Neural Text Summarization
Machine Learning for Natural Language Processing 2021

Ryan Boustany
ENSAE Paris
ryan.boustany@ensae.fr

Emma Sarfati
ENSAE Paris
emma.sarfati@ensae.fr

Abstract

Text summarization is a famously hard natural language processing task. Due to its difficulty in nature but also the non-natural way to evaluate a summarizer's performance, it is currently a hot topic of research and the field is prone to gain improvements. In this paper, we elaborate an abstractive summarization model using a state-of-the-art Transformer that we adapted to our dataset and problematic, and compare it with an end-to-end designed extractive summarizer.

1 Problem Framing

Our objective is to generate automatically the headlines from a relatively short summary of a newspaper article. That is, from a set of words vectors, we want to generate a shorter set of such vectors which is able to capture the essence of the original one. A typical way to treat this NLP problem is to use an Encoder-Decoder architecture (see figure 2), which allows us to predict sequentially the conditional probability of a word being at the t^{th} position of a sentence, *i.e* $\mathbb{P}(y_t|y_{t-1}, h_{t-1})$, where we consider y_t as the target words and h_t the hidden state of the RNN. In short, in our problem, input data is a sequence $(x_1, ..., x_{T_1})$ and output is a sequence $(y_1, ..., y_{T_2})$, with $T_2 < T_1$.

2 Experiments Protocol

2.1 Dataset

For our summarization purpose, we worked on the news summary dataset available on Kaggle[1] which gathers news articles from Hindu, Indian Times and The Guardian. Time period ranges from February to August 2017. The dataset consists of 4515 examples and contains mainly the following features: complete article, short text, and text headlines. In our experiments, we decided to focus on the headlines predictions from the short text data, mostly for computational reason (complete article was in average $3\times$ larger than short text).

2.2 Models

The approaches to summarize an input vector are twofolds; **extractive** summarization is the strategy of concatenating extracts taken from a corpus into a summary, while **abstractive** summarization involves paraphrasing the corpus using novel sentences. From a practical point of view, implementing an abstractive summarization technique would imply to use a whole language dictionnary, which can cause bad results when we deal with a large amount of words, but it could generate more human-like sentence. On the other hand, extractive summarization could help generating straightforward summaries but with lack of idiomatic nuances.

***Abstractive* summarizer.** The difficulty of the tasked enhanced motivation to search for state-of-the-art structure that could allow us to retrieve the best features of our original text data. We decided to work with the T5 model, proposed by the Google NLP team in 2020 (Raffel et al., 2020). T5 stands for **T**ext-**t**o-**T**ext **T**ransfer **T**ransformer and is a slightly-modified version of the original Transformer model proposed in 2017 (Vaswani et al., 2017).

The original transformer model consists in an encoder-decoder as presented in 4 and is mainly best on feed-forward neural networds, basic attention and self-attention layers ((Bahdanau et al., 2014), (Vaswani et al., 2017)). The model from Raffel et al. is roughly equivalent to the orig-

[1] https://www.kaggle.com/sunnysai12345/news-summary

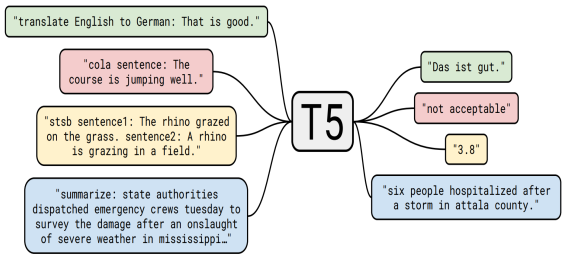

Figure 1: The T5 inputs and outputs types.

inal Transformer in nature[2]. The main contribution however is that the model has been pre-trained taking into account the type of task it has to achieve with the given input, by **expressively encoding the task into a vector**, e.g., for translation: `translate English to German:...`. We fine-tuned this model to use it for our headlines generation task. To do so, we carried the necessary pre-processing to use the T5 model[3] on the News Summary Dataset. We then indicated to T5 that we wanted a sequence generation by passing the `ConditionalGeneration` call to the model.

***Extractive* summarizer.** For comparison, we led an extractive summarization model, that time based on more classical neural structures that we designed ourselves. We extracted more rows of the original dataset, based on a "bigger" version of the news summary dataset that was also available on Kaggle. We built a vocabulary based on the words present in the ≈100,000 rows in this dataset, which led to a vocabulary size of 75,322. We built an encoder-decoder architecture, based on a sequence-to-sequence model, as follows:

- Encoder: word embedding of dimension 128, with 2 layers of LSTM of each size 256 and a dropout probability of 0.5 (Srivastava et al., 2014).
- Decoder: same structure, with an original vocabulary of size 32,096 corresponding to the headlines vocabulary.

3 Results

Evaluate a text summarization model is generally an arduous task in computational linguistics. Some work have been made in order to find a relevant metric, for instance the BLEU score (bilingual evaluation understudy, (Papineni et al., 2002)) relies on an adaptation of the usual precision score for machine translation and text summarization. In this work, we evaluate our model with ROUGE score (Lin, 2004), which is an adapted and interpretable metric for our problem. The ROUGE score proposes to count the number of concordant N-grams between a reference summary and a candidate summary. Denoting S the reference summary, C the candidate (or predicted) one, N the length of the N-gram and $\#_{\text{match}}\,\text{gram}_N(S,C)$ is the maximum number of N-grams co-occurring in S and C.

$$\text{Rouge-N}(S,C) = \frac{\sum_{\text{gram}_N \in S} \#_{\text{match}}\,\text{gram}_N(S,C)}{\sum_{\text{gram}_N \in S} \#\,\text{gram}_N(S)}$$

ROUGE-L measures longest matching sequence of words using Longest Common Subsequence. To evaluate the model on a full validation set, all the ROUGE scores between each summary and its target are averaged. After splitting our training and validation data in a classical 75/25% share, we evaluated our summarizer on the validation set. We tested our models in 2 for the abstractive one and 3 for the extractive one. Note that for the extractive model, the examples are generally worse than the one presented in this paper (see our notebook).

Rouge-1	0.566
Rouge-2	0.297
Rouge-L	0.552

Table 1: ROUGE scores on our validation data with T5 model.

4 Discussion/Conclusion

We built two deep-learning based models, with each relying on well-known approaches for text summarization: extractive and abstractive summarization. Our extractive summarizer, built from scratch, led to poor results in terms of summarization coherence. Our T5-based model leads to impressive results in terms of grammatical coherence and language understanding.

As a baseline, we could have tried to test classical machine learning models before diving into deep-learning ones by considering it a classification task on the whole vocabulary at each timestep.

[2]Changes: removing the Layer Norm bias, placing the layer normalization outside the residual path, and using a different position embedding scheme.

[3]see https://huggingface.co/transformers/model_doc/t5.html for documentation.

Text type	Text
Original short text	In a first, the National Security Guard (NSG) commandos participated in the Republic Day parade. While a contingent of some 60 commandos in black overalls and full-armed gear marched at the Rajpath, several commandos were on seven vehicles with hydraulic ladders mounted on them. Further, NSG's anti-hijacking van 'Sherpa', a bullet-proof armoured vehicle, also made its debut in the parade.
Target headline	NSG commandos debut at Republic Day parade
Predicted headline	NSG commandos participate in Republic Day parade for 1st time

Table 2: Example of generated summary with T5 Transformer on our News Summary Dataset.

Text type	Text
Original short text	the islamic state militant group has claimed responsibility for the terror attack in new york <unk> us in its weekly newsletter . the group called the attacker <unk> sayfullo saipov <unk> a <unk> soldier of the caliphate <unk> <unk> however it provided no evidence to support its claim . the attack killed <unk> people and injured several others .
Target headline	isis claims responsibility for new york terror attack
Predicted headline	isis claims responsibility for isis attack in syria

Table 3: Example of generated summary with the extractive summarizer on our News Summary Dataset. The encoding of the sentences is different from the abstractive one, hence the <unk> symbols, lack of capital letters and spaces between dots.

References

Kishore Papineni, Salim Roukos, Todd Ward, and Wei Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Appendix

A Models

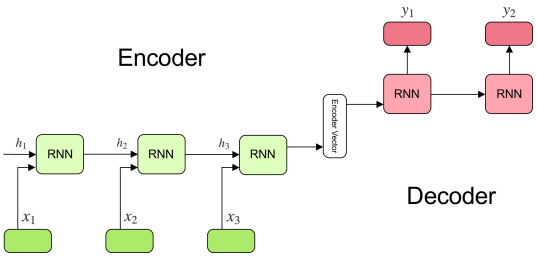

Figure 2: The Encoder-Decoder architecture.

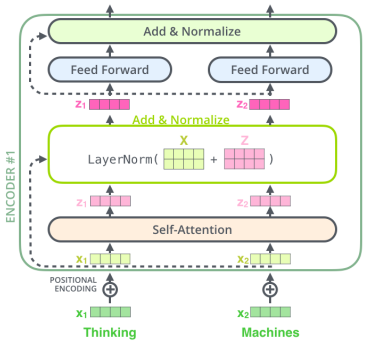

Figure 3: The Encoder of the Transformer.

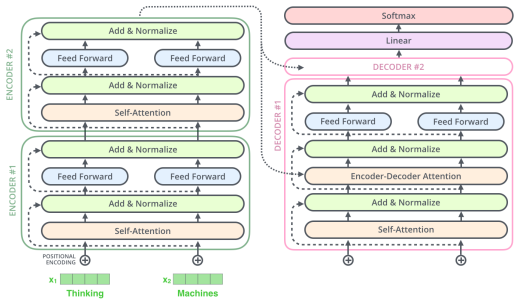

Figure 4: The full Transformer architecture.

B Data Analysis

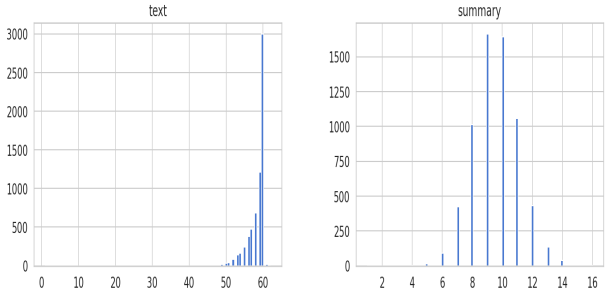

Figure 5: Full text vs. headlines sequence lengths repartitions.

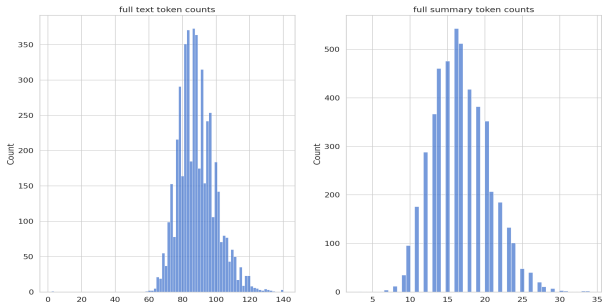

Figure 6: Full text vs. headlines token lengths repartitions. We use the T5 tokenizer.