Abstractive Text Summarization using Transfer Learning

Ekaterina Zolotareva, Tsegaye Misikir Tashu and Tomáš Horváth

ELTE- Eötvös Loránd University, Faculty of Informatics,
Department of Data Science and Engineering,
Telekom Innovation Laboratories
Pázmány Péter sétány 1/C, 1117, Budapest, Hungary
(dnbo45, tomas.horvath , misikir)@inf.elte.hu

Abstract: Recently, abstractive text summarization has achieved success in switching from linear models via sparse and handcrafted features to nonlinear neural network models via dense inputs. This success comes from the application of deep learning models on natural language processing tasks where these models are capable of modeling intricate patterns in data without handcrafted features. In this work, the text summarization problem has been explored using Sequence-to-sequence recurrent neural networks and Transfer Learning with a Unified Text-to-Text Transformer approaches. Experimental results showed that the Transfer Learning-based model achieved considerable improvement for abstractive text summarization.

1 Introduction

Summarization is closely related to data compression and information understanding both of which are key to information science and retrieval. The technology of text summarization can improve information extraction systems and also allows readers to quickly view a large number of documents for important information. Indeed, automatic summarization has been recently recognized as one of the most important natural language processing (NLP) tasks, yet one of the least solved one.

In the literature, there are two main approaches to text summarization. While extractive methods are arguably well suited for identifying the most relevant information, such techniques may lack the fluency and coherency of human-generated summaries. Abstractive text summarization is the task of generating a summary consisting of a few sentences that capture the salient ideas of the input text document. The adjective ‘abstractive’ is used to denote a summary that is not a mere selection of a few existing passages or sentences extracted from the source, but a compressed paraphrasing of the main contents of the document, potentially using vocabulary unseen in the source document [9].

Abstractive summarization has shown the most promise towards addressing issues in extracting important information from the text documents but Abstractive generation may produce sentences not seen in the original input document. Motivated by neural network success in machine translation experiments, the attention-based encoder-decoder paradigm has recently been widely studied in abstractive summarization. By dynamically accessing the relevant pieces of information based on the hidden states of the decoder during the generation of the output sequence, the model revisits the input and attends to important information.

Recent abstractive document summarization models are yet not able to achieve convincing performance. In this paper, we investigate the Transfer learning for abstractive text summarization to address a key challenge in summarization, which is to optimally compress the original document while preserving the key concepts in the original document. The rest of this paper is organized as follows: Section 2 provides an overview of the existing works and approaches. In Section 3, the approach to be investigated is introduced. Section 5 presents Experimental setting ,data sets used and results. Finally, Section 6 presents the discussion and concludes the paper and discusses prospective plans for future work.

2 Related work

The number of summarization models introduced every year has been increasing rapidly. Advancements in neural network architectures [1, 11], and the availability of largescale data enabled the transition from systems based on expert knowledge and heuristics to data-driven approaches powered by end-to-end deep neural models. Current approaches to text summarization utilize advanced attention and copying mechanisms [3, 12] multi-task and multi-reward training techniques [7], graph-based methods that involve arranging the input text in a graph and then using ranking or graph traversal algorithms in order to construct the summary [5] [13], reinforcement learning strategies [4], and hybrid extractive-abstractive models [6].

This work is based on the most recent and novel Text-To-Text Transfer Transformer (T5) [10] and on one of the main known Sequence to sequence (Seq2Seq) model [**6**]. The T5 model, pre-trained on Colossal Clean Crawled Corpus (C4), achieved state-of-the-art results on many NLP benchmarks while being flexible enough to be fine-tuned to a variety of important tasks.

3 The Transformer Model

It is possible to formulate most NLP tasks in a “text-to-text” format – that is, a task where the model is fed some text for context or conditioning and is then asked to produce some output text. This approach provides a consistent training objective both for pre-training and fine-tuning. Specifically,the model is trained with a maximum likelihood objective regardless of the task.

3.1 The Transformer: Model Architecture

Most competitive and successful neural sequence transduction models have an encoder-decoder structure [14, 11]. Here, the encoder maps an input sequence of symbol representations $(x_1,...,x_n)$ to a sequence of continuous representations $z = (z_1,...,z_n)$ [14]. Given z, the decoder then generates an output sequence $(y_1,...,y_m)$ of symbols one element at a time. At each step, the model is automatically regressive, with the previously generated symbols being consumed as additional input when generating the next step. The Transformer [14] follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 1, respectively (See [14] for more).

Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has a multi-head self-attention mechanism, and a simple, position-wise fully connected feed-forward network. A residual connection is employed around each of the two sub-layers followed by layer normalization. That is, the output of each sub-layer is $LayerNorm(x + Sublayer(x))$ Where $Sublayer(x)$ is the function implemented by the sub-layer itself [14].

Decoder: The decoder also consists of a stack of $N = 6$ identical layers. The decoder inserts a third sub-layer which, in addition to the two sub-layers, provides multi-head attention to the output of the encoder stack. Similar to the encoder, a residual connection around each of the two sub-layers is used, followed by a layer normalization. To prevent positions from paying attention to subsequent positions, a modified self-attention sub-layer is used in the decoder [14].

Attention: An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, the keys, the values and the output are all vectors [14]. The output can be calculated as a weighted sum of the values, where the weight assigned to each value is calculated by a compatibility function of the query with the corresponding key.

The advantage of using multi-head attention allows the model to share information from different representation subspaces at different positions. With a single attention head this is prevented by averaging [14]. The Transformer uses multi-head attention in the following manner:

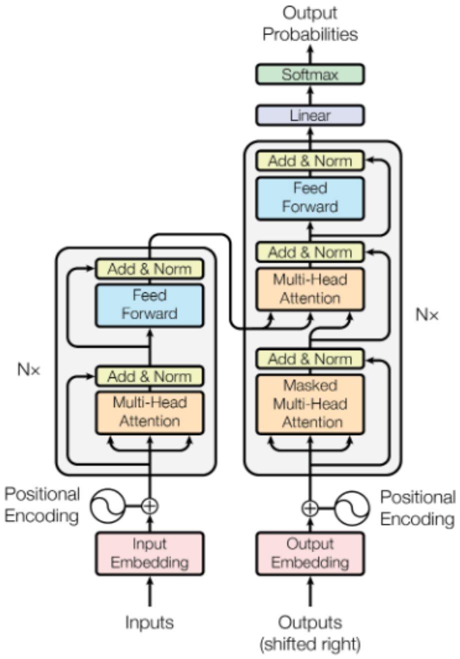

Figure 1: The Transformer - Model Architecture [14]

- In “encoder-decoder attention” layers, the queries come from the previous decoder layer and the memory keys and values come from the output of the encoder. This allows every position in the decoder to attend over all positions in the input sequence [15, 2].
- The encoder contains self-attention layers. In a self-attention layer, all keys, values and queries come from the same location, in this case from the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder [14].
- Similarly, self-attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including that position [14].

3.2 T5 approach

Attention Masks: A major distinguishing factor for different architectures is the “mask” used by different attention mechanisms in the model. Recall that the self-attention operation in a Transformer takes a sequence as input and outputs a new sequence of the same length [10]. Each entry of the output sequence is produced by computing a weighted average of entries of the input sequence. Specifically, let y_i refer to the i^{th} element of the output sequence and x_j refer to the j^{th} entry of the input sequence.

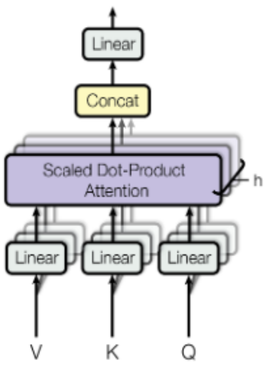

Figure 2: Multi-Head Attention [14]

In practice, we compute the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V. We compute the matrix of outputs as:

$$y_i = \sum_j w_{i,j} x_j \tag{1}$$

Where $w_{i,j}$ is the scalar weight produced by the self-attention mechanism as a function of x_i and x_j. The attention mask is then used to zero out certain weights in order to constrain which entries of the input can be attended to at a given output time step.

Encoder-Decoder: An encoder-decoder Transformer consists of two layers of stacks: the encoder, which is fed an input sequence, and the decoder, which generates a new output sequence. The encoder uses a “fully visible” attention mask. The “fully visible” masking allows a self-attention mechanism to pay attention to each input of its output. This form of masking is suitable when the attention is over a “prefix”, i.e. a context that is provided to the model that will later be used to make predictions. The self-attention operations in the decoder of the transformer use a “causal” masking pattern. Within model training process, approaching with "causal" mask let decoder prevent the model from attending to the j^{th} entry during handling i^{th} input sequence for $j > i$. This is used during training so that the model cannot “see into the future” while producing its output.

4 Sequence to Sequence Model

The Recurrent Neural Network(RNN) is a natural generalization of feed forward neural networks to sequences. Given a sequence of inputs$(x_1, ..., x_T)$, a standard RNN computes a sequence of outputs $(y_1, ..., y_T)$ by iterating the equation 2 and 3:

$$h_t = sigmoid(W^{hx} x_t + W^{hh} h_{t-1}) \tag{2}$$

$$y_t = W^{yh} h_t \tag{3}$$

The RNN can easily map sequences to sequences whenever the alignment between the inputs and the outputs is known ahead of time. However, it is not clear how to apply an RNN to problems whose input and the output sequences have different lengths with complicated and non-monotonic relationships.

Sequence learning consists of mapping the input sequence with one RNN to a vector of fixed size and then mapping the vector with another RNN to the target sequence. Although it could work in principle, since the RNN is supplied with all relevant information, it would be difficult to train the RNNs due to the resulting long-term dependencies. However, the Long Short-Term Memory (LSTM) is known to learn problems with long-range time dependencies, so an LSTM can be successful in this setting.

The objective of the LSTM is to estimate the conditional probability $p(y_1, ..., y_{M'} | x_1, ..., x_M)$ where $(x_1, ..., x_M)$ is an input sequence and $(y_1, ..., y_{M'})$ is its corresponding output sequence whose length M' may differ from M. The LSTM computes the conditional probability by first obtaining the fixed-dimensional representation v of the input sequence $(x_1, ..., x_M)$ given by the last hidden state of the LSTM, and then computing the probability of $(y_1, ..., y_{M'})$ with a standard LSTM language model formulation whose initial hidden state is set to the representation v of $(x_1, ..., x_T)$:

$$p(y_1, ..., y_{M'} | x_1, ..., x_M) = \prod_{m=1}^{M'} p(y_m | v, y_1, ..., y_{m-1}) \tag{4}$$

In this equation, each $p(y_m | v, y_1, ..., y_{m-1})$ distribution is represented with a soft max over all the words in the vocabulary. The LSTM formulation from Graves has been used. It is require that each sentence ends with a special end-of-sentence symbol “<EOS>”, which enables the model to define a distribution over sequences of all possible lengths.

5 Experimental Setting and Results

5.1 Dataset Selection

The experiment was carried out on the BBC News dataset provided by Kaggle[1]. The dataset consists of 2225 documents from the BBC news website corresponding to stories in five topical areas from 2004 to 2005 and includes five class labels which are business, entertainment, politics, sport, technology.

[1] https://www.kaggle.com/pariza/bbc-news-summary

5.2 Data Preprocessing

In preprocessing the documnets, the following tasks were performed: tokenization using the NLTK[2] tokenizer; removing punctuation marks, determiners, and prepositions; a transformation to lower-case; stopword removal and word stemming. In the stop word removal step, the words that are in the english stop word list were removed. After removing the stopwords, the words have been stemmed to their roots.

Python was used to implement the proposed LSH-based AEE algorithm. The Scikit-learn[3] , gensim [4]and the Numpy[5] and PyTorch[6]libraries were used.

5.3 T5 Model Hyper-Parameter Setting

The following parameters were selected by taking into account the computation power and resources at hand. Therefore, We selected the Hyper parameters using the manual configuration method. The dataset is split into 80% training data and 20% testing data with `sample` function from `pandas` framework.

- `TRAIN_BATCH_SIZE` = 2 (default: 64)
- `VALID_BATCH_SIZE` = 2 (default: 1000)
- `TRAIN_EPOCHS` = 2 (default: 10)
- `VAL_EPOCHS` = 1 (default: 10)
- `LEARNING_RATE` = $1^e - 4$ (default: 0.01)
- `SEED` = 42 (default: 42)

Initiating Fine-Tuning for the model on BBC News dataset:

- Epoch: 0, Loss: 14.0325
- Epoch: 0, Loss: 2.9507
- Epoch: 1, Loss: 2.8506
- Epoch: 1, Loss: 2.0221

5.4 Seq2Seq Model Settings

Abstractive summarization neural network model is built using TensorFlow and Keras machine learning and neural networks python libraries.

First, set up the maximum cleaned text and summary lengths based on the distribution of sequence lengths from the chosen sample. Add “sostok” – START and “eostok” – END tokens to the reference summary as this will help the model to determine when the sequence starts and ends respectively. The dataset is split into 80% for training data and 20% for testing data with `train_test_split` package from `sklearn.model_selection`.

Then, both the training and testing data are tokenized to form the vocabulary and converted the word sequences into equal length integer sequences by using Tokenizer and pad sequences modules from `keras.preprocessing` package.

Our Seq2Seq model has three LSTM layers for the encoder network and a single LSTM layer for the decoder network with an embedding layer on both the encoder and decoder network. The custom attention layer was also used to remember the lengthy sequences, and the output layer uses the SoftMax activation function. The hidden layers have a dimension of 256 units and the embedding layers have a size of 200 units. Besides, a drop-out value of 0.4 is used in each hidden layer to reduce model overfitting and improve performance. These layers have been implemented and the model is built using different wrappers like Input, LSTM, Embedding, Dense from the `tensorflow.keras.layers`.

Different values for each hyper-parameters was used and the following hyper-parameters setting were selected during training based on the their performance :

- Epochs = 25
- Optimizer = “rmsprop”
- Batch size = 64
- Latent dimension = 256
- Embedding dimension = 200
- Loss function = “sparse_categorical_crossentropy”

Hyper parameters were selected using the manual configuration method. In the accuracy and loss values are determined and analyzed. After training phase comes the inference phase, in which we input the testing data to our model and get the output predicted summary.

5.5 Evaluation Metrics

In Text Summarization, summary evaluation is an essential chore. Manual and semi-automatic evaluation of large-scale summarization models is costly and cumbersome. Much effort has been made to develop automatic metrics that would allow for fast and cheap evaluation of models. The ROUGE package introduced by Lin [8] offers a set of automatic metrics based on the lexical overlap between candidate and reference summaries .

We used ROUGE metrics for our evaluation process. ROUGE refers to Recall Oriented Understudy for Gisting Evaluation which is an automatic summary evaluation

[2] http://www.nltk.org

[3] `http://scikit-learn.org/`

[4] `https://radimrehurek.com/gensim/`

[5] `http://www.numpy.org/`

[6] `https://www.pytorch.org/`

	ROUGE-1	ROUGE-2	ROUGE-L
F1	0.473	0.265	0.361
Precision	0.467	0.261	0.338
Recall	0.480	0.269	0.389

Table 1: Results on the BBC test set using T5 Model

	ROUGE-1	ROUGE-2	ROUGE-L
F1	0.313	0.193	0.262
Precision	0.388	0.275	0.289
Recall	0.324	0.132	0.199

Table 2: Results on the BBC test set using Seq2Seq Model

Results	
Generated Text	Actual text
Veteran Labour MP and former Cabinet minister Jack Cunningham has said he will stand down at the next election Mr Blair said He was an...	Labour s Cunningham to stand down Veteran Labour MP and former Cabinet minister Jack Cunningham has said he will stand down...
Ministers would not rule out scrapping the Child Support Agency if it failed to improve Work and Pension Secretary...	CSA could close says minister Ministers would not rule out scrapping the Child Support...

Table 3: Sample results using T5 model

bench-marking metric that is widely used by researchers to determine the quality of the summary produced by comparing the machine generated summary with the reference summary (ideal or human written ones). ROUGE scores are computed from the number of overlapping words between the reference summary and machine generated summary. There are different types of ROUGE such as ROUGE-N, ROUGE-L,ROUGE-S and ROUGE-W. But the most commonly used ones are ROUGE-N (ROUGE-1,ROUGE-2) and ROUGE-L and hence we also use the same.

ROUGE-N : It denotes the overlapping of n-grams between the system generated summary and the ideal reference summary. For instance, unigram (ROUGE-1), bigram (ROUGE-2), trigram (ROUGE-3) and so on. The ROUGE-n is given by:

$$ROUGE-n = \frac{\sum\limits_{S\in RS}\sum\limits_{gram_n\in S} Count_{match}(gram_n)}{\sum\limits_{S\in RS}\sum\limits_{gram_n\in S} Count(gram_n)} \tag{5}$$

Where RS is a set of reference summaries, n stands for the length of the n-gram, $gram_n$, and $Countmatch(gram_n)$ is the maximum number of n-grams co-occurring in a generated summary and a set of reference summaries.

ROUGE-L: It denotes the Longest Common Subsequence (LCS) matching between the reference summary and system generated summary.

5.6 Results

The experimental results of Text-To-Text Transfer Transformer (T5) method were compared with attention based Sequence to sequence based methods. The experimental results are presented in Table 1 and Table 2. The Results shown in Table 1 are from Transformer (T5) method and the results in table 2 are the baseline method. According to the experimental results presented, Text-To-Text Transfer Transformer (T5) based abstractive text summarization outperformed the baseline attention based seq2seq approach in all of the matrices used. Sample prediction results from the test are presented in Table 3

6 Conclusion

In this paper, we have dealt with the demanding task of abstractive document summarization. We used a newly introduced approach [10], the Transformer or T5 framework, to create a multi-sentence summary. Experiments were carried out to verify the effectiveness of the proposed method. Experimental results on the BBC News dataset showed that the T5 model performed well in the abstractive document summarization. The future direction is to study the Transformer method for the task of summarizing multiple documents and also to very the T5 approach on other benchmark dataset.

Acknowledgment

The research has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013, Thematic Fundamental Research Collaborations Grounding Innovation in Informatics and Infocommunications).

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv preprint arXiv:1409.0473* (2014).

[3] Arman Cohan et al. "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 615–621. DOI: 10.18653/v1/N18-2097. URL: https://www.aclweb.org/anthology/N18-2097.

[4] Yue Dong et al. "BanditSum: Extractive Summarization as a Contextual Bandit". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 3739–3748. DOI: 10.18653/v1/d18-1409. URL: https://doi.org/10.18653/v1/d18-1409.

[5] Günes Erkan and Dragomir R Radev. "Lexrank: Graph-based lexical centrality as salience in text summarization". In: *Journal of artificial intelligence research* 22 (2004), pp. 457–479.

[6] Sebastian Gehrmann, Yuntian Deng, and Alexander M. Rush. "Bottom-Up Abstractive Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 4098–4109. DOI: 10.18653/v1/d18-1443.

[7] Wojciech Kryscinski et al. "Improving Abstraction in Text Summarization". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*. Ed. by Ellen Riloff et al. Association for Computational Linguistics, 2018, pp. 1808–1817. DOI: 10.18653/v1/d18-1207.

[8] Chin-Yew Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://www.aclweb.org/anthology/W04-1013.

[9] Ramesh Nallapati et al. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond". In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 280–290. DOI: 10.18653/v1/K16-1028. URL: https://www.aclweb.org/anthology/K16-1028.

[10] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer". In: *arXiv preprint arXiv:1910.10683* (2019).

[11] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks". In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani et al. Curran Associates, Inc., 2014, pp. 3104–3112.

[12] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. "Abstractive Document Summarization with a Graph-Based Attentional Neural Model". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1171–1181. DOI: 10.18653/v1/P17-1108.

[13] H. Van Lierde and Tommy W.S. Chow. "Query-oriented text summarization based on hypergraph transversals". In: *Information Processing Management* 56.4 (2019), pp. 1317–1338. ISSN: 0306-4573. DOI: https://doi.org/10.1016/j.ipm.2019.03.003.

[14] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon et al. Curran Associates, Inc., 2017, pp. 5998–6008.

[15] Yonghui Wu et al. "Google's neural machine translation system: Bridging the gap between human and machine translation". In: *arXiv preprint arXiv:1609.08144* (2016).