ABSTRACTIVE SUMMARISATION REPRODUCTION

Peter Bell, William Kingsnorth, Bhumika Mistry
Electronics and Computer Science
Southampton University
Southampton, SO17 1BJ, UK
{pjb1g15, wgk1g15, bm4g15}@soton.ac.uk

ABSTRACT

This report investigates the reproducibility of the "A Deep Reinforced Model for Abstractive Summarization" paper submitted to the ICLR 2018 conference. Experiments focus on the effects of a new intra-decoder attention mechanism and the relationship between ROUGE scores and readability as stated in the ICLR paper. This project found the results to be unreproducible by this team, likely due to subtle missing architecture details.

1 INTRODUCTION

This report has been produced to assess the reproducibility of the architecture and results of "A Deep Reinforced Model for Abstractive Summarization" Paulus et al. (2017). This project attempted to implement a model which combines both machine and reinforcement learning as well as a new intra-decoder attention mechanism to produce state of the art results in the problem of abstractive text summarisation. According to the paper, the combination of a classic machine learning approach (the teacher forcing algorithm) with reinforcement learning allowed for enhanced readability of the output results. Furthermore, the new intra-decoder attention mechanism takes the previous decoder output into account to avoid repetition and hence improve performance.

This project had three objectives. Firstly, the paper should be re-implemented to an extent that meets the time and hardware constraints. Secondly experiments should be designed to recreate selected results from the paper. Finally the findings of these experiments should be analysed to conclude whether the papers results are indeed reproducible.

2 TARGET QUESTIONS

The authors primarily explored the following questions:

1. Does using intra-decoder attention improve ROUGE scores?
2. Do ROUGE scores directly correlate to the readability of a summary?
3. Does the use of a mixed training objective improve ROUGE scores?

Though question 2 was not a primary objective of the original paper, this was still an interesting finding to be investigated. Due to time constraints, the reinforcement learning aspect of the paper was not implemented, resulting in question 3 remaining unanswered.

3 IMPLEMENTATION

As no original source code was available on the authors' implementation, the model was re-implemented from scratch. Numerous implementations were available[1,2,3,4] but these were either non-functional or a complex blend of multiple papers. Hence, they were used extremely minimally

[1] https://github.com/ymfa/seq2seq-summarizer
[2] https://github.com/rohithreddy024/Text-Summarizer-Pytorch
[3] https://github.com/OpenNMT/OpenNMT-py/pull/319
[4] https://github.com/oceanypt/A-DEEP-REINFORCED-MODEL-FOR-ABSTRACTIVE-SUMMARIZATION

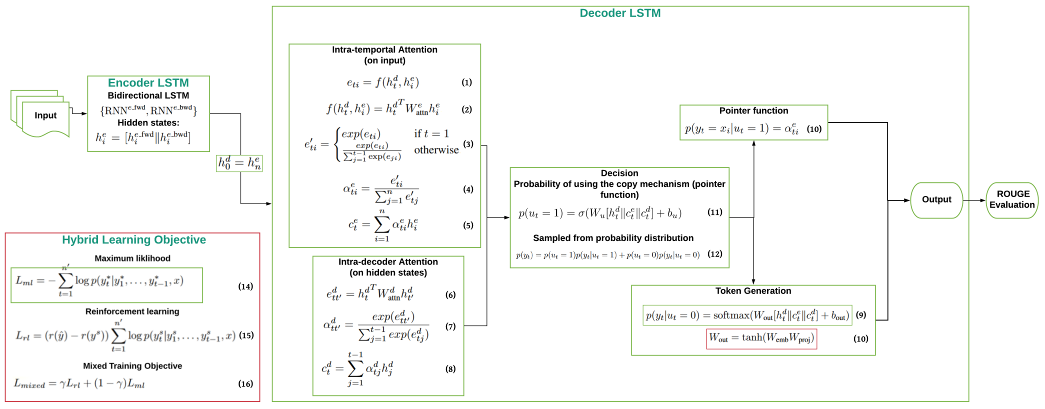

Input
Encoder LSTM
Bidirectional LSTM
$\{\text{RNN}^{e\_\text{fwd}}, \text{RNN}^{e\_\text{bwd}}\}$
Hidden states:
$h_i^e = [h_i^{e\_\text{fwd}} \| h_i^{e\_\text{bwd}}]$
$h_0^d = h_n^e$
Decoder LSTM
Intra-temportal Attention (on input)
$e_{ti} = f(h_t^d, h_i^e)$ (1)
$f(h_t^d, h_i^e) = h_t^{d^T} W_{\text{attn}}^e h_i^e$ (2)
$e'_{ti} = \begin{cases} exp(e_{ti}) & \text{if } t = 1 \\ \frac{exp(e_{ti})}{\sum_{j=1}^{t-1} \exp(e_{ji})} & \text{otherwise} \end{cases}$ (3)
$\alpha_{ti}^e = \frac{e'_{ti}}{\sum_{j=1}^{n} e'_{tj}}$ (4)
$c_t^e = \sum_{i=1}^{n} \alpha_{ti}^e h_i^e$ (5)
Intra-decoder Attention (on hidden states)
$e_{tt'}^d = h_t^{d^T} W_{\text{attn}}^d h_{t'}^d$ (6)
$\alpha_{tt'}^d = \frac{exp(e_{tt'}^d)}{\sum_{j=1}^{t-1} exp(e_{tj}^d)}$ (7)
$c_t^d = \sum_{j=1}^{t-1} \alpha_{tj}^d h_j^d$ (8)
Decision
Probability of using the copy mechanism (pointer function)
$p(u_t = 1) = \sigma(W_u[h_t^d \| c_t^e \| c_t^d] + b_u)$ (11)
Sampled from probability distribution
$p(y_t) = p(u_t = 1)p(y_t|u_t = 1) + p(u_t = 0)p(y_t|u_t = 0)$ (12)
Pointer function
$p(y_t = x_i|u_t = 1) = \alpha_{ti}^e$ (10)
Token Generation
$p(y_t|u_t = 0) = \text{softmax}(W_{\text{out}}[h_t^d \| c_t^e \| c_t^d] + b_{\text{out}})$ (9)
$W_{\text{out}} = \tanh(W_{\text{emb}} W_{\text{proj}})$ (10)
Output
ROUGE Evaluation
Hybrid Learning Objective
Maximum liklihood
$L_{ml} = -\sum_{t=1}^{n'} \log p(y_t^* | y_1^*, \ldots, y_{t-1}^*, x)$ (14)
Reinforcement learning
$L_{rl} = (r(\hat{y}) - r(y^s)) \sum_{t=1}^{n'} \log p(y_t^s | y_1^s, \ldots, y_{t-1}^s, x)$ (15)
Mixed Training Objective
$L_{mixed} = \gamma L_{rl} + (1 - \gamma) L_{ml}$ (16)

Figure 1: Architecture split into its main components, including equations from the paper. Green boarders indicate implemented components whereas red boarders indicate unimplemented components.

as references. A high-level diagram (see Figure 1) was created representing the key components including relevant equations. Following this, lead to the architecture being created in the following a bottom-up manner:

Encoder LSTM→Decoder LSTM→Pointer mechanism→Inter-temporal attention→Inter-decoder attention→Token generation→Teacher-forcing algorithm→ROUGE evaluation

The implementation order was chosen to such that a minimal working network was constructed as soon as possible. Due to time constraints, the reinforcement learning part of the paper and weight sharing (equation 10) remained unimplemented. Implementing the teacher forcing algorithm was prioritised instead, by virtue of it achieving better readability than reinforcement learning individually in the paper. An assumption was made where the authors stated that they used *'two 200-dimensional LSTMs for the bidirectional encoder and one 400-dimensional LSTM for the decoder'* Paulus et al. (2017). This was taken to mean that the hidden sizes of each LSTM were equal to the numbers specified. This seemed to be the more sensible choice compared to the dimensions being multiple LSTMs stacked together as suggested by Zaremba et al. (2014), which would have dramatically increased training time and memory.

For computational efficiency, calculations for both attention mechanisms were completed for all hidden states in a time step rather than individually. Epsilon values (i.e. 10^{-6}) were required to be added to denominators for all divisions, to prevent any division by zero errors. The loss function also ignored any output after the summary length such that padding (needed to fit variable length input and output into the same tensor for batching) was not a factor during training.

The pointer mechanism and token generation components were integrated together by mapping the probabilities of each word to a vector representing the entire vocabulary See et al. (2017). Token generation, by definition already formed a probability distribution over the vocabulary, but the pointer mechanism only returns probabilities for words in the current article being evaluated. Therefore, each probability had to be mapped to the correct position in the vocabulary distribution vector, and the probabilities of repeated words in the article should be added together. This was done with the use of the pytorch function `scatter_add`.

The bilinear layer, used for attention, resulted in increasingly large values after backpropagation, causing a later exponential function to produce integer overflows resulting in NaNs to appear later on when using division operations. To try mitigate against this, the bilinear output (e^t) was normalised for each batch across all time steps. Intuitively this should scale values accordingly, without affecting the attention process. However, this led to performance suffering drastically, with the model unable to learn very effectively at all. It was also found to only postpone the problem. A similar result occurred when replacing the bilinear with a linear layer (which concatenated the two bilinear inputs). The final solution used required applying a regulariser by setting the 'weight decay' parameter to 0.01 for the Adam optimiser, which helped to avoid the weight explosion.

Table 1: Example summary from D2

Source	**Ground truth**	**Our summary**
<sos> all kernels that ever dared approach geoff hinton woke up convolved . <eos>	<sos> kernels that approach geoff hinton get convolved . <\t> <eos>	called ever ever approach geoff dared get

4 Datasets

Although the original paper used both a CNN/Daily Mail (CNN/DM) and a New York Times (NYT) dataset, the NYT data was found to be much harder to obtain as it was locked behind a paywall, and required additional pre-processing as it was in an NITF format. Therefore, only the CNN/DM dataset was used, as a pre-processed version was available for download[5]. This was then converted to GloVe word embeddings and used as in the paper, however 50-dimensional embeddings were used rather than the paper's 100, and 400-token articles were used rather than the original 800. This helped reduce the computational load - meaning the model could fit on the GPUs available to the group - without preventing the model from working as it should, if a little less accurately.

For quick and regular testing during the early development stages, a dummy dataset, D1, was created. This consisted of three sentences, each with four words. Each word was represented by a five-dimensional one-hot encoding. Due to the minuscule nature of the dataset, it would unreasonable to imagine that the model would learn any real semantics, but it was nonetheless a useful tool to ensure data could flow through the model without issue. The second dummy dataset (D2) was created to test the data loading pipeline (as D1 was hard-coded in tensor form in the code itself). This pipeline used GloVe word embeddings as detailed in the paper, however due to limited computational power and lack of available GPU memory, 50-dimensional embeddings were used rather than 100 dimensional ones.

5 Experimental Methodology

Experiments were created to answer each of the target questions from Section 2. Both D2 and the CNN/DM datasets were tested with and without intra-decoder attention, with ROUGE (R-1, R-2 and R-L) scores measured. Printed summaries were also taken using the CNN/DM dataset to examine the readability in relation to the ROUGE scores. On the CNN/DM dataset, a 500-article validation set was used regularly, allowing for fair measurements of the loss over time. A 1000-article test set was used at the end to gain accurate final loss and ROUGE statistics.

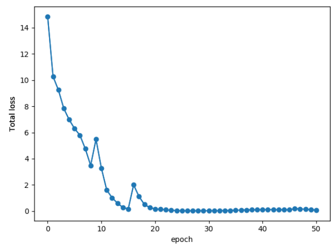
6 Results and Analysis

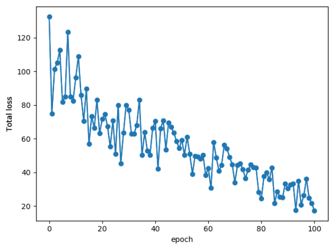
Figure 2 describes how the loss function changes over time during training on each of the three datasets described in Section 5. Figure 2a shows that the model was at least able to learn extremely simple data - this indicates the teacher forcing algorithm was implemented correctly. Figure 2b shows a strong downward trend, but even after 100 epochs the loss is still far from 0. One would instead expect very swift overfitting, as it is being repeatedly shown the same six sentences. The large fluctuations in loss are likely due to the small batches, but also highlight the complexity of the landscape in abstractive summarisation problems.

Unfortunately, on the CNN/DM dataset, all experiments showed large variations in loss even over a constant validation dataset (Figure 2c). This shows that though the model can learn over small datasets, it struggles to learn over larger articles (despite the slight downward trend in loss). One possible reason for this is that larger, more complex articles require a longer training and more data to learn, which, due to large VRAM requirements, was infeasible. Furthermore, the model was found to be very sensitive to the learning rate, implying further tuning could lead to better performance.

Table 2 shows impressive D2 ROUGE scores, but this is to be expected with such short sentences. Still, similar to the paper, R-2 scores were much smaller than R-1 and R-L. The impact of attention in the original paper is much lower for the NYT data (+1.43) than the CNN/DM set (+3.15), suggesting

[5] `https://github.com/rohithreddy024/Text-Summarizer-Pytorch`


(**a**) D1 Epoch Loss


(**b**) D2 Epoch Loss

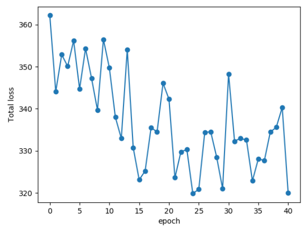

(**c**) CNN/DM Epoch Loss

Figure 2: Epoch vs Loss graphs for datasets D1, D2 and CNN/DM

Table 2: ROUGE Findings - no trigram avoidance

		R-1		**R-2**		**R-L**	
Dataset	**Intra-Attention**	**Ours**	**Original**	**Ours**	**Original**	**Ours**	**Original**
D2	✘	72.24	-	35.85	-	76.35	-
	✓	72.15	-	37.75	-	76.28	-
CNN/DM	✘	13.02	35.15	0.27	13.28	15.63	32.13
	✓	12.60	38.30	0.30	14.81	11.53	35.49

that the summary length does impact the ROUGE score (NYT contains smaller summaries than CNN/DM). The intra-decoder attention did not have a significant effect on the ROUGE scores, indicating that the summaries were too short to make use of the attention.

On the CNN/DM dataset, ROUGE scores were not nearly as high as the original paper's, which is unsurprising given the previously discussed extreme stochasticity in loss. The R-2 scores being close to zero are particularly disappointing, as it suggests that the R-1 scores are merely due to the pointer mechanism copying large portions of the sentences out of order.

7 Conclusion

Overall, the results were disappointing. Metrics were not close to the original paper, and model outputs were largely unreadable. Considering the target questions, intra-attention did not appear to improve ROUGE scores (besides R-2), on any dataset. Even on D2, the output statements were often not particularly readable, despite relatively high ROUGE scores. Reasons for this could be a lack of computational power leading to reduced training time, but it is more likely there were issues with the architecture which prevented effective model training. Also, reducing the article length and embedding dimensions may have been a factor in some issues, however the large disparity between this paper's results and the original paper suggests that there are larger issues at play. While it cannot be declared that the original paper is unreproducible, the results reported in the original paper were certainly unable to be reproduced in this case.

References

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

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

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.