MSBERT: Multi-Step Document Summarization with BERT

Udhav Sethi
School of Computer Science
University of Waterloo
udhav.sethi@uwaterloo.ca

Abstract

Some recent works have shown promising results in using pre-trained Transformer models like Bidirectional Encoder Representations from Transformers (BERT; [1]) for a variety of NLP tasks, including abstractive summarization. One of the obstacles of abstractive summarization is the presence of unimportant information in the input text, which acts as training noise. In this paper, we propose a new approach that performs an extra extractive step, the output of which is then used to condition the abstractive model on only the relevant information before being assigned the task of generating a summary. We compare the performance of our approach against the previous BERT based approach BERTSumExtAbs [2] as the baseline, and achieve better ROUGE-1, ROUGE-2 and ROUGE-L scores on the CNN/DM dataset [3].

1 Introduction

With a vast amount of data available in today's digital world, users are constantly faced with the problem of sifting through large amounts of text to find relevant content. Text summarization is a machine learning technique that aims to solve this problem by automatically generating coherent and concise summaries from vast amounts of text, while preserving the most important information. There are two main methods of summarization: *Extractive* and *Abstractive*. Extractive summarization is the method of extracting the most important sentences in a document and combining them to form a summary. Under Abstractive summarization formulation, the task is to construct summaries containing words or phrases not originally present in the input text, and thus may require language generation capabilities.

The summarization task requires broad-scope natural language understanding in order to produce meaningful and effective summaries. Large-scale pretrained natural language models such as BERT [1], ELMo [4] and GPT [5] have shown promising results in a wide variety of NLP tasks due to their ability to learn contextualized language representations. Recently, these models have also advanced the state of the art in the task of both extractive and abstractive text summarization ([2], [6]).

Previous work ([7] ,[8]) has shown that using extractive objectives can boost the performance of abstractive summarization. In a recent work ([9]), the authors reinforced this idea by showing that an extra extractive step helps an abstractive summarization system produce cogent yet succinct summaries. In this work, we take this idea further by utilizing both extractive and abstractive summarization capabilities of BERT to generate effective abstractive summaries. We hypothesize that a simple extractive step helps introduce enough inductive bias for the abstractive summarization system to produce fluent and precise summaries. Specifically, the extractive summarization task helps extract the most relevant sentences from long documents, which may then be used to construct concise abstractive summaries (See Figure 1). To this end, we propose a Multi-Step BERT (MSBERT) summarization architecture comprising of an extractive step and an abstractive step. We show that

Submitted to Deep Learning and Natural Language Processing (CS886, 2020 Winter). Do not distribute.

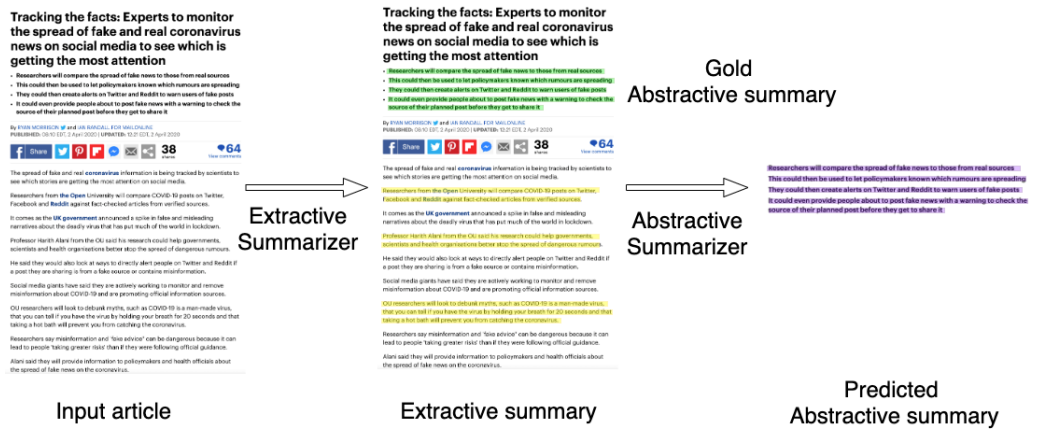


Figure 1: An example DailyMail article and the proposed method for its abstractive summarization. The text highlighted in yellow represents the extracted sentences. The text highlighted in violet is the summary generated by the model. The text highlighted in green comprises the story highlights and is used as the gold abstractive summary.

the extractive step improves summarization results on the CNN/DailyMail dataset. The contribution of this work are two-fold as follows:

- We present a Multi-Step summarization architecture that leverages both extractive and abstractive summarization capabilities of BERT to produce relevant yet concise summaries.
- We quantitatively measure the positive impact of adding an extra extractive step by comparing our performance with the the previous BERT based model BERTSumExtAbs [2] that only sees the input text itself. We achieve superior performance on all the three ROUGE F1 scores on the CNN/DailyMail dataset.

2 Related Work

The summarization task aims to produce a condensed form of a given input text, while preserving its most salient information. The earliest summarization techniques focused on extractive objectives. While extractive methods help identify the most important information, they are lacking in fluency and coherence. Abstractive summarization shows promise in overcoming these drawbacks, as it aims to generate novel sentences not originally seen in the input text.

Pre-trained encoder-decoder models have recently shown impressive results in various natural language tasks. In these models, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $\mathbf{z} = (z_1, ..., z_n)$. Given $\mathbf{z}$, the decoder then generates an output sequence $(y_1, ..., y_n)$ of symbols one element at a time. At each step the model is auto-regressive, consuming the previously generated symbols as additional input when generating the next. Fine-tuning Bidirectional Encoder Representations from Transformers (BERT) [1] to learn better representations for down-stream tasks have become popular. More recently, these attention-based encoder decoder paradigm has been widely adopted to the summarization tasks [2, 10] demonstrating good results. More sophisticated models like [11] [12], and [13] also apply pre-trained language models to the text summarization tasks to achieve higher performance. A recent work [9] shows that that a simple extractive step helps an abstractive summarization system to produce fluent yet precise summaries.

3 Methodology

Our proposed model, MSBERT, derives some part of the architecture from the model BERTSumExtAbs proposed by [2] to take advantage of the pre-trained language model. While there has been recent work that has further advanced the state of the art since BERTSumExtAbs, we use this model to test our hypothesis, and leave it to future work to apply the same idea to other models.

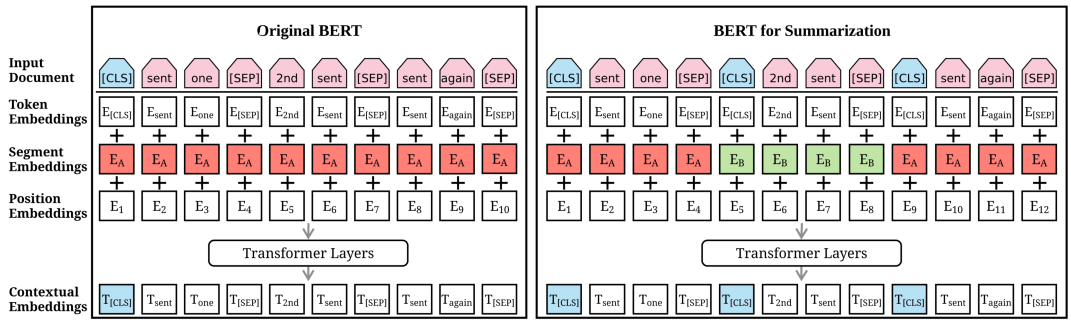


Figure 2: Architecture of the original BERT model (left) and the BERT model modified for summarization (right)

In this section we first discuss how BERT is modified for the summarization task in Section 3.1, and then we introduce our model architecture in Section 3.2 and elaborate on its two components in Sections 3.3 and 3.4.

3.1 BERT for summarization

In order to apply BERT to the summarization problem, we modify its input sequence and embeddings following the approach in [2]. First, we insert [CLS] tokens at the beginning of each input sentence, instead of just the first sentence as in vanilla BERT. This is done to represent individual sentences; each [CLS] symbol represents the sentence preceding it. Thus, the vector corresponding to the i-th [CLS] symbol in the BERT output layer is used as the representation for the i-th sentence. Second, we use interval segment embeddings to distinguish between the multiple sentences of the input document. The odd numbered sentences are assigned a segment embedding E_A while the even numbered sentences are assigned a segment embedding E_B. Several summarization specific layers are stacked on top of the BERT output to capture the document level features essential for summarization. The modified BERT architecture for summarization is illustrated in Figure 2.

3.2 Model Architecture

The architecture diagram of our approach is outlined in Figure 3. In our formulation, we define a multi-step architecture comprising of an extractive step and an abstractive step. The extractive step receives a set of input sentences $S_{doc} = \{S_1, S_2, ..S_n\}$ that comprise the input document. In turn, it outputs a set of sentences $S_{ext} = \{S'_1, S'_2, ..S'_k\}$ comprising the extractive summary, with the requirement that $S_{ext} \subseteq S_{doc}$. Successively, the abstractive step consumes S_{ext} and outputs another set of sentences $S_{abs} = \{A_1, A_2, ..A_m\}$, where $S_{abs} \nsubseteq S_{ext}$, i.e., the abstractive step generates a set of new sentences that paraphrase the key concepts in the extractive summary. The set of sentences S_{abs} generated by the abstractive step is designated for consumption by the reader. It is important to note that the extractive and abstractive steps that constitute our model are two distinct and independently trainable components. These steps are described in the following sections in detail.

3.3 Extractive Step

The extractive summarization task can be formulated as a problem of binary classification, where for each sentence we assign a label $y_i \in \{0, 1\}$ indicating if the sentence is included in the summary. The extractive step comprises of pre-trained BERT modified for summarization as described in 3.1, along with an inter-sentence transformer as the summarization layer on top of the BERT outputs (Figure 4(a)):

$$g^l = \text{LN}(h^{l-1} + \text{MHAtt}(h^{l-1}))$$

$$h^l = \text{LN}(g^l + \text{FFN}(g^l))$$

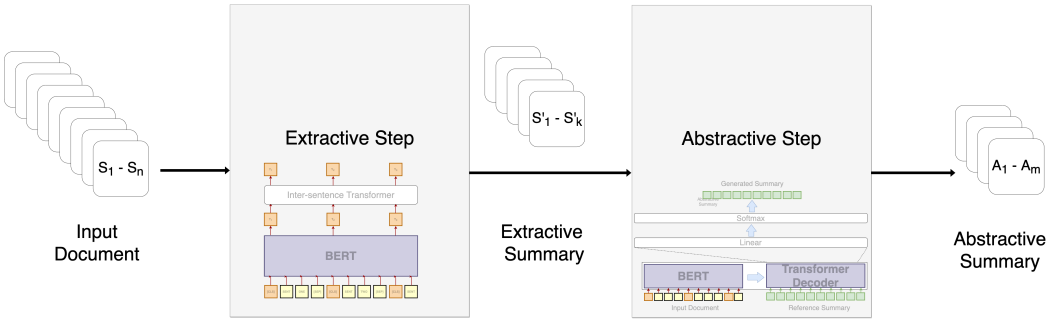


Figure 3: MSBERT Model Architecture. $S_1, S_2, ..S_n$ represents the sentences in the input document; $S'_1, S'_2, ..S'_k$ represents the extractive summary consisting of the top-k sentences selected from the document; $A_1, A_2..A_m$ represents the abstractive summary.

where $h_0 = \text{PosEmb}(T)$, T are the sentence vectors output by BERT, and PosEmb is the function of adding positional embeddings [14] to T, which indicates the position of each sentence. LN is the layer normalization operation [15], MHAtt is the multi-head attention operation [14], and l indicates the depth of the stacked layer.

The final output layer is a sigmoid classifier:

$$\hat{y}_i = \sigma(W_o h_i^L + b_o)$$

where h_i^L is the vector for the i-th sentence from the L-th layer of the Transformer. In our implementation, we use $L = 2$. The training objective of the model is minimizing the Binary Cross-Entropy loss of $\hat{y}_i$ against gold label y_i. The summarization layers are jointly fine-tuned with BERT. For fine-tuning, we use the Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We follow the learning rate schedule in [14]:

$$lrate = \tilde{lr} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

where $\tilde{lr} = 2e^{-3}$ and $warmup\_steps = 10,000$. The learning schedule increases the learning rate linearly for the first $warmup\_steps$ training steps, and then decreases it proportionally to the inverse square root of the step number.

At inference, we assign a score to each of the input sentences, which signifies their importance of being a part of the extractive summary. The sentences are then ranked by their scores from the highest to lowest, and the top $k = 8$ sentences are selected and passed on to the abstractive step. We experimented with different k values, and as discussed in detail in Section 5.2.1.

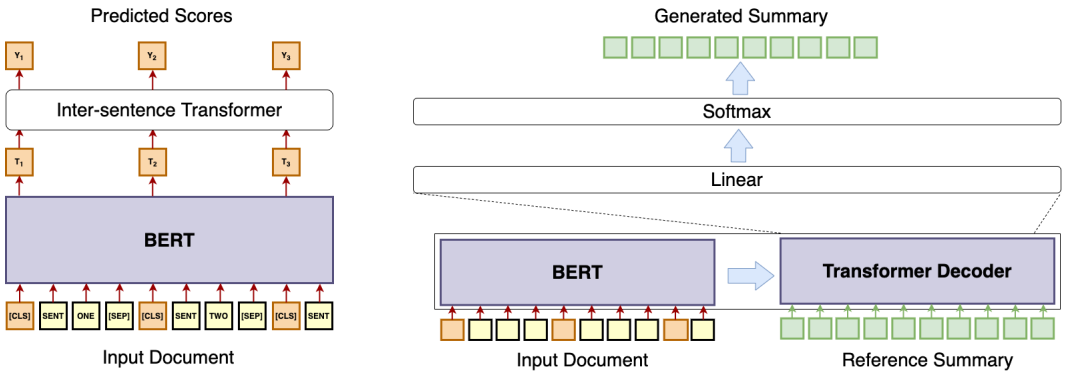
3.4 Abstractive Step

The abstractive summarization step conditions on the extracted sentences to generate a summary. It consists of an encoder-decoder framework where the encoder is pre-trained BERT, and the decoder is a 6-layered Transformer similar to [14] with random initialization (see Figure 4(b)). There exists a mismatch in the state of weights between the encoder and decoder, since the encoder is pretrained but the decoder must be trained from scratch. This may cause the encoder to overfit while the decoder is still becoming stable. To solve this problem, we follow a fine tuning schedule which separates the optimizers of the encoder and the decoder as described in [2]. The encoder is trained with a smaller learning rate and smoother decay to avoid overfitting.

Two Adam optimizers are used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ for the encoder and the decoder, each with different learning rates and warmup steps as described below:

$$lrate_E = \tilde{lr_E} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps_E^{-1.5})$$
$$lrate_D = \tilde{lr_D} \cdot min(step\_num^{-0.5}, step\_num \cdot warmup\_steps_D^{-1.5})$$

where for encoder, $\tilde{lr_E} = 2e^{-3}$ and $warmup\_steps_E = 10,000$, while for decoder, $\tilde{lr_D} = 0.1$ and $warmup\_steps_D = 5,000$



(a) Extractive step (b) Abstractive step

Figure 4: Extractive and Abstractive steps for MSBERT.

4 Experimental Setup

In this section, we present the details of our implementation and the dataset used.

4.1 Dataset

We used the CNN/DailyMail (CNN/DM) dataset [3] to evaluate our approach. The CNN/DM dataset consists of news articles and corresponding summaries, which cover the major highlight points in the story. These summaries are used as abstractive gold summaries for our purpose. To reduce the training time for the purpose of this project, we used roughly 10% of the original dataset. The statistics for the original and reduced datasets are shown in Table 1. For training, validation, and testing, we used 28009, 2001, and 2001 documents respectively. We did not anonymize entities. We split the sentences with Stanford CoreNLP [16] and pre-processed the dataset following the methods in [17]. Also, we truncated the input documents to 512 tokens.

Table 1: Statistics for the full and reduced CNN/DailyMail dataset

Dataset	# docs			avg. doc length		avg. summary length	
	train	valid	test	words	sentences	words	sentences
CNN (Full)	90,266	1,220	1093	760,50	33.98	45.70	3.59
DailyMail (Full)	196,961	12,148	10,397	653.33	29.33	54.65	3.86
CNN/DM (Reduced)	28,009	2001	2001	706.91	31.65	50.17	3.72

4.2 Implementation Details

We used the PyTorch library and OpenNMT for our implementation. In both the extractive and abstractive step, we used 'bert-base-uncased' from huggingface [18] to implement our model. We used the BERT subwords tokenizer to tokenize the input text, which helps us mitigate the out-of-vocabulary problem.

4.2.1 Extractive step

For the purpose of this work, we used the best performing trained model of BERTSumExt from [2] as the extractive step. This model is trained for 50,000 steps against oracle summaries generated for each document using a greedy algorithm similar to [10]. The algorithm generates an oracle summary which maximizes the ROUGE-2 score against the gold summary.

It is important to note that our model differs from the previous implementation in its approach of generating extractive summaries during inference. [2] uses Trigram Blocking [19] in order to minimize the similarity between the sentence being considered and the sentences already selected

as a part of the summary. On the other hand, in our model, it is essential to capture all the salient information for the subsequent abstractive step to generate an effective and complete summary. Conclusively, we do not use Trigram Blocking in the extractive step of our model, even if it means selecting sentences with redundant phrases.

4.2.2 Abstractive step

We trained the abstractive step model with 20,000 steps with gradient accumulation every 5 steps. The model was trained on 2 GPUs (Nvidia Tesla P40 with 24GB RAM) and took 5-6 hours for training due to the reduced data size and training steps. During training, we applied dropout with probability 0.1 to the output of each sub layer and employed label smoothing with a smoothing factor of 0.1. The dimensionality of input and output is 512 and the inner feed forward later has dimension 2048. We employed 8 parallel attention heads in our model. During the generation process, beam search with a beam size of 5 along with length penalty parameter $\alpha = 0.6$ was used to produce the output summary. Min length of 50 tokens and max length of 200 tokens was applied during decoding. Decoding was done until an end-of-sequence token is emitted. Also, we used Trigram Blocking [19] in the abstractive step to reduce redundancy in our generated summaries.

To adjust the hyper-parameters according to the reduced dataset size and training steps, we tried a number of combinations of warmup steps and learning rates following [20]. To deal with the problem of diverged training after a few thousand steps, we tried adding more warmup steps and decreased the learning rate, to finally arrive at the hyperparameters described in Section 3.4.

5 Evaluation

5.1 Results

We evaluate our MSBERT model only on the abstractive summarization task, as the goal of our extractive step is to eliminate the inessential information from the document and not to generate an effective extractive summary. Some example summaries generated by MSBERT are shown in Table 5. We use the widely used ROUGE metrics [21] for evaluation, which include the F-1 scores of ROUGE-1, ROUGE-2, AND ROUGE-L. These metrics report the unigram overlap, bigram overlap, and the longest common subsequence respectively between the generated summary and the abstractive gold summary.

Table 2: ROUGE F1 Evaluations for various models. The first section reports results on the full CNN/DM dataset, while the second section reports results on our reduced CNN/DM dataset.

Model	Evaluation Metrics		
	ROUGE-1	ROUGE-2	ROUGE-L
Full Dataset			
SummaRuNNer [10]	39.60	16.20	35.30
BERTSumExtAbs [2]	42.13	19.60	39.18
BART [11]	44.16	21.28	40.90
Reduced Dataset			
BERTSumExtAbs (Baseline)	37.38	15.47	34.82
MSBERT (Ours)	**38.49**	**16.39**	**35.88**

We list the ROUGE scores for SummaRuNNer [10], BERTSumExtAbs [2], and BART [11] on the full CNN/DM dataset in Table 2. These scores are taken from their respective papers and provide a perspective on the potential performance of our model on the full dataset. We compare our model MSBERT with the BERTSumExtAbs model, using our version of the reduced and preprocessed dataset. For each model, we compare the last 3 checkpoints saved during training, and choose the best performing checkpoint on the validation set. We evaluate the chosen checkpoint for each model against the test set 3 times and report the average scores. We list out the scores of BERTSumExtAbs (Baseline) and MSBERT (Our model) in Table 2. MSBERT outperforms the baseline model, BERTSumExtAbs, in all the three ROUGE metrics with an improvement of 1.11, 0.92 and 1.06 (relative improvement of 2.96%, 5.94%, and 3.04%) in ROUGE-1, ROUGE-2, AND ROUGE-L respectively.

5.2 Model Analysis

5.2.1 Number of Extracted Sentences

As discussed in Section 3.3, the top-k sentences are chosen in the extractive step and passed on to the abstractive step. This extracts the important sentences from the document that helps the abstractive step focus on only the salient information and ignore the noise. In this section, we study the impact of the selected value of k on the quality of the generated abstractive summary.

The selected value of k signifies the amount of information passed on to the abstractive step. We trained and evaluated our model with three different values of k, i.e., 4, 6, and 8. These values of k were selected after considering the average number of sentences in gold summaries of the CNN/DailyMail dataset, which is 3.72 (see Table 1). The resulting ROUGE scores for the different k values are outlined in Table 3. We observe that among the three values, $k = 8$ performs the best. We estimate that this is because just 4 or 6 sentences are unable to capture enough information for the abstractive step to generate a valid summary. We also predict that the number of sentences in the extractive summary can further be increased to improve the performance of our model, till we reach the point where the scores start declining due to too much information. We leave it to future work to estimate (or learn) the ideal number of sentences in the summary produced by the extractive step.

Table 3: ROUGE F1 Evaluations for MSBERT on the reduced CNN/DailyMail dataset against values of k, i.e., the number of sentences chosen in the extractive step and passed on to the abstractive step.

k value	Evaluation Metrics		
	ROUGE-1	ROUGE-2	ROUGE-L
4	37.24	15.15	34.24
6	37.86	15.43	35.65
8	**38.49**	**16.39**	**35.88**

5.2.2 Two stage fine-tuning

While trying to improve the performance of our approach, we employed a two stage fine-tuning approach proposed in [2], where we first fine-tune the BERT encoder on the extractive summarization task and then fine-tune it on the abstractive summarization task. We implemented this by pre-loading the weights from the BERT encoder trained for the extractive step and fine-tuning it further for the abstractive step. The scores for this model compared to our implementation are listed in Table 4. We observe that the model performs poorly as compared to BERT fine-tuned directly on the abstractive task. We attribute this result to the fact that our abstractive model is already trained to generate abstractive summaries from extractive summaries. Thus, the use of extractive objectives to train the abstractive step does not help and instead hurts the performance of our model.

Table 4: ROUGE F1 Evaluations for MSBERT on the reduced CNN/DailyMail dataset with and without the 2-stage fine-tuning approach

Model	Evaluation Metrics		
	ROUGE-1	ROUGE-2	ROUGE-L
With 2-stage fine-tuning	37.81	15.72	35.21
Without 2-stage fine-tuniung	**38.49**	**16.39**	**35.88**

6 Conclusion

In this paper, we propose a multi-step approach for training abstractive summarization models. Using our method, called Multi-Step BERT (MSBERT) Summarizer, we demonstrate that Transformer language model based frameworks can generate high quality summaries of long spans of text by using an extractive step before performing abstractive summary generation. We quantitatively measure the improvement due to our extractive step by using ROUGE metrics. We show that our model

achieves better performance than the baseline in ROUGE-1, ROUGE-2 and ROUGE-L metrics on the CNN/DM dataset.

Table 5: Example abstractive summaries generated by MSBERT on the CNN/DailyMail dataset

Input Document (Abbreiated)	**MSBERT Generated Summary**
It is the smallest computer in the world - and 150 of them can fit in a thimble. Called the Michigan Micro Mote, to tiny technology is a complete computer system. Its inventors say it can act as a smart sensor, and give everyday objects computing capabilities. The Michigan Micro Mote is the smallest computer in the world, measuring less than 2mm across. The Michigan Micro Mote contains solar cells that power the battery with ambient light ... 'To be 'complete,' a computer system must have an input of data, the ability to process that data - meaning process and store it, make decisions about what to do next – and ultimately, the ability to output the data.' Professor David Blaauw explained.	michigan micro mote is the smallest computer in the world , measuring less than 2 mm across. created by strobing light at a high frequency , the operator is able to send data to the computer. the michigan micro motes is the only computers in the global history of solar cells. it is the first complete computer system in the u.s. - and 150 of them can fit in a thimble.
New York City cab drivers spend their entire work days eavesdropping on their customers conversations, so it's no surprise that many of them have learned a thing or two about love and relationships along the way. With that in mind, New York-based writer Danielle Page set out to ask every cabbie she came across to dispense their best piece of relationship advice in hopes unlocking the key to a successful union. The drivers, many of whom are married themselves, revealed their personal tips, life lessons and cultural anecdotes all in the name of love, which Danielle then shared on YourTango. A 60-year-old named Michael revealed that his trick to marital bliss ... 'When you're only given one option, you have to fight to make it work,' he said, adding: 'People here don't do that.'	new york-based writer danielle page set out to ask every cabbie she came across to dispense their best piece of relationship advice. the drivers , many of whom are married themselves , revealed their personal tips , life lessons and cultural anecdotes all in the name of love.
Hashim Amla is to join Derbyshire next month for three NatWest T20 Blast and two LV= County Championship matches. Amla has been recruited to replace Martin Guptill after the New Zealander was called up by his country for the Test series against England. South Africa Test captain Amla, 31, will begin his short stint with Derbyshire in the Division Two championship fixture at home to Northamptonshire on May 10. Hashim Amla has signed a short-term contract to play five matches for Derbyshire, starting next month. He said: 'I'm delighted to have the opportunity to play in England this summer ... The South Africa Test captain celebrates as he reaches 150 against Ireland at the World Cup.	hashim amla has signed a short-term contract to play five matches for derbyshire. the 31-year-old will begin his short stint with derbyshire in the division two championship. he will play five games for derbyshire next month. the new zealander was invited to replace martin guptill after he was called up for the test series against england.

6.1 Future Work

In this work, we used BERTSumExtAbs as the baseline model, and built upon a part of its architecture to test our hypothesis. Recently, a number of other language model based approaches ([11], [22], [13], [12]) have shown remarkable performance and advanced the state of the art in abstractive summarization. In future work, we would like to adapt a multi-step architecture similar to MSBERT for these approaches to achieve new state of the art performance.

In this work, we select the top-k sentences in the output of the extractive step to be passed on to the abstractive step, where k is a configurable parameter. In future work, we would like to explore the possibility of formalizing k as a trainable parameter, and learning it along with the model so as to choose its optimum value and achieve the best summarization results.

Lastly, in our proposed model, the extractive and abstractive components are defined as disjoint components that are independently trained. In future work, we would also be interested in exploring the possibility of training the two components together in an end to end manner.

References

[1] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[2] Y. Liu and M. Lapata, "Text summarization with pretrained encoders," *arXiv preprint arXiv:1908.08345*, 2019.

[3] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in neural information processing systems*, pp. 1693–1701, 2015.

[4] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[5] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*, 2018.

[6] H. Zhang, Y. Gong, Y. Yan, N. Duan, J. Xu, J. Wang, M. Gong, and M. Zhou, "Pretraining-based natural language generation for text summarization," *arXiv preprint arXiv:1902.09243*, 2019.

[7] S. Gehrmann, Y. Deng, and A. M. Rush, "Bottom-up abstractive summarization," *arXiv preprint arXiv:1808.10792*, 2018.

[8] W. Li, X. Xiao, Y. Lyu, and Y. Wang, "Improving neural abstractive document summarization with explicit information selection modeling," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1787–1796, 2018.

[9] S. Subramanian, R. Li, J. Pilault, and C. Pal, "On extractive and abstractive neural document summarization with transformer language models," *arXiv preprint arXiv:1909.03186*, 2019.

[10] R. Nallapati, F. Zhai, and B. Zhou, "Summarunner: A recurrent neural network based sequence model for extractive summarization of documents," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[11] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[12] Y. Yan, W. Qi, Y. Gong, D. Liu, N. Duan, J. Chen, R. Zhang, and M. Zhou, "Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training," *arXiv preprint arXiv:2001.04063*, 2020.

[13] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "Pegasus: Pre-training with extracted gap-sentences for abstractive summarization," *arXiv preprint arXiv:1912.08777*, 2019.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[16] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60, 2014.

[17] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[18] H. Face, "Hugging face github." https://github.com/huggingface/transformers.

[19] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," *arXiv preprint arXiv:1705.04304*, 2017.

[20] M. Popel and O. Bojar, "Training tips for the transformer model," *The Prague Bulletin of Mathematical Linguistics*, vol. 110, no. 1, pp. 43–70, 2018.

[21] C.-Y. Lin, "Rouge: A packagefor automatic evaluation of summaries," in *ProceedingsofWorkshop on Text Summarization Branches Out, Post2Conference Workshop of ACL*, 2004.

[22] W. Yoon, Y. S. Yeo, M. Jeong, B.-J. Yi, and J. Kang, "Learning by semantic similarity makes abstractive summarization better," *arXiv preprint arXiv:2002.07767*, 2020.