SCAR: Sentence Compression using Autoencoders for Reconstruction

Chanakya Malireddy
IIIT Hyderabad
chanakya.malireddy
@research.iiit.ac.in

Tirth Maniar
IIIT Hyderabad
tirth.maniar
@students.iiit.ac.in

Manish Shrivastava
IIIT Hyderabad
m.shrivastava
@iiit.ac.in

Abstract

Sentence compression is the task of shortening a sentence while retaining its meaning. Most methods proposed for this task rely on labeled or paired corpora (containing pairs of verbose and compressed sentences), which is often expensive to collect. To overcome this limitation, we present a novel unsupervised deep learning framework (SCAR) for deletion-based sentence compression. SCAR is primarily composed of two encoder-decoder pairs: a compressor and a reconstructor. The compressor masks the input, and the reconstructor tries to regenerate it. The model is entirely trained on unlabeled data and does not require additional inputs such as explicit syntactic information or optimal compression length. SCAR's merit lies in the novel Linkage Loss function, which correlates the compressor and its effect on reconstruction, guiding it to drop inferable tokens. SCAR achieves higher ROUGE scores on benchmark datasets than the existing state-of-the-art methods and baselines. We also conduct a user study to demonstrate the application of our model as a text highlighting system. Using our model to underscore salient information facilitates speed-reading and reduces the time required to skim a document.

1 Introduction

Our fast-paced lifestyle precludes us from reading verbose and lengthy documents. How about a system that highlights the salient content for us (as shown in Fig.1)? We model this problem as the well-known sentence compression task. Sentence compression aims to generate a shorter representation of the input that captures its gist and preserves its intent. Compression algorithms are broadly classified as abstractive and extractive. Extractive compression or deletion-based algorithms only select relevant words from the input, whereas abstractive compression algorithms also allow paraphrasing.

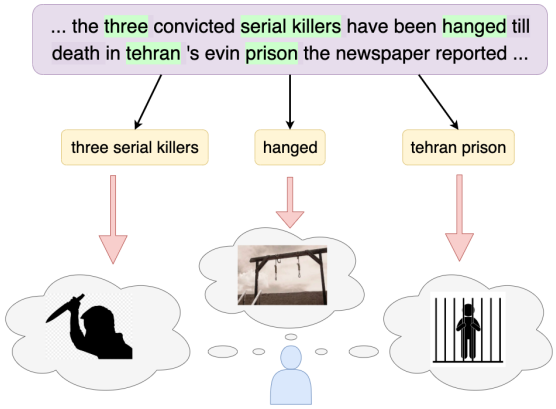

Figure 1: An example of a system that highlights the salient content, allowing the user to skim through the document quickly.

In the past, compression approaches have revolved around statistical methods (Knight and Marcu, 2000) and syntactic rules (McDonald, 2006). Current state-of-the-art methods model the problem as a sequence-to-sequence learning task (Filippova et al., 2015). Although these methods perform well, they require massive parallel training datasets that are difficult to collect (Filippova and Altun, 2013). Recently, unsupervised approaches have been explored to overcome this limitation. Fevry and Phang (2018) model compression as a denoising task but barely reach the baselines. Baziotis et al. (2019) propose SEQ^3, an autoencoder which uses a Gumbel-softmax to represent the distribution over summaries. But a qualitative analysis of their outputs shows that SEQ^3 mimics the lead baseline.

In this work, we present an unsupervised deep learning framework (SCAR) for deletion-based sentence compression. SCAR is composed of a compressor and a reconstructor. For each word in the input, the compressor determines whether or not to include it in the compression. A length loss restricts the compression length. The reconstruc-

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop, pages 88–94

tor tries to regenerate the input using the words retained by the compressor. A reconstruction loss motivates the compressor to include words that aid in reconstruction. However, without an additional loss to govern word masking, the network fails to converge. We introduce a novel linkage loss that ties together the compressor and the reconstructor. It penalizes the network if a) it decides to drop a word but is unable to reconstruct it or b) it decides to include a word which it could reconstruct easily.

2 Related Work

Early compression algorithms were formulated using strong linguistic priors and language heuristics (Jing, 2000; Knight and Marcu, 2002; Dorr et al., 2003; Cohn and Lapata, 2008). McDonald (2006) use syntactical evidence to condition the output of the model. Berg-Kirkpatrick et al. (2011) prune dependency edges to remove constituents for compression.

Deep learning-based approaches have gained popularity owing to their success in core NLP tasks such as machine translation (Bahdanau et al., 2014). Filippova et al. (2015) propose an RNN based encoder-decoder network for deletion based compression. Although this approach achieves superior performance over metric-based approaches, a large amount of paired sentences are needed to train the network.

The first attempt to reduce the dependence on paired corpora for deletion based deep learning compression models was made by Miao and Blunsom (2016). They train separate compressor and reconstruction models, to allow for both supervised and unsupervised training. The compressor consists of a discrete variational autoencoder. The model is trained end-to-end using the REINFORCE algorithm. However, the reported results still use a sizeable amount of labeled data.

Recent approaches have sought completely unsupervised solutions. Fevry and Phang (2018) use a denoising autoencoder (DAE) for sentence compression. The input sentence is shuffled and extended to add noise. DAE tries to reconstruct the original denoised sentence from the noisy input. An additional signal is needed to specify the output length. At test time, the sentence is fed to the model without any noise. In an attempt to denoise the input, the network generates a compressed output. However, the model often fails to capture the information present in the input and is barely able to reach the baselines.

SEQ^3 (Baziotis et al., 2019) proposes an autoencoder using a Gumbel-softmax to represent the distribution over summaries. A compressor generates a summary, and a reconstructor tries to reconstruct the input using the summary. A pre-trained language model acts as a prior, to incentivize the compressor to produce human-readable summaries. An additional topic loss is required to ensure that the summary contains relevant words, making the model non-generic and fine-tuned to the domain. A qualitative analysis of the outputs shows that SEQ^3 merely mimics the lead baseline and generates compressions by blindly copying a prefix of the input.

3 SCAR

SCAR is composed of two encoder-decoder pairs: compressor **C** and reconstructor **R**, as shown in Fig. 2. Given an input sentence $\mathbf{s} = w_1, w_2 ..., w_k$ containing k words, **C** generates an indicator vector $\mathbf{I_v} = I_{v1}, I_{v2}, ..., I_{vk}$ which indicates the presence/absence of each word in the summary. The summary is represented as $\mathbf{s'} = \mathbf{s} \odot \mathbf{I_v}$, where $\odot$ represents element-wise multiplication. Therefore, words corresponding to $\mathbf{I_{vi}} \approx 0$ are effectively skipped. The network tries to reconstruct the input sentence from $\mathbf{s'}$.

Formally, the network tries to find an I_v^* such that the probability $p(\mathbf{s}|\mathbf{s} \odot \mathbf{I_v})$ is maximized and $\sum_{t=1}^{k} I_{vt}$ is minimized, jointly. The probability $p(\mathbf{s}|\mathbf{s} \odot \mathbf{I_v})$ can be decomposed further as shown in Eq.(1)

$$I_v^* = \arg\max_{I_v} \prod_{t=1}^{k} p(w_t|(w_1 \times I_{v1}), ..., (w_{k-1} \times I_{vk-1})) \quad (1)$$

For every word in the sentence, we learn a 300-dimensional embedding initialized with GloVe (Pennington et al., 2014). These embeddings are sequentially fed as input to the **Sentence Encoder** (E_s), composed of a bi-LSTM. The input is fed forwards and backward. The hidden states are a concatenation of the forward and backward states. The sentence representation is obtained from the final hidden state of $E_s(i.e., h_{e1})$. The **Indicator Extraction Module (IEM)**, a bi-LSTM decoder, is initialized using h_{e1}. The output of this decoder at each time step is passed through a network of two fully connected layers to generate

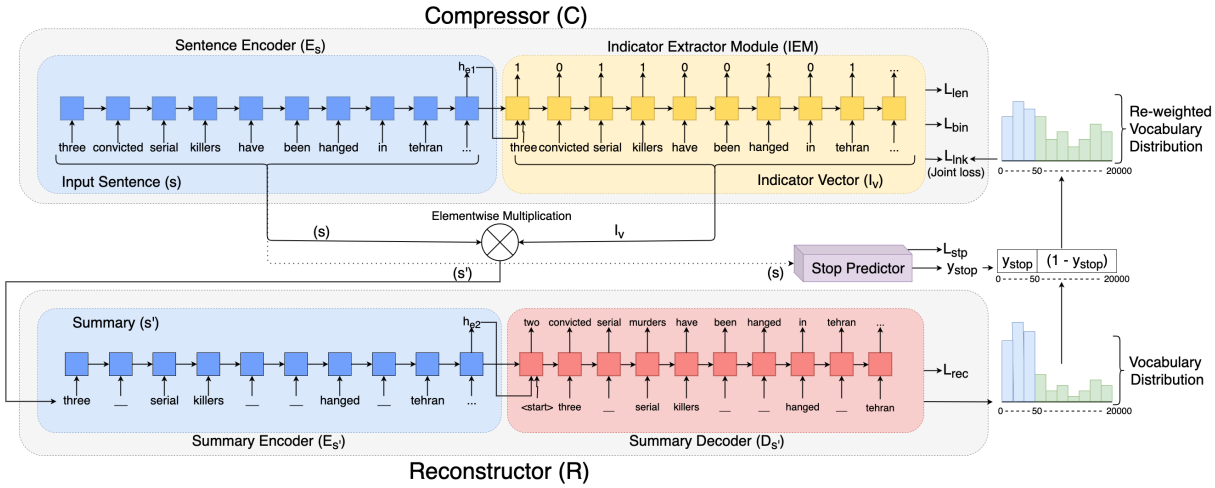

Figure 2: The figure shows the proposed SCAR architecture (details are described in Section 3)

a single indicator value. We intend this value to be close to either one or zero, denoting the presence/absence of each word from the summary.

The masked sentence, $\mathbf{s}' = \mathbf{s} \odot \mathbf{I}_v$, is encoded using the **Summary Encoder** ($E_{s'}$), composed of a bi-LSTM. The **Summary Decoder** ($D_{s'}$), also a bi-LSTM, is initialized using the final hidden state of $E_{s'}(h_{e2})$. This decoder aims to regenerate the input sentence $\mathbf{s}$ from $\mathbf{s}'$. This motivates IEM to generate $\mathbf{I}_v$ such that $\mathbf{s}$ can be easily reconstructed. The output at each time step in $D_{s'}$ is fed to a dense layer, W_s, which computes a distribution over the vocabulary from the decoder's hidden states.

3.1 Loss functions

Compression Length loss (L_{len}) is used to constrain the summary length. It is calculated from the output of IEM as shown in Eq. (2). $Len(\mathbf{s}')$ is the sum of elements of $\mathbf{I}_v$. We set $r = 0.4$ in our experiments.

$$L_{len} = \left(\frac{Len(\mathbf{s}')}{Len(\mathbf{s})} - r\right)^2 \quad (2)$$

Sentence Reconstruction loss (L_{rec}) is applied to ensure $\mathbf{s}'$ contains enough information to reconstruct $\mathbf{s}$. It is calculated from the output of $D_{s'}$ as shown in Eq. (3).

$$L_{rec} = -\sum_{i=1}^{Len(s)} logP(w_i|w'_{<i}, h_{e2}) \quad (3)$$

To help ease reconstruction, L_{rec} steers the network to keep larger summaries, whereas L_{len} forces it to it cut down. This makes it hard for the model to converge optimally. We introduce a novel **Linkage loss** (L_{lnk}), which correlates the indicator vector and its effect on reconstruction. It penalizes the network if a) it decides to mask a word but is unable to reconstruct it or b) it decides to include a word which it could reconstruct easily. It is applied to the outputs of IEM and $D_{s'}$, as shown in Eq. (4).

Ref:	the olympic village for the winter games in turin was officially opened on tuesday
Summ:	___ olympic village ___ ___ winter _____ __ turin ___ _____ opened __ _____
Recon:	the olympic village of the winter olympics a turin was officially opened here wednesday

Figure 3: Linkage loss guides the model to drop words that can be inferred during reconstruction (light green) and retain words that are harder to infer (dark green).

$$L_{lnk} = \sum_{i=1}^{Len(s)} \left(\mathbf{I}_{vi} e^{(1-\chi_i)} + (1-\mathbf{I}_{vi}) e^{\chi_i} - 1\right) \quad (4)$$

The variable $\chi_i \in [0, 1]$, in Eq. (5), is the normalized value of a word's logit in a sentence. It denotes the relative difficulty of decoding word w_i, given $w'_{<i}$ and h_{e2}. L_{lnk} is minimized when either a) $\chi_i = 0$ and $\mathbf{I}_{vi} = 0$ (signifying that w_i is easy to decode and should be dropped) or b) $\chi_i = 1$ and $\mathbf{I}_{vi} = 1$ (signifying that hard-to-decode words should be retained). The effect of L_{lnk} can be seen in Fig. 3. The model retains words with a higher χ_i (dark green), whereas words with a lower χ_i (light green) can be inferred during reconstruction and

therefore dropped.

$$\chi_i = \frac{|logP(w_i|w'_{<i}, h_{e2})|}{\max_{1\leq j\leq Len(s)}|logP(w_j|w_{<j}, h_{e2})|} \quad (5)$$

Binarization loss (L_{bin}) is applied to the output of IEM, as shown in Eq. (6), to push the values of $\mathbf{I}_v$ close to 0 and 1 (since setting them to these hard values directly introduces non-differentiability). In our experiments, b is set to 5 and a is such that L_{bin} is always non-negative. At test time, only the words with $I_{vi} > 0.5$ are included in the compression.

$$L_{bin} = \frac{1}{Len(s)} \sum_{i=1}^{Len(s)} (a - b(I_{vi} - 0.5)^2) \quad (6)$$

3.2 Re-weighting Vocabulary Distribution

Due to the nature of Zipf's law (Zipf, 1949), most of the probability mass in the vocabulary distribution output by the Summary Decoder is retained by *stopwords*. As a result, χ_i corresponding to *stopwords* is much lower compared to *content words*. This causes the network to blindly drop *stopwords* and retain most *content words*. In this case, many content words that may be inferable are not dropped. To remedy this, we introduce **Stop Predictor** (D_{stop}), which assigns a score to the next word based on whether it is a *stopword* or not. When the network believes that the next word is not a *stopword*, it re-distributes the probability mass from *stopwords* proportionally among *content words* and vice-versa.

The word embeddings' of **s** are sequentially fed as input to D_{stop}, a bi-LSTM decoder. The output of D_{stop} at each time step is passed through a network of two fully connected layers to generate a single score, $y_{stop,i} \in [0, 1]$. In order to train D_{stop} we apply L_{stp} (mean-square-error loss with the ground truth) as shown in Eq.(7). The ground truth is obtained from the *stopword-list*, defined as the collection of 50 most frequent words (0.25% of the vocabulary size) found in the dataset.

We re-weight the vocabulary distribution using $y_{stop,i}$, similar to p_{gen} in (See et al., 2017), as shown in Eq. (8). $\mathbb{I}_s$ is a vocabulary sized vector with the 50 elements of *stopword-list* set to 1 and the rest to 0.

$$L_{stp} = \frac{1}{Len(s)} \sum_{i=1}^{Len(s)} (y_{stop,i} - y^{gt}_{stop,i})^2 \quad (7)$$

$$P'(w_i|w'_{<i}, h_{e2}) = softmax(\mathbb{I}_s \cdot y_{stop,i} \cdot P(w_i) + (1 - \mathbb{I}_s) \cdot (1 - y_{stop,i}) \cdot P(w_i)) \quad (8)$$

This re-weighted distribution is plugged into Eq.(5) and used to calculate L_{lnk}.

The final loss function (L) is a linear combination of the above losses. Since this is an unsupervised approach, currently, the weights are experimentally determined. Initial weights for each loss were selected to normalize the output range of all loss functions. We performed a grid search in the neighborhood of these initial weight values to determine optimal weights that maximized the ROUGE scores on the validation set. The weights have been set to 8 (L_{len}), 1 (L_{rec}), 5 (L_{lnk}), 100 (L_{bin}) and 10 (L_{stp}) in our experiments.

3.3 Training

In our experiments, we used the annotated Gigaword corpus (Rush et al., 2015). The model is trained only on the reference section. We only considered sentences where the length was between 15 and 40 words (3.5M samples). A small portion of the training set (200k samples) was held out for validation. The batch size is set to 128. Vocabulary is restricted to 20000 most frequent words from the dataset. All bi-LSTM cells are of size 600 and weights are initialized normally $\mathcal{N}(\mu = 0, \sigma = 0.1)$. The output from IEM and D_{stop} is passed through a hidden layer (150 units) and an output layer with ReLU and sigmoid activations, respectively. We use Adam optimizer (Kingma and Ba) (lr=0.001, β_1=0.9 and β_2=0.999). Gradients larger than 1.0 are clipped. The model is trained for 5 epochs using early stopping by monitoring the performance on the validation set.[1]

4 Experiments

Since the test set of the Gigaword corpus is small (1.9k samples) and does not capture the true behavior of the models, we report our results on the significantly larger validation set (189k samples). Note that SCAR does not make use of the validation set during training, and it can be treated as a test set. We also test (without retraining) SCAR on DUC-2003 and DUC-2004 shared tasks (Over et al., 2007), containing 624/500 news articles each, paired with 4 reference summaries capped at 75

[1] https://github.com/m-chanakya/scar

	Gigaword			DUC-2003			DUC-2004		
	R-1	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**	**R-1**	**R-2**	**R-L**
Baselines									
All-Text	28.07	10.02	24.49	-	-	-	-	-	-
Prefix	26.28	9.54	24.73	20.82	6.14	18.44	22.18	6.30	19.33
Lead50	**30.22**	**10.99**	**27.40**	**20.92**	**6.22**	**18.59**	**22.26**	**6.33**	**19.38**
Unsupervised									
SEQ^3	30.23	10.24	27.26	20.89	6.07	18.54	22.12	6.17	19.29
DAE	26.84	7.35	23.15	18.45	3.94	15.79	20.06	4.73	17.03
SCAR	**29.80**	**7.52**	**26.10**	**21.71**	**4.73**	**18.81**	**22.92**	**5.52**	**19.85**
Supervised									
Seq2Seq	33.72	14.18	30.65	26.12	9.67	23.37	27.31	10.43	24.18
Ablation									
w/o L_{lnk}	27.24	5.16	23.87	20.31	3.41	17.60	19.94	3.25	17.07
w/o D_{stop}	28.86	7.02	25.29	21.46	4.66	18.62	21.94	4.70	19.10
$r = 0.3$	27.80	5.07	24.39	20.25	3.16	17.46	20.28	3.09	17.53
$r = 0.2$	25.36	3.36	22.38	18.97	2.31	16.23	18.43	2.20	15.90

Table 1: Average ROUGE scores on Gigaword and DUC datasets.

bytes. We report average ROUGE (1,2,L) F1 scores (Lin, 2004) obtained by all the models in Table 1.

We compare our model with three standard baselines - **Prefix** (first 8 words for Gigaword/first 75 bytes for DUC), **Lead50** (50% tokens) and **All-Text** (entire input). To compare with supervised approaches, we train a baseline **Seq2Seq** model, similar to (Fevry and Phang, 2018). Finally, we compare our model with the recent unsupervised approaches, **DAE** (Fevry and Phang, 2018) [2], and **SEQ^3** (Baziotis et al., 2019) [3].

4.1 Pitfalls of SEQ^3

Lead50 achieves the highest ROUGE scores, but it does not make for a viable compression method as it blindly drops the latter half of the sentence. The scores obtained by SEQ^3 are strikingly similar to Lead50. The authors of SEQ^3 note that "the model tends to copy the first words of the input sentence in the compressed text". We observed that SEQ^3 introduces very little abstractiveness (only 0.001% of the words are different from the input) and copies the first half of the sentence.

To corroborate our findings, we introduce the notion of *summary coverage*. It is a measure of how well each position of the input is represented in the compression. We divide the input sentence into equal-sized segments and measure how often each segment is included in the compression. We plot the summary coverage for Lead50, SEQ^3, and SCAR, as shown in Fig.4. A visualization is shown in Fig.5. Lead50 and SEQ^3 only cover the first half (initial segments) of the input, leading to incomplete/incorrect compressions. SCAR has more uniform coverage and represents all segments of the input well, leading to more informative compressions.

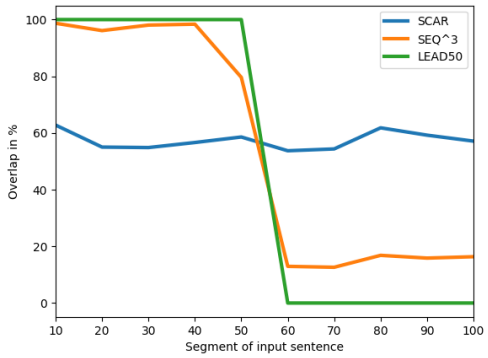

Figure 4: We divide the input sentence into equal-sized segments and measure how often each segment (x-axis) is included in the compression (y-axis).

4.2 Quantitative evaluation

Given the pitfalls of SEQ^3, SCAR achieves state-of-the-art performance in unsupervised sentence compression on Gigaword and DUC datasets. SCAR's R-2 scores on both benchmark sets are low because it tends to drop the inferable portion

[2] https://github.com/zphang/usc_dae
[3] https://github.com/cbaziotis/seq3.git

LEAD50:	malaysia 's government on monday announced an immediate ##-million dollar plan to expand roads , build underground bypasses and overhead bridges to ease kuala lumpur 's traffic jams .
SEQ^3:	malaysia 's government on monday announced an immediate ##-million dollar plan to expand roads , build underground bypasses and overhead bridges to ease kuala lumpur 's traffic jams .
SCAR	malaysia 's government on monday announced an immediate ##-million dollar plan to expand roads , build underground bypasses and overhead bridges to ease kuala lumpur 's traffic jams .
Headline:	malaysia announces ##-million dollar plan to ease kuala lumpur traffic woes

Figure 5: Visualization of summary coverage by overlaying the compressions onto the reference.

Ref (SCAR Highlight)	president bill clinton this week unveils a budget proposal offering nearly ### billion dollars in tax relief over the next six years and calling for the elimination of the federal deficit by #### .
SEQ^3	president bill clinton this week unveils a budget proposal offering nearly ### billion dollars in tax relief deficit (Wrong content retained)
DAE	president bill clinton unveils the federal budget deficit this week by offering nearly ### billion dollars (Wrong content retained)
SCAR	bill this budget proposal nearly billion tax relief next six calling elimination federal deficit
Headline	clinton calls for elimination of the federal deficit by ####

Figure 6: An example of the reference (with SCAR highlight), compressions, and headline.

	Correct	Unsure	Time
Reference	93.4%	6.6%	2m 31s
SCAR (Highlight)	93.4%	6.6%	**1m 54s**
Compressions			
SEQ^3	53.3%	46.67%	**2m 13s**
DAE	26.67%	73.34%	2m 29s
SCAR	**66.67%**	33.33%	2m 42s

Table 2: Average correctness and time scores.

of a bi-gram. Without Linkage loss (L_{lnk}), SCAR loses its ability to drop inferable portions of the input. Without D_{stop}, a mechanism to re-distribute probability mass from stop words, SCAR tends only to drop stopwords. Lower values of r, cause the model to generate smaller compressions. As expected, all of the above factors cause a dip in performance.

4.3 Qualitative evaluation

ROUGE only measures the content overlap and does not account for coherence. We conduct a Qualitative study to address the known issues with ROUGE (Schluter, 2017) and evaluate SCAR's effectiveness as a speed reading system.

Human evaluators are asked to match the reference/compression that they are shown with the correct headline from a set of 5 options. 3 incorrect options are generated by selecting Gigaword headlines that share tokens with the reference. The fifth option is "unsure." Fifteen English speaking participants were divided into 5 sets. They were shown the reference (1), the reference with SCAR highlighting (2), compressions generated by SCAR (3), SEQ^3 (4), and DAE (5), respectively. Each user was asked to match 10 samples.

An example is shown in Fig.6. Compressions generated by DAE fail to preserve the meaning and intent of the reference. SEQ^3 habitually retains the first half of the input, and the evaluators fail to match the headline if it corresponds to the latter half. Due to collocation, SCAR tends to drop the inferable portion of a bi-gram. For example, "Bill" is retained, and "Clinton" is dropped. The average correctness and time scores are reported in Table 2. Compared to other compressions, SCAR has the highest score in terms of correctness. Using SCAR to highlight, reduces reading time by 25%.

5 Conclusion and Future Work

SCAR addresses a significant limitation of the unavailability of labeled data for sentence compression. It outperforms the existing state-of-the-art unsupervised models. Since SCAR learns to drop inferable components of the input and therefore reduces noise, it can be used as a preprocessing step for machine translation and other information retrieval tasks.

References

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

Christos Baziotis, Ion Androutsopoulos, Ioannis Konstas, and Alexandros Potamianos. 2019. Seqˆ 3: Differentiable sequence-to-sequence-to-sequence autoencoder for unsupervised abstractive sentence compression. *arXiv preprint arXiv:1904.03651*.

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

Trevor Cohn and Mirella Lapata. 2008. Sentence compression beyond word deletion. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 137–144. Association for Computational Linguistics.

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*, pages 1–8. Association for Computational Linguistics.

Thibault Fevry and Jason Phang. 2018. Unsupervised sentence compression using denoising auto-encoders. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 413–422, Brussels, Belgium. Association for Computational Linguistics.

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.

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.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Proceedings of the sixth conference on Applied natural language processing*, pages 310–315. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR), arXiv preprint arXiv*, volume 1412.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization – step one: Sentence compression. In *In Proceedings of AAAI*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

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

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

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 319–328, Austin, Texas. Association for Computational Linguistics.

Paul Over, Hoa Dang, and Donna Harman. 2007. Duc in context. *Information Processing & Management*, 43(6):1506–1520.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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: Volume 2, Short Papers*, pages 41–45.

Abigail See, Peter Liu, and Christoper Manning. 2017. Get to the point: Summarization with pointer-generator networks. pages 1073–1083.

George K. Zipf. 1949. *Human Behaviour and the Principle of Least Effort*. Addison-Wesley.