Natural Language Processing

M2 - Data Science

Get to the Point: Summarization with Pointer-Generator Networks (ACL 2017)

Author:
Alexandre Zajac

Professors:
Chloe Clavel
Matthieu Labeau

March 25, 2021

Contents

Introduction to the main contributions

The task of text summarization

The purpose of this part is to highlight what are the main contributions of the paper to the field of text summarization. More precisely, the task tackled in the paper is the one of **abstractive summarization** which is the branch of text summarization that is the hardest to solve because it is much easier to select text from the source content (also called extractive summarization) than it is to generate text from scratch.

A simple yet powerful way of differentiating abstractive from extractive summarization is with a pen and highlighter: while abstractive summarization can be seen as writing new sentences with a pen, extractive corresponds to simply highlighting important sentences from the source text. While extractive summarization had a lot of contributed papers at the time of publication of this paper, abstractive summarization was still difficult to approach, so the authors decided to contribute on this subject. The paper presented is dating from 2017, but is still based on an architecture that is commonly used for neural languages tasks: the sequence-to-sequence models with attention [4] and [2].

Extending the Seq2Seq architecture

At the time when the paper was published, recurrent neural networks, which are designed to perform operations on sequential data, were the state of the art for text summarization. In particular, the sequence-to-sequence based models were reaching the best ROUGE(s) scores on various datasets. Here is a detailed overview of this base network architecture, for a summarization task:

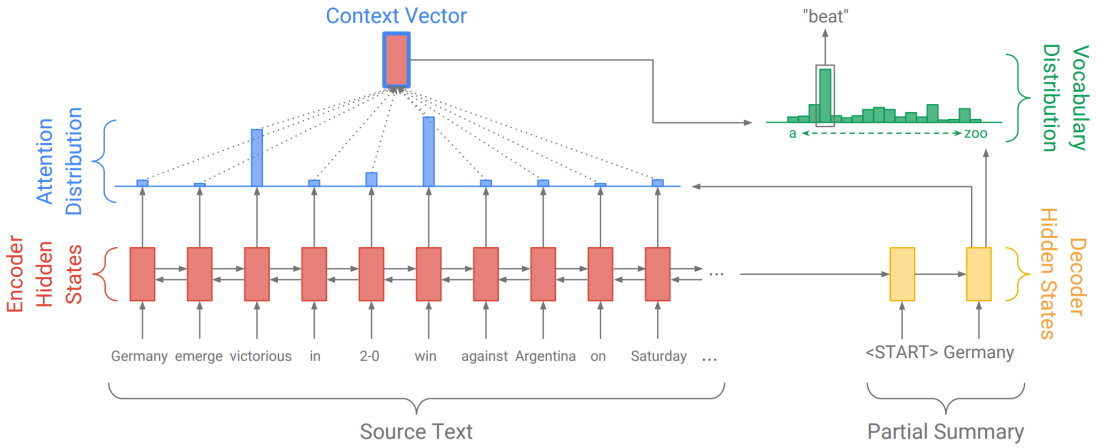

Figure 1: Sequence to Sequence with Attention architecture for summarization

Since this a well known architecture in the field of text summarization in general, it won't be extensively covered here, but only up to a point for understanding the contributions of the main paper. The *encoder* part of the network (here in red) is a bi-directional LSTM, producing hidden states for each word and timestamp. The *decoder* then generates a summary, starting by a START token, and taking its output at state $t-1$ for generating a word at time t. The decoder hidden states are used to calculate what we call the attention distribution, representing where the network is looking in the input text for generating the word at time t. This distribution is used for generating the context vector, a weighted mean of the decoder hidden states, and is finally combined to pick the most likely word to take from the **initial vocabulary** as candidate for summarization.

However, the authors identified two main issues with the summarization results of this architecture:

- The details of the summaries are sometimes not really accurate in terms of the facts they expose, and this is happening quite a lot for words that are **out of the vocabulary** with which the model was trained.
- The summaries tend to **repeat themselves** even when their length is not that large.

This is why the authors introduced/reused two novel systems by extending the base Seq2Seq architecture to address the above-stated problems. The first mechanism is the use of **pointer-generator networks**, extending the pointer networks [6], which are hybrid in the sense that the network can choose to copy words from the source content via pointing, while retaining the ability to generate words from the fixed vocabulary on which it was trained. This helped tackle the issue concerning out of vocabulary issues.

The other improvement added to the network is the introduction of the **coverage mechanism**, which is utilizing the attention distribution to keep track of what's been covered by the network in the source content so far, and penalizes the network for attending to same parts of the input again. This part was a big improvement step for avoiding repeated sequences. Let's analyze it in the further in the following part.

Advantages and issues of the approach

Pointer-Generator network

This first contribution addresses the issue of difficult replication of rare words present in the original source text. Indeed, the embedding for a rare word is unlikely to be of good quality, which ends up clustering this word around not so related or irrelevant other words. So the task of simply copying words from the source text is not trivial with the initial network. Let's now see the pointer-generator addition in the following figure:

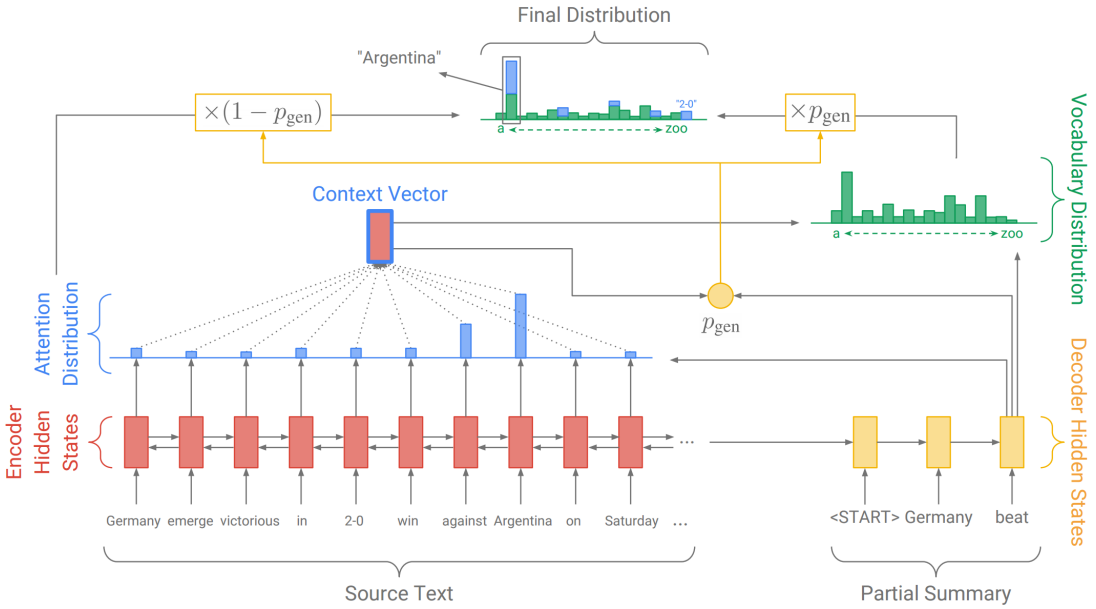

Figure 2: Addition of the Pointer Generator mechanism to the Sequence to Sequence architecture for summarization

As before we perform all the previous attention calculations, but we add a generation probability p_{gen}. It is the probability of generating a word from the vocabulary and is used to weight and combine the vocabulary distribution P_{vocab} (used for word generation or sampling from the vocabulary), and the attention distribution a (used for pointing to source words w_i) into the final distribution P_{final} with the following equation:

$$P_{final}(w) = p_{gen} * P_{vocab}(w) + (1 - p_{gen}) * \sum_{w_i} a_i \tag{1}$$

This equation replaces the sampling from the base vocabulary with the highest probability. One of the core strength of this technique is the ability to copy words that are out of vocabulary, since they can be "pointed" from the source text: it only depends on the attention weights placed on this word and the p_{gen} value. Moreover, the fact than copying from the base text is now almost easy, we can train the network with a smaller vocabulary/embedding size, and it makes the overall loop of developing, maintaining and monitoring the model's performance easier. It results in improved training time to achieve the same if not more accurate results than the base Seq2Seq model.

These advantages come up with few issues though. The first one is that directly being able to copy from the source text can still be seen as abstractive summarization if it's correctly blended with the generation of words from the vocabulary, but a fine detail has to be put on the value of p_{gen} and the values of the attention distribution, so that the method doesn't converge to being only extractive. It's exactly what the researchers noticed at testing time, where they got a value of p_{gen} of 0.17, meaning the network was 6 times more tented to only copy words with high attention weights, while this value was different at train time (0.57). A special "attention" has to be put on the parameters for the pointer-generator network, and this issue is related to the second mechanism.

The coverage mechanism

As explained above, the second problem with baseline Seq2Seq is the repetition of the words generated, and it is still an issue with the addition of the pointer-generator network. It might be related to the fact that if the decoder starts generating not sensible words at the beginning of the sentence, it will bias its confidence and most likely interfere with the relationships between the words, and it is also an aspect that will be addressed in the following papers, related to the difficulty of traditional RNN-based models to model long-term relationships.
To overcome this, the coverage mechanism is simply introducing a memory, storage, record of the attention calculated for each of the words in the input sequence. By doing so, it's modifying the loss function and penalizing the network for attending to same parts of the input again. If we denote c^t the coverage vector, and a^t the attention at time t, we have:

$$c^t = \sum_{t'=0}^{t-1} a^{t'} \tag{2}$$

Using this coverage in the attention mechanism, the original loss being optimized by the network is modified, with the addition of what is named the covloss:

$$covloss_t = \sum_i min(a_i^t, c_i^t) \tag{3}$$

This term is then regularized using a λ parameter, producing a new composite final loss function for the network:

$$\mathcal{L}_{\sqcup} = -logP(w_t) + \lambda * \sum_i min(a_i^t, c_i^t) \tag{4}$$

This addition of the coverage mechanism yields a significant boost both in the ROUGE score and the interpretability of the model. The fact that words are prevented from being attended too much has a

significant advantage: the most obvious is that words/sequence tend to not be repeated frequently, which helps both the base network architecture and the pointer-generator addition.

Relevance, related works and further contributions

Observations and notes on the approach

The paper and method presented outperformed the state of the art implementations at the time of publication, but a few remarks were raised to interest. The first is that even if was a big step towards abstractive summarization, there is a large room for improvement. The model tends to adopt a syntax style that is **similar to the one it was trained on**, and it might be because the level of abstraction of the network is too tight to model complex relationships. It is related with the idea that sometimes the **attention mechanism is too "local"**, resulting in parts of the text that are not much relevant to the summary being attended and summarized on. The necessity for a longer range global "attention" was a possibility here, to put more attention to general context and correctly choose the best fragments of the source text to generate summaries.

Another part in which the network is not that efficient is related to the generative aspects of the summary. Since this is not a generative model and the "generation" of words for summaries is a probabilistic sampling on the base vocabulary, the network is sometimes not good at **composing well structured sentences**, can switch the logic for important parts of the summary or fail to make a sentence understandable. This might again partly be due to the RNN-based architecture, that is not able to catch longer dependencies in the source text, and this is precisely what future research has improved upon.

Comparison with posterior research papers

As stated above, the approach to use a modified version of Seq2Seq with attention showed really promising results in 2017, but it's only a few months/years later than a disruptive architecture emerged from the literature: **the transformers**. Introduced only a few months after this pointer-generator with coverage paper, the paper [5] introduced this architecture as an enhancement, not so much as a replacement, of the sequential RNN-LSTM-GRU networks. This mechanism doesn't uses the sequential component of natural language the same way that RNN do, because the transformer is not a sequential network, and uses a system based on trigonometry to encode positional characteristics of the input sequence, and this allows a **parallelization of the computations across the time dimension**. There are a lot more concepts introduced in this paper and it would impossible to fit them all here, so let's see a summarized figure of the model:

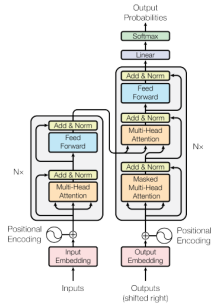

Figure 3: The transformer model architecture

The capacity of the architecture to model much longer dependencies through time, given that the vanishing gradient problem present in sequential networks wasn't present in this network form, is one component that improved the SOTA scores on text summarization. Even if the paper didn't explicitly quoted a ROUGE score for summarization, the transformer architecture alone gave path to other variations that did exceed the score for pointer-generator networks. The below-mentioned scores were reported from the Papers with code website, for the task of abstractive Text Summarization on CNN / Daily Mail dataset. The paper presented here is now at rank 28 on the benchmark, and the best networks are all from 2020:

- The 3rd position is for **ProphetNet** by Microsoft [8], introducing a n-step ahead prediction on a sequence-to-sequence architecture, encouraging the model to plan for the future words and prevent overfitting on local structure. It is also based on self-attention, as for the first transformer architecture.
- The 2nd position is for **ERNIE-GEN** [7], which expose an interesting framework on the pre-training of models instead of introducing a novel architecture. To make summary generation closer to human writing patterns, this paper introduces a span-by-span generation flow that trains the model to predict semantically complete batch of words consecutively rather than predicting word by word.
- The first place is holded by **BART+R3F** [1], also introducing a framework for better fine tuning language models more efficiently.

The common point to see here is that all the first models' results are based on transformer architectures, meaning that it has become the go-to architecture for abstractive summarization. The best models outperform this paper by about 5 ROUGE1 points (44.38 vs 39.53), which is a lot, but this metric used for evaluating summaries still remain not intuitive and tough to represent clearly for humans.
One last note about the best model based on the pointer-generator mechanism for abstractive summarization, presented in [3], is achieving 41.72 on ROUGE1. It uses a general plug-and-play module (called SELECTOR) that wraps around and guides an existing encoder-decoder model. It uses a diversficiation stage, with a mixture of "experts" to sample different binary masks on the source sequence for diverse content selection. The generation stage uses a standard encoder-decoder model given each selected content from the source sequence. The paper dates from 2019, and shows that even this type of networks are able to **beat transformer based architectures** with novel approaches for text generation.

References

[1] Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. arXiv preprint arXiv:2008.03156, 2020.

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

[3] Jaemin Cho, Minjoon Seo, and Hannaneh Hajishirzi. Mixture content selection for diverse sequence generation. arXiv preprint arXiv:1909.01953, 2019.

[4] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. arXiv preprint arXiv:1409.3215, 2014.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. arXiv preprint arXiv:1706.03762, 2017.

[6] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. arXiv preprint arXiv:1506.03134, 2015.

[7] Dongling Xiao, Han Zhang, Yukun Li, Yu Sun, Hao Tian, Hua Wu, and Haifeng Wang. Ernie-gen: An enhanced multi-flow pre-training and fine-tuning framework for natural language generation. arXiv preprint arXiv:2001.11314, 2020.

[8] Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. arXiv preprint arXiv:2001.04063, 2020.