Addressing issues in training Neural Extractive Summarisation models

by

Ramkishore Saravanan, Nikhil Pinnaparaju, Vasudeva Varma

: 1
-9

Report No: IIIT/TR/2021/-1

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
December 2021

Addressing issues in training Neural Extractive Summarisation models

Ramkishore Saravanan[1],
Nikhil Pinnaparaju[2], and Vasudeva Varma[3]

[1] ramkishore.s@research.iiit.ac.in
[2] nikhil.pinnaparaju@research.iiit.ac.in
[3] vv@iiit.ac.in
International Institute of Information Technology, Hyderabad,
Telengana, India - 500032
https://www.iiit.ac.in

Abstract. In many works, the problem of extractive summarisation has been framed as the problem of extracting the best summary from a given document. Many popular recent works aim to solve this by employing neural networks. And many of them are trained using a very limited scope, for example, a vast majority of the neural models are trained only using the best summary. Some also consider pruning useless summaries using other models.
In this work, we show the problems that can arise when training neural models using such methods. We analyse those problems in some major milestones in Neural Extractive Summarisation. We also show and demonstrate ways to overcome them experimentally.

Keywords: Extractive Summarisation · training bias

1 Introduction

Automatic Summarisation aims to shorten the content a given document in such a way so that only the salient content is retained. Extractive summarisation is a sub domain of Automatic Summarisation where the contents of the document are either extracted or copied from the document itself to form the summary. The extracted items can be words, phrases, sentences or even paragraphs. In this way, Extractive Summarisation differs from Abstractive Summarisation, which aims to paraphrase the salient content of the document. But Extractive Summarisation enjoys a lot of popularity over Abstractive Summarisation due to its inherent ability to produce grammatically correct sentences, especially when sentences are extracted.

This work mainly deals with extractive summarisation, especially with models that extract sentences to construct summary given a document. For this paper, we mainly consider CNN/DailyMail dataset introduced by [6].

Extractive Summarisation has been modelled as the problem of selecting sentences which have the maximum ROUGE score. This involves selecting a

subset of sentences which best represent the document as summary from among other subsets. ROUGE scores are not additive and hence simply selecting the best sentences alone is not enough. We also need to check the compatibility of these sentences with each other. This makes this problem of summarisation different from a ranking problem. (This issue was quantitatively shown by Zhong et al [20])

One issue with training a network directly for selecting a subset from among subsets (as opposed to selecting sentences from among all sentences) is the combinatorial explosion problem. As the number of sentences increase, the number of possible candidate summaries increase exponentially. For example, a document of length 25 can have as many as 2600 possible candidate summaries, only considering those made of 2 and 3 sentences.

Many popular works try to model Extractive Summarisation by avoiding this combinatorial explosion problem altogether. Many model the problem as Sequence to Sequence [9, 2], Sequence Labelling [11, 1], Reinforcement Learning [4, 12] and Representation Learning [20]. Yet, except for Representation Learning, all of them use sentence ranking to construct the predicted summary. Post processing methods like Tri-gram blocking also helps a lot whenever sentence ranking is involved. One main issue with sentence ranking approaches is that they cannot optimize for ROUGE directly as shown by [12]

Zhong et al [20], on the other hand, optimize for rouge directly, since they try to rank summaries according to their rouge scores. But then, they are facing the problem of combinatorial explosion head on. To avoid this problem, they used another previous state-of-the-art model, BertSum, [9] to shortlist candidate summaries to train their model on.

In almost all of the cases, the number of possible summaries shown to the model during training is very very less than the actual number of possible summaries. And since most of these summaries are very similar to each other (as shown in section 2), such shortsighted training can lead to a lot of issues with the model. In this work, we explore how optimising only for a small set of summaries can sabotage a model, and verify the same with quantitative experiments on some recent works.

Specifically, our contribution in this work is two fold:

- We showcase the bias in two different kinds of seminal works in Extractive Summarisation quantitatively, and postulate reasons behind it and also show some evidence.
- We demonstrate the importance of training models using multiple summaries.

The content of the paper is organised as follows. Section 2 discusses Dataset Statistics which will be helpful for making analysis in the following sections. Section 3 demonstrates issues in the existing work and the corresponding experiments conducted to reach the conclusions. Section 4 shows the importance of having variance when training extractive summarisation models using a simple experiment. Section 5 discusses Results, Section 6 Expeirmental Details, and Section 7 Conclusions.

2 Dataset Overview

In this section, we will summarize statistics that are important to our work which will be useful when making conclusions in further sections.

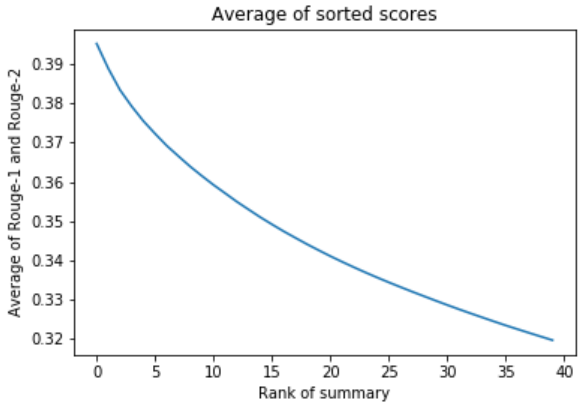

Fig. 1. Here we show the average of ROUGE scores of summaries when sorted according to their rank on randomly selected 10,000 documents. Notice that the top 5 candidates are within a 0.02 margin of each other, which makes them qualitatively very hard to distinguish.

- The average number of unique sentences in top 10 summaries is around 8.4, while the average in top 20 summaries is 12.6
- Each document contains many possible candidate summaries. The distinction between the neighbouring summaries are not really meaningful, atleast from ROUGE's perspective, since all of them have very similar scores. We show this in Figure 1. We can see that the top 5 candidates are within a 0.02 margin of each other, which makes it qualitatively very hard to distinguish between them.
- On average, each document contain around 25 sentences, which brings the average number of possible candidate summaries to 2600.

3 Issues with existing models

In this section, we analyse two seminal works in the field of Extractive Summarisation. The two works that we consider are 1) BertSum [9] (representing sentence extractors, where the model constructs summaries by ranking sentences), 2) MatchSum [20] (representing summary extractors, where the model constructs summaries by ranking summaries). We are mainly interested in the caveats that occur due to the limited scope of data shown to the models during training.

3.1 BertSum

Model Overview BertSum [9] is a Bert model [3] fine-tuned for Extractive Summarisation. It is a sentence ranking model, providing a score for each sentence, and the top 3 sentences (after trigram-blocking) are considered as summary. For details, we refer the reader to [9]

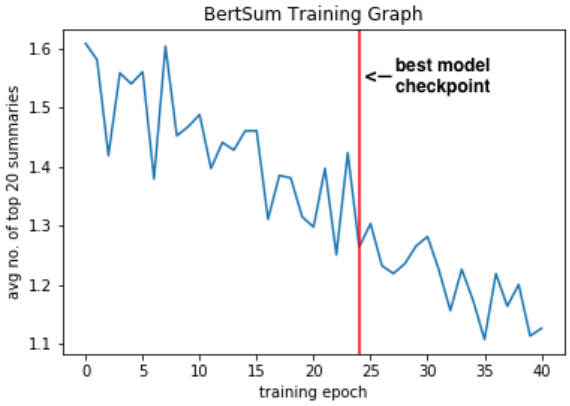

Fig. 2. Here we show the number of actual top 20 candidate summaries that are among those that can be constructed from the top 5 sentences of BertSum on Test Set, as training progresses. We notice that the number is decreasing with increasing epochs. This could either mean 1) That the model is learning to predict the correct first summary with increased accuracy, while ignoring others, or 2) The model is overfitting the first summary, since that is the only summary it ever sees.

Issue 1. We notice that the candidate summaries generated by the model are the best at just after 1000 training steps, after which the quality of those candidates slowly deteriorate. We show this in Figure 2. This figure shows the average number of candidates which can be constructed from the top 5 sentences predicted by the model on test set. Each unit on x axis consists of 1000 training steps. At the end of every 1000 steps, we plot the model's quality of candidates. Notice how the average number of candidates keep dropping as training progresses. This is one issue which can greatly affect MatchSum [20], since they use the candidates generated by BertSum. Had they selected a savepoint which wasn't the best in predicting the best summary, and went with a model which hadn't even finished its warmup training steps (warmup steps = 10000), they might have had better candidates.

The gradual decrease in the quality of candidates can be due to either of the following reasons: 1) The model is learning the nuances of predicting the best summary, 2) The model is overfitting the training data. We examine what the model is doing next.

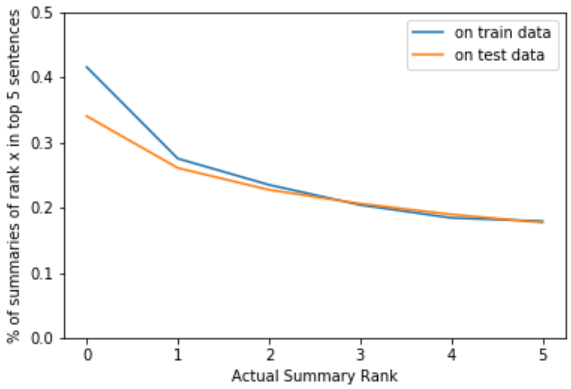

Fig. 3. Here we show the BertSum's bias for the first summary during training, which is not as prelevant on test set. Notice how the slope is lower on the testset. This could be due to the model not being able to distinguish as much between 1st, 2nd and 3rd summaries, since they are similar, while they are overfitting the training data.

Issue 2. Since the model is trained using only one summary, and since there is not much difference between the top summaries (as noted in section 3), we suspect that the model should overfit the best summary on training data, while should not be able to distinguish as much between the top summaries on test data. We show this in Figure 3. The difference in accuracy in predicting the first summary between train data and test data is almost 10%, 0.7% for the second summary and almost negligible from the third.

We postulate that this discrepancy arises between the train and test data because the model is expected to do a lot with a limited architecture. By showing the model only the best summary during training, we expect the model to ideally also learn to select the best summary on test set. But since there isn't much difference between the first best summary, the second and the third, even qualitatively w.r.t to the evaluation metric, we cannot expect the model alone to learn it. This leads to overloading, where the model should not only determine which combination of sentences to select as a good summary, but also which combinations to avoid, without us actually teaching the model how to do so. Due to this, we believe the model struggles during inference to single one out of them over others.

3.2 MatchSum

Model Overview MatchSum [20] is a Siamese-Bert Model [15]. The model is directly trained to rank summaries by constructing representation for each of them. To solve the problem of Combinatorial Explosion, MatchSum [20] uses only candidate summaries constructed from the top 5 sentences predicted by BertSum [9]. For more details, we refer the reader to [20]

Issue 1. From Section 3, we can see that the top 20 summaries of a document on average contain around 12 sentences, which is way higher than the 5 in the 20 candidate summaries used by MatchSum. Not only that, MatchSum has also never looked at a bad summary during training. And since all the candidates are very similar, we suspected that there is liitle reason for MatchSum to actually learn to match text using their semantics. We postulate that MatchSum is learning some other features which help distinguish better summaries from good ones, since semantics alone might not be the best way to distinguish between them. We suspect that, due to the way the model is trained, this model should fail on candidates which have more variance.

We show this in table 1, where the model is asked to predict the ranks of 120 candidate summaries on the test set. We can immediately see that the model performs even worse than BertSum, whose predictions were used to train this model. This shows that the model has actually not learned to match the semantics of the content it encodes, but rather even more nuanced features.

4 Importance of training with variance

In this section, we demonstrate the limitations of training with limited scope (either using one/small set of summaries) using a simple experiment. We train a simple summary ranking model and compare it with the existing state-of-the-art models on how well they are able to rank summaries. We explain the details of the model briefly below.

4.1 Model details

We modify the architecture of BertSum [9] to allow for summary ranking. We use the same input representation as BertSum, and get the representation for each sentence. To get representations of summary, we use an LSTM based Summary encoder, which encodes multiple sentence representations. The final hidden state of the summary encoder is considered as the representation. We consider the representation of the document to be summary encoder's final state for all the sentences in the document. The cosine distance between the summary representation and the document representation is considered the score of the summary. We train our network as a ranking network using Margin-ranking-loss, similar to MatchSum [20].

We train two models, one (Un-Randomized SR) for which we sample summaries at every step and one (Randomized SR) which uses a static summaries of summaries to train. MatchSum uses a static list of summaries to train, and we can see the effects of such training impacting the ability to rank clearly in table 1.

4.2 Evaluation

To evaluate, we take the top 20 summaries predicted by each model and compare it with the actual top k summaries of the document, for various values of k.

Table 1. This table shows the ability of the model to rank high quality summaries higher. In each column, the number at the top represents the average top k summaries per document was among the top 20 summaries selected by the model. The number in the bottom shows the percentage of documents which had at least one summary among the actual top k in its top 20 summaries.

	Top 1	Top 3	Top 5	Top 10	Top 20
BertSum	0.15	0.37	0.54	0.89	1.42
	(15%)	(28%)	(36%)	(46%)	(57%)
MatchSum	0.07	0.17	0.27	0.49	0.86
	(7%)	(14%)	(20%)	(29%)	(41%)
Un-Randomized SR	0.13	0.3	0.43	0.72	1.17
	(13%)	(23%)	(29%)	(38%)	(49%)
Randomized SR	**0.18**	**0.44**	**0.65**	**1.07**	**1.71**
	(18%)	**(31%)**	**(38%)**	**(48%)**	**(60%)**

Specifically, we compute the precision with which each model is able to predict the top 20 summaries correctly. We show this result in table 1. We also show the percentage of documents in test set which had atleast 1 summary out of the actual k summaries in their top 20 predictions. We score the summaries using the perl script provided by Lin et al. [8].

5 Results

In Table 1, we show the qualitative results. We can see immediately that MatchSum [20] performs worse than BertSum [9], on whose candidates it was trained on. We suspect that this is due to the fact that MatchSum is trained only on very similar candidate summaries. Since the content of the candidates were similar, it is highly possible that MatchSum learned much more nuance features to distinguish between them which wasn't semantic based. If so, it can explain why MatchSum would perform badly when tested on candidates with a lot of variance in their content. For comparision, MatchSum's 20 candidates only contained on 5 sentences per document, while our 165 candidates contained on average around 16 sentences.

We also note the difference between the variants of our proposed model. We can see that the UnRandomized model performs poorly compared to the Randomized model. The act of randomizing the candidates during training made it difficult for the model to overfit the training data.

6 Experimental Details

Models For BertSum, we trained our own model following the implementation details from [9], and we produce similar results. For MatchSum, we use the model

provided by [20] on their github repository [4]. We use Bert based models for all cases. All of our experiments were done using PyTorch [13] and Transformers Library [19].

Hyper-parameters We experimented with multiple hyper-parameter setting for our models and found the below to work best on the validation set. Most of them are the default for training Bert. Our learning rate function, dropouts and regularisation are the same as [9]. We save the model every 1000 training steps, and use validation set to select the best model. Our effective batch size was 32 for all models, and whenever we weren't able to fit 32 in a batch, we used gradient accumulation every 2 steps with batch size of 16, effectively same as batchsize 32. For all models (except MatchSum), we trained using 4 Nvidia GTX 1080 Ti and 120 GB of RAM.

7 Conclusions

In this work, we showed the dangers of using ROUGE to evaluate only the best summaries produced by Extractive Summarisation models and presented some means to overcome them. We also showed how important the process of candidate selection is to train these models.

References

1. K. Al-Sabahi, Z. Zuping, and M. Nadher. 2018. A Hierarchical Structured Self-Attentive Model for Extractive Document Summarization (HSSAS).IEEE Access6 (2018), 24205–24212. https://doi.org/10.1109/ACCESS.2018.2829199
2. Jianpeng Cheng and Mirella Lapata. 2016. Neural Summarization by ExtractingSentences and Words. arXiv:1603.07252 [cs.CL]
3. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT:Pre-training of Deep Bidirectional Transformers for Language Understanding.arXiv:1810.04805
4. Yue Dong, Yikang Shen, Eric Crawford, Herke van Hoof, and Jackie Chi KitCheung. 2019. BanditSum: Extractive Summarization as a Contextual Bandit.arXiv:1809.09672
5. Kavita Ganesan. 2018. ROUGE 2.0: Updated and Improved Measures for Evaluation of Summarization Tasks. arXiv:1803.01937 [cs.IR]
6. Karl Moritz Hermann, Tomáš Kočisk'y, Edward Grefenstette, Lasse Espeholt, WillKay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read andcomprehend.arXiv preprint arXiv:1506.03340(2015).
7. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory.Neural Comput.9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
8. Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries.InText Summarization Branches Out. Association for Computational Linguistics,Barcelona, Spain, 74–81. https://www.aclweb.org/anthology/W04-1013

[4] https://github.com/maszhongming/MatchSum

9. [9]Yang Liu. 2019.Fine-tune BERT for Extractive Summarization.arXiv:1903.10318 [cs.CL]
10. Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven-Bethard, and David McClosky. 2014. The Stanford CoreNLP natural languageprocessing toolkit. InProceedings of 52nd annual meeting of the association forcomputational linguistics: system demonstrations. 55–60.
11. Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A Recurrent Neural Network Based Sequence Model for Extractive Summarizationof Documents.Proceedings of the AAAI Conference on Artificial Intelligence31, 1(Feb. 2017). https://ojs.aaai.org/index.php/AAAI/article/view/10958
12. Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018.Rank-ing Sentences for Extractive Summarization with Reinforcement Learning.arXiv:1802.08636 [cs.CL]
13. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang,Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer.2017. Automatic differentiation in PyTorch. InNIPS-W.
14. Martin F Porter. 1980. An algorithm for suffix stripping.Program(1980).
15. Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddingsusing Siamese BERT-Networks. arXiv:1908.10084 [cs.CL]
16. Yang Shao. 2017. Hcti at semeval-2017 task 1: Use convolutional neural networkto evaluate semantic textual similarity. InProceedings of the 11th InternationalWorkshop on Semantic Evaluation (SemEval-2017). 130–133.
17. Raghuram Vadapalli, Litton J Kurisinkel, Manish Gupta, and Vasudeva Varma.2017. SSAS: Semantic Similarity for Abstractive Summarization. InProceedings ofthe Eighth International Joint Conference on Natural Language Processing (Volume2: Short Papers). Asian Federation of Natural Language Processing, Taipei, Taiwan,198–203. https://www.aclweb.org/anthology/I17-2034
18. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones,Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is AllYou Need. arXiv:1706.03762 [cs.CL]
19. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue,Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, JoeDavison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu,Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest,and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural LanguageProcessing. InProceedings of the 2020 Conference on Empirical Methods in NaturalLanguage Processing: System Demonstrations. Association for ComputationalLinguistics, Online, 38–45.https://www.aclweb.org/anthology/2020.emnlp-demos.6
20. Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, andXuanjing Huang. 2020.Extractive Summarization as Text Matching.arXiv:2004.08795