**	**64.3**	**68.3**	**31.9**	**16.3**	26.9	**17.2**	**3.1**	13.6
EXCONSUMM Compressive	**32.5**	**12.7**	**29.2**	41.7	18.5	38.4	68.4	62.9	67.3	31.7	16.1	**27.0**	17.1	**3.1**	**14.1**
Pointer+Coverage ◇	—	—	—	—	—	—	39.1	28.0	36.2	25.5	11.0	21.1	14.7	2.3	11.4
Tan et al. (2017)*	30.3	9.8	20.0	—	—	—	—	—	—	—	—	—	—	—	—

Table 2: Results on the CNN, DailyMail and Newsroom test sets. We report ROUGE R1, R2 and RL F_1 scores. Extractive systems are in the first block, compressive in the second and abstractive in the third. We use — whenever results are not available. Models marked with * are not directly comparable to ours as they are based on an anonymized version of the dataset. The model marked with ◇ show here the results for the best configuration of See et al. (2017), referred to as Pointer-N in Grusky et al. (2018), which is trained on the whole Newsroom dataset.

Models	CNN+DailyMail		
	R1	R2	RL
LEAD	39.6	17.7	36.2
SUMMARUNNER*	39.6	16.2	35.3
REFRESH	40.0	18.2	36.6
LATENT	41.1	**18.8**	37.4
EXCONSUMM Extractive	**41.7**	18.6	37.8
LATENT+COMPRESS	36.7	15.4	34.3
EXCONSUMM Compressive	40.9	18.0	37.4
Pointer+Coverage	39.5	17.3	36.4
ML + RL*	39.9	15.8	36.9
Tan et al. (2017)*	38.1	13.9	34.0
Li et al. (2018)	39.0	17.1	35.7
Chen and Bansal (2018)	40.4	18.0	37.1
Hsu et al. (2018)	40.7	18.0	37.1
Pasunuru and Bansal (2018)	40.9	17.8	**38.5**
Gehrmann et al. (2018)	41.2	18.7	38.3

Table 3: Results for combined CNN/DailyMail test set.

5 Results

5.1 Automatic Evaluation

Table 2 and 3 show results for the evaluations on the CNN/DailyMail and Newsroom test sets.

Comparison with Extractive Systems. EXCONSUMM Compressive performs best on the CNN dataset and EXCONSUMM Extractive on the DailyMail dataset, probably due to the fact that the CNN dataset is less biased towards extractive methods than the DailyMail dataset (Narayan et al., 2018b). We report similar results on the Newsroom dataset. EXCONSUMM Compressive tends to perform better for mixed (Mixed) and abstractive (Abs.) subsets, while EXCONSUMM Extractive performs better for the extractive (Ext.) subset. Our experiments demonstrate that our compressive model tends to perform better on the dataset which promotes abstractive summaries.

We find that EXCONSUMM Extractive consistently performs better on all metrics when compared to any of the other extractive models, except for the single case where it is narrowly behind LATENT on R2 (18.6 vs 18.8) for the CNN/DailyMail combined test set. It even outperforms REFRESH, which is trained with reinforcement learning. We hypothesize that its superior performance stems from the ability to generate variable length summaries. REFRESH or LATENT, on the other hand, always produces a fixed length summary.

Comparison with Compressive System. EXCONSUMM Compressive reports superior performance compared to LATENT+COMPRESS (+4.2 for R1, +2.6 for R2 and +3.1 for RL). Our results demonstrate that our compressive system is more suitable for document summarization. It first selects sentences and then compresses them by removing irrelevant spans of words. It makes use of an advance oracle sentence compressor trained on a dedicated sentence compression dataset (Sec. 4.1). In contrast, LATENT+COMPRESS naively trains a sequence-to-sequence compressor to map a sentence in the document to a sentence in the summary.

Comparison with Abstractive Systems. Both EXCONSUMM Extractive and Compressive outperform most of the abstractive systems including Pointer+Coverage (See et al., 2017). When comparing with more recent methods (Pasunuru and Bansal, 2018; Gehrmann et al., 2018), our model has comparable performance.

Summary Versatility. We evaluate the ability of our model to generate variable length summaries. Table 4 show the Pearson correlation coefficient between the lengths of the human generated summaries against each unbounded model. Our compressive approach obtains the best results, with a Pearson correlation coefficient of 0.72 ($p < 0.001$).

Figure 4 also shows the distribution of words

Models	Bounded					Unbounded					
	Human QA		ROUGE			Human QA		ROUGE			Pearson
	score	rank	R1	R2	RL	score	rank	R1	R2	RL	r
LEAD	25.50	4rd	30.9	11.9	29.1	36.33	5th	31.6	13.5	29.3	0.40
REFRESH	20.88	6th	37.4	17.3	34.8	**66.34**	1st	**43.8**	**25.8**	**41.6**	0.60
LATENT	38.45	2nd	**38.9**	19.6	36.4	53.38	4th	40.7	22.0	38.1	-0.02
EXCONSUMM Extractive	36.34	3rd	38.4	18.5	35.9	54.93	3rd	40.8	21.0	38.2	0.68
EXCONSUMM Compressive	**39.44**	1ST	38.8	19.0	**37.0**	57.32	2nd	41.4	22.6	39.1	**0.72**
Pointer+Coverage	24.51	5th	38.4	**19.7**	36.7	28.73	6th	40.2	21.4	38.0	0.30

Table 4: QA evaluations: limited length (Bounded) and full length (Unbounded) summaries. We also show ROUGE scores for the summaries being evaluated. We report the Pearson correlation coefficient between the human and predicted summary lengths

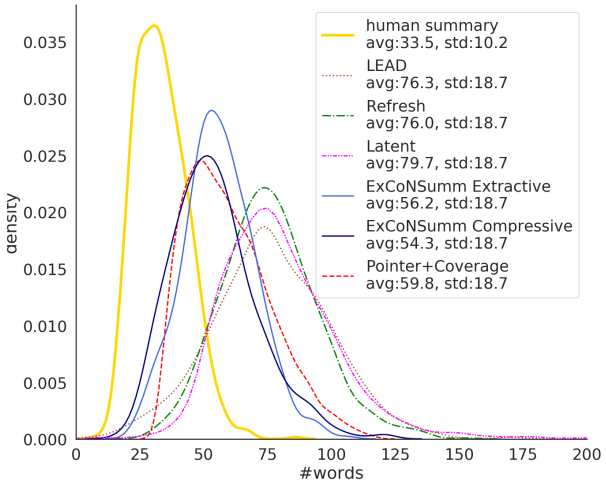

Figure 4: Word distribution in comparison with the human summaries for CNN dataset. Density curves show the length distributions of human authored and system produced summaries.

per summary for the models where predictions were available. Interestingly, both EXCONSUMM Extractive and Compressive follow the human distribution much better than other extractive systems (LEAD, REFRESH and LATENT), since they are able to generate variable-length summaries depending on the input text. Our compressive model generates a word distribution much closer to the abstractive Pointer+Coverage model but achieves better compression ratio; the summaries generated by Pointer+Coverage contain 59.8 words, while those generated by EXCONSUMM Compressive have 54.3 words on average.

5.2 QA Evaluation

Table 4 shows results from our question answering based human evaluation. We elicited human judgements in two settings: the "Unbounded", where participants were shown the full system produced summaries; and the "Bounded", where participants were shown summaries that were limited to the same size as the gold summaries.

For the "Unbounded" setting, the output summaries produced by REFRESH were able to answer most of the questions correctly, our Compressive and Extractive systems were placed at the 2nd and 3rd places respectively.[9]

We observed that our systems were able to produce more concise summaries than those produced by REFRESH (avg. length in words: 76.0 for REFRESH, 56.2 for EXCONSUMM Extractive and 54.3 for EXCONSUMM Compressive; see Figure 4). REFRESH is prone to generating verbose summaries, consequently it has an advantage of accumulating more information. In the "Bounded" setting, we aim to reduce this unfair advantage. Scores are overall lower since the summary sizes are truncated to gold size. The EXCONSUMM Compressive summaries rank first and can answer 39.44% of questions correctly. EXCONSUMM Extractive retains its 3rd place answering 36.34% of questions correctly.[10] These results demonstrate that our models generate concise and informative summaries that correlate well with the human summary lengths.[11]

5.3 Summary State Representation

Next, we performed an ablation study to investigate the importance of the summary state representation $\boldsymbol{o}_i^s$ w.r.t. the quality of the overall sum-

[9] We carried out pairwise comparisons between all models to assess whether system differences are statistically significant. We found that there is no statistically significant difference between REFRESH and EXCONSUMM Compressive. We use a one-way ANOVA with posthoc Tukey HSD tests with $p < 0.01$. The differences among LATENT and both variants of EXCONSUMM, and between LEAD and Pointer+Coverage are also statistically insignificant. All other differences are statistically significant.

[10] The differences among both variants of EXCONSUMM and LATENT, and among LEAD, REFRESH and Pointer+Coverage are statistically insignificant. All other differences are statistically significant. We use a one-way ANOVA with posthoc Tukey HSD tests with $p < 0.01$.

[11] App. §A.2 shows more examples of our summaries.

LEAD
• (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."
• The Al-Malahem Media obituary characterized al-Rubaish as a religious scholar and combat commander.

REFRESH
• (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."
• Al-Rubaish was once held by the U.S. government at its detention facility in Guantanamo Bay, Cuba.

LATENT
• (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike." The Al-Malahem Media obituary characterized al-Rubaish as a religious scholar and combat commander.
• A Yemeni Defense Ministry official and two Yemeni national security officials not authorized to speak on record confirmed that al-Rubaish had been killed, but could not specify how he died.

EXCONSUMM Extractive
• (CNN) A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen, the terror group said, showing the organization is vulnerable even as Yemen appears close to civil war.
• Ibrahim al-Rubaish died Monday night in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

EXCONSUMM Compressive
• A top al Qaeda in the Arabian Peninsula leader–who a few years ago was in a U.S. detention facility–was among five killed in an airstrike in Yemen. • Ibrahim al-Rubaish died in what AQAP's media wing, Al-Malahem Media, called a "crusader airstrike."

Pointer+Coverage
• Ibrahim al-Rubaish was among a number of detainees who sued the administration of then-president George W. Bush to challenge the legality of their confinement in Gitmo. • al-Rubaish was once held by the U.S. government at its detention facility in Guantanamo bay, Cuba.

GOLD
• AQAP says a "crusader airstrike" killed Ibrahim al-Rubaish
• Al-Rubaish was once detained by the United States in Guantanamo

Question-Answer Pairs
• Who said that an airstrike killed Ibrahim al-Rubaish? (AQAP) • What was the airstrike called? (crusader airstrike) • Where was Ibrahim al-Rubaish once detained? (Guantanamo)

Figure 5: Example output summaries on the CNN/DailyMail dataset, gold standard summary, and corresponding questions. The questions are manually written using the GOLD summary. The same EXCONSUMM summaries are shown in Figure 1, but the strike-through spans are now removed.

mary. We tested against a STATE AVERAGING variant, where we replace $\boldsymbol{o}_i^s$ by a weighted average, analogous to Nallapati et al. (2017), $\boldsymbol{o}_i^{avg\ s} = \sum_{i=1}^{j-1} \boldsymbol{e}(s_i)p(z_i \mid \boldsymbol{p}_i^{avg})$, where $\boldsymbol{p}_i^{avg}$ has the same form as $\boldsymbol{p}_i$ but depends recursively on the previous summary state $\boldsymbol{o}_{i-1}^{avg\ s}$. Table 5 shows that using an LSTM state $\boldsymbol{o}_i^s$ to model the current sentences in the summary is very important. The other ablation study shows how learning to extract and compress in a disjoint approach (EXCONSUMM Ext+Comp oracle) performs against a joint learning approach (EXCONSUMM Compressive). We compared summaries generated from our best extractive model and compressed them with a compressive oracle. Our joint learning model achieves the best performance in all metrics compared with the other ablations, suggesting that joint learning and using a summary state representation is beneficial for summarization.

State	ROUGE		
	R1	R2	RL
EXCONSUMM Extractive	**32.5**	**12.6**	**28.5**
STATE AVERAGING	30.0	12.3	26.9
EXCONSUMM Compressive	**32.5**	**12.7**	**29.2**
EXCONSUMM Ext+Comp oracle	25.5	9.3	23.7

Table 5: Summary state ablation for the CNN dataset.

6 Conclusions

We developed EXCONSUMM, a novel summarization model to generate variable length extractive and compressive summaries. Experimental results show that the ability of our model to learn a dynamic representation of the summary produces summaries that are informative, concise, and correlate well with human generated summary lengths. Our model outperforms state-of-the-art extractive and most of abstractive systems on the CNN and DailyMail datasets, when evaluated automatically, and through human evaluation for the bounded scenario. We further obtain state-of-the-art results on Newsroom, a more abstractive summary dataset.

Acknowledgments

This work is supported by the EU H2020 SUMMA project (grant agreement N^{o} 688139), by Lisbon Regional Operational Programme (Lisboa 2020), under the Portugal 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within project INSIGHT (N^{o} 033869), by the European Research Council (ERC StG DeepSPIN 758969), and by the Fundação para a Ciência e Tecnologia through contracts UID/EEA/50008/2019 and CMUPERI/TIC/0046/2014 (GoLocal).

References

Miguel Almeida and Andre Martins. 2013. Fast and robust compressive summarization with dual decomposition and multi-task learning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 196–206.

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*, pages 481–490, Portland, Oregon, USA. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Yen-Chun Chen and Mohit Bansal. 2018. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 675–686. Association for Computational Linguistics.

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.

James Clarke and Mirella Lapata. 2008. Global inference for sentence compression: An integer linear programming approach. *Journal of Artificial Intelligence Research (JAIR)*, 31:399–429.

Georgi Dimitroff, Laura Tolosi, Borislav Popov, and Georgi Georgiev. 2013. Weighted maximum likelihood loss as a convenient shortcut to optimizing the f-measure of maximum entropy classifiers. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 207–214, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

Alexander Dlikman and Mark Last. 2016. Using machine learning methods and linguistic features in single-document extractive summarization. In *DMNLP@PKDD/ECML*.

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 on Text Summarization Workshop - Volume 5*, HLT-NAACL-DUC 2003, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Greg Durrett, Taylor Berg-Kirkpatrick, and Dan Klein. 2016. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1998–2008, Berlin, Germany.

Angela Fan, David Grangier, and Michael Auli. 2018. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54, Melbourne, Australia.

Katja Filippova, Enrique Alfonseca, Carlos A. 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*, pages 360–368. Association for Computational Linguistics.

Katja Filippova and Yasemin Altun. 2013. Overcoming the lack of parallel data in sentence compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1481–1491, Seattle, Washington, USA. Association for Computational Linguistics.

Sebastian Gehrmann, Yuntian Deng, and Alexander Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 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 *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1693–1701. Curran Associates, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Wan Ting Hsu, Chieh-Kai Lin, Ming-Ying Lee, Kerui Min, Jing Tang, and Min Sun. 2018. A unified model for extractive and abstractive summarization using inconsistency loss. *CoRR*, abs/1805.06266.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 260–270. Association for Computational Linguistics.

Chenliang Li, Weiran Xu, Si Li, and Sheng Gao. 2018. Guiding generation for abstractive text summarization based on key information guide network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, volume 2, pages 55–60.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*.

Andre F. T. Martins and Noah A. Smith. 2009. Summarization with a joint model for sentence extraction and compression. In *North American Chapter of the Association for Computational Linguistics: Workshop on Integer Linear Programming for NLP*.

Ryan McDonald. 2006. Discriminative sentence compression with soft syntactic evidence. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 297–304.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: a recurrent neural network based sequence model for extractive summarization of documents. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, pages 3075–3081.

Ramesh Nallapati, Bowen Zhou, and Mingbo Ma. 2016a. Classify or select: Neural architectures for extractive document summarization. *CoRR*, abs/1611.04244.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gulcehre, and Bing Xiang. 2016b. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290, Berlin, Germany. Association for Computational Linguistics.

Shashi Narayan, Ronald Cardenas, Nikos Papasarantopoulos, Shay B. Cohen, Mirella Lapata, Jiangsheng Yu, and Yi Chang. 2018a. Document modeling with external attention for sentence extraction. In *Proceedings of the 56st Annual Meeting of the Association for Computational Linguistics*, pages 2020–2030, Melbourne, Australia.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018b. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018c. Ranking sentences for extractive summarization with reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1747–1759, New Orleans, Louisiana.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Ramakanth Pasunuru and Mohit Bansal. 2018. Multi-reward reinforced summarization with saliency and entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 646–653. Association for Computational Linguistics.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

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.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Short Papers*, pages 41–45, Valencia, Spain.

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, Vancouver, Canada. Association for Computational Linguistics.

Karen Spärck Jones. 2007. Automatic summarising: The state of the art. *Information Processing & Management*, 43(6):1449–1481.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, page 9.

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, Vancouver, Canada. Association for Computational Linguistics.

Xingxing Zhang, Mirella Lapata, Furu Wei, and Ming Zhou. 2018. Neural latent extractive document summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 779–784, Brussels, Belgium.