Summarizing Abstract for Automatic Title Assignment (SAATA)

Dicky Woo, Marshal Ma, Waqas Ali

Abstract

Recently there have been several breakthroughs in neural sequence-to-sequence models for abstractive text summarization tasks. These pre-trained models are quite powerful once being fine-tuned for specific training corpus. In this work we explore the application of these pre-trained models to summarize abstract paragraphs from academic papers archived at arXiv and propose a novel architecture that augment the seq2seq models and hybrid pointer-generator model that can copy words from the source text via pointing (to address the names/acronyms often appears on academic papers), while expanding the ability to produce novel words through the generator from "ensemble" result from multiple pre-trained (fine-tuned) transformer models.

1 Introduction

The goal of our project is to build a text summarization model, which can generate an appropriate (context-focus) title given a piece of academic paper's abstract. Text summarization is considered one of the more challenging NLP problems due to low text resources and its requirements to extract context from a sequence of text i.e., abstract in this case, and express them in brevity. This is especially challenging in long documents, where context potentially needs to be "ranked/weighted" (not necessarily according to the frequency of appearance) to create a summary text i.e., title in this case, which can express the most important context of the given long document.

The dataset we are planning to use as a starter would be the Kaggle arXiv Paper Abstracts [1], which includes 38,972 entries of academic papers (written in English) sourced from free distribution, open access archives arXiv portal[2]. However, we may also expand our dataset to include more papers from other sources. Our text summarization model would generate a predicted title as summary given an abstract of a paper as the input corpus.

Potential challenges to build reliable text summarization models specific to our data source can be due to

- Appearance of unfamiliar (technical) terms in academic papers/journals that our pre-trained models may not have encountered before
- Names/acronyms from the paper titles or abstracts that may need special handling

To address the above challenges in text summarization to generate appropriate title for academic papers, we compare and evaluate the performance following NLP summarization models trained with our train/validation dataset:

- TextRank
- BART
- PEGASUS
- T5
- Pointer-Generator model with a selected fine-tuned transformer model
- Multi-Transformer Pointer-Generator (MTPG) model with multiple fine-tuned transformer models

2 Background

The history of text generation, abstract generation specifically, dates back to the 1950s, when Hans Peter Luhn published the paper "The automatic

[1]https://www.kaggle.com/datasets/spsayak-paul/arxiv-paper-abstracts

[2] https://arxiv.org/

creation of literature abstracts"[3]. In his paper, Hans used frequency of words and phrases to extract and rank the importance of sentences. His work laid a foundation of automatic text generation. Before the deep learning-based models were developed, the techniques in automatic text summarization had been mainly extractive, where the sentences and terms used to summarize are extracted from the corpuses using some sort of text-ranking algorithm. After transformer-based language models were introduced in 2017, highly performant abstractive models started to be feasible - through fine-tuning pre-trained generative language models like BART[4], Pegasus[5] and T5[6], we can achieve much better results than text rank models.

However, these abstractive models are less reliable in reproducing factual details accurately (especially terms that are not present in training sources will result in <UNK> token). In paper "Get To The Point: Summarization with Pointer-Generator Networks"[7], See, Liu and Manning proposed a hybrid pointer-generator network architecture to address this shortcoming of these abstractive attentional models. The point-generator architecture can copy words from the source input text via *pointing*, which aids accurate reproduction of information while retaining the ability to produce novel words through the *generator*. In this paper, we replaced the bi-LSTM with attention mechanism (as encoders in) from the original pointer-generator model with pre-trained transformers that are fine-tuned with training data to leverage the *generation* capability of the pre-trained text-summarization seq2seq model.

3 Methods

In this project, the objective is to summarize academic abstracts into titles. The nature of such texts poses special challenges to build reliable text summarization models. For example:

- Appearance of unfamiliar (technical) terms
- Names/acronyms from the paper titles or abstracts that may need special handling

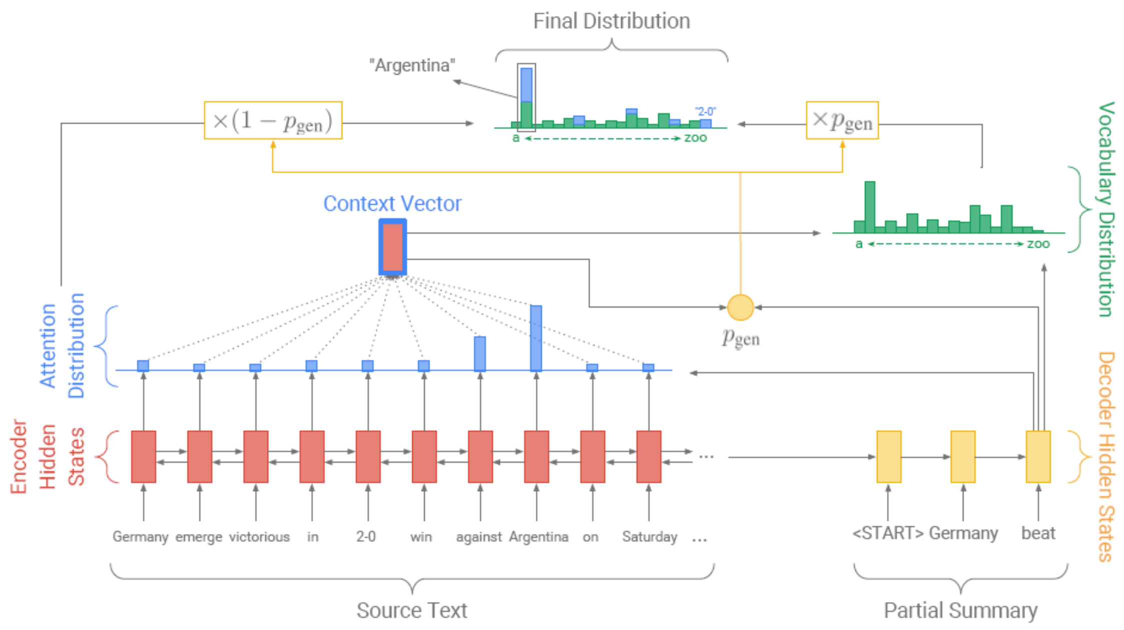

Figure 1 Pointer-generator model from "Get To The Point: Summarization with Pointer-Generator Networks". A bi-LSTM encoder-decoder with attention mechanism is used to provide information to calculate a generation probability $p_{gen_e} \in [0,1]$ to weight the probability of *generating* words from the vocabulary, versus *copying* words from source text.

[3]https://courses.ischool.berkeley.edu/i256/f06/papers/luhn58.pdf
[4] https://arxiv.org/pdf/1910.13461.pdf
[5] https://arxiv.org/pdf/1912.08777.pdf
[6] https://arxiv.org/pdf/1910.10683.pdf
[7] https://arxiv.org/pdf/1704.04368.pdf

We propose two methods, one focuses on text pre-processing, the other focuses on neural network architecture, that could potentially tackle the challenges above and out-perform the fine-tuned benchmark models.

In this section, we describe 1) our baseline seq2seq models for text summarization 2) our method to improve baseline model through academic term substitution during text pre-processing 3) our proposed TPG (transformer pointer-generator) model based on our fine-tuned baseline seq2seq models 4) our proposed MTPG (multi-transformer pointer-generator model) utilizing outputs and multi-head cross-attention from multiple of our fine-tuned baseline seq2seq models.

3.1 Sequence-to-sequence Transformer Models

The nature of abstract summarization is sequence-to-sequence. Thus, we will use the pre-trained sequence-to-sequence models, namely BART, PEGASUS and T5 as the starting point. In addition, we also included the algorithm-based TextRank approach to serve as a baseline model to compare our results. In using the pre-trained models, we will use fine-tuning as a treatment method and observe the performance lift/dip of fine-tuned vs raw models.

3.2 Academic Term Substitution During Text Pre-processing

Through observational studies, we found that the unfamiliar academic technical terms that appear in the abstracts often has the following characteristics:

1. They are often the key methods proposed in the paper
2. They will very likely appear in the title

Thus, our hypothesis is, if we can identify these terms automatically and replace all of them with a replacement token during pre-processing, then it could potentially reduce the confusion of the model and improve performance. However, one challenge on this approach is, if we replace the academic terms during preprocessing, then the generated title will only contain the replacement token, but not the actual term, which would make the model less usable in practice. Thus, we need to also record what is being replaced as each text is being preprocessed.

3.3 Transformer Pointer-Generator (TPG) Model

Despite the fine-tuned seq2seq transformer models are performing quite well, they still run the risk of not being able to generate summary title with words that are important to the input abstract text, but not necessarily as a part of the vocabulary of the pretrained fine-tuned model (refer as out-of-vocabulary, OOV).

To address this potential issue, we propose a novel variant of the original pointer-generator network architecture, *TPG (transformer pointer-generator)* model, to enable the model to copy words from the source input text via *pointing*, which we expect to improve the performance of text summarization tasks of our baseline fine-tuned seq2seq models.

The proposed hybrid pointer-generator network architecture combines the vocabulary distributions (prediction output) and multi-head cross attention distributions generated from a pre-trained text summarization seq2seq transformer models (i.e., BART, PEGASUS, T5) to predict the generation probability for each decoder timestep $p_{gen} \in [0,1]$. The probability p_{gen} for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t:

$$p_{gen} = \sigma\left(w_h^T * \mathrm{h}_\mathrm{t}^* + w_s^T * s_t + w_x^T * x_t + b_{ptr}\right)$$

where vectors w_h , w_s , w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function.

p_{gen} is used as a soft switch to choose between generating a word from the vocabulary by sampling from p_{vocab}, or copying a word from the input sequence by sampling from the multi-head attention distribution a^t over the input text. This allows the TPG model to generate summary titles that contain out-of-vocabulary (OOV) words, which is especially useful in the summarization task of turning an abstract from academic papers into an appropriate title.

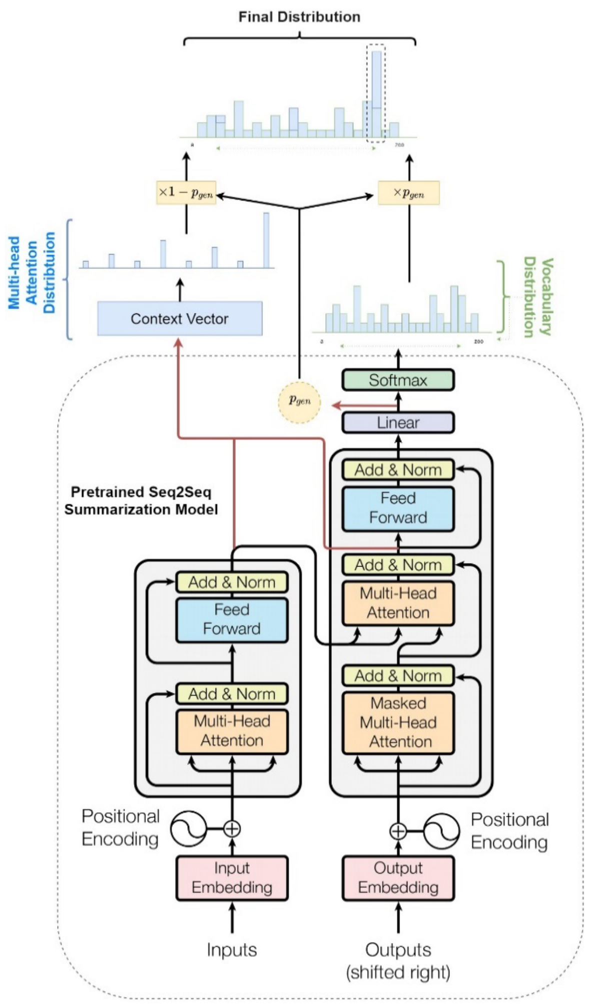

Figure 2 A pre-trained seq2seq transformer model with pointer-generator architecture, referred to as *transformer-pointer-generator (TPG)* model. Similar to the original pointer-generator model, for each decoder timestep a generation probability $p_{gen_e} \in [0,1]$ is calculated by weighting the probability of *generating* words from the vocabulary (yield from the output distribution of a pre-trained seq2seq transformer text summarization model) , versus *copying* words from the source text (given the multi-head encoder-decoder cross-attention yield from the pretrained seq2seq model. Pointer-generator architecture would have a final distribution that include copied out-of-vocabulary (OOV) words from the source input text, while retaining the text generation capability of the original pre-trained seq2seq transformer model.

The final probability distribution over the *extended vocabulary* (union of source text and the seq2seq model vocabulary) generated from each encoder can be calculated as the following:

$$P(w) = p_{gen}P_{vocab}(w) + (1 - p_{gen})\sum_{i} a_i^t$$

Note that if w is an out-of-“extended”-vocabulary (OOV) word for a specific encoder, then $P_{vocab}(w) = 0$ for the word from that specific encoder.

3.4 Multi-transformer Pointer-Generator (MTPG) Model

MTPG (multi-transformer pointer-generator) model is an extension of *TPG (transformer pointer-generator)* model proposed in the previous section. This proposed model combines multiple vocabulary distributions (prediction outputs) and multi-head cross attention distributions generated from multiple pre-trained fine-tuned text summarization seq2seq transformer models (i.e., BART, PEGASUS, T5) for an “ensemble” of distributions to predict the generation probability for each decoder timestep $p_{gen_e} \in [0,1]$. Similar to *TPG* architecture, the probability p_{gen_n} for timestep t is calculated from the context vector from each pre-trained transformer model n would be $h_{t_n}^*$, the decoder state s_{t_n} and the decoder input x_t:

$$p_{gen_n} = \sigma\big(w_{h_n}^T * h_{t_n}^* + w_{s_n}^T * s_{t_n} + w_x^T * x_t + b_{ptr}\big)$$

where vectors w_{h_n} , w_{s_n}, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Note that the decoder input x_t and its respective weight w_x should be common among these seq2seq transformers, where the word token with the highest probability in the final distribution is selected for each time step t.

The final probability distributions over the *extended vocabulary* (union of source text and the seq2seq model vocabulary) generated from each pretrained transformer model n can be calculated as the following:

$$P_n(w) = p_{gen_n}P_{vocab_n}(w) + (1 - p_{gen_n})\sum_{i_n} a_{i_n}^t$$

Note that each pretrained transformer model is likely to have a slightly different vocabulary and the resulting $P_n(w)$ could have different vector sizes, which needs to be resolved to calculate the "ensemble" final distribution as a weighted sum of the final probability distributions (predictions) from multiple pretrained transformers $P_n(w)$ would be

$$P_{ensemble}(w) = \sum_n \alpha_n * P_n(w)$$

where α_n are the learnable weights of the final probability distributions (predictions) from the *TPG* constructed from attaching pointer-generator networks to the pretrained transformer models. The *TPG* (pretrained transformer models) that can reliably generate more "accurate" summarization titles result in higher weight a_n after training has been done.

By taking an "ensemble" approach of the distribution generated from multiple transformers, we expect to improve the average prediction performance over the any individual contributing text summarization *TPG* (and to the extent of the underlying baseline seq2seq model) through reducing the variance component of prediction errors by the contributing *TPG* models.

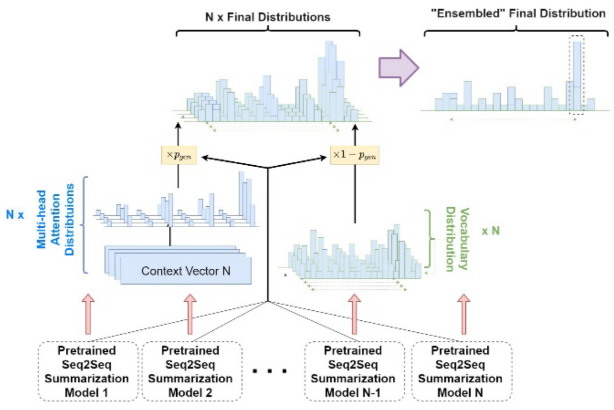
N x Final Distributions
"Ensembled" Final Distribution
$\times p_{gen}$
$\times 1 - p_{gen}$
N x
Multi-head Attention Distributions
Context Vector N
Vocabulary Distribution
x N
Pretrained Seq2Seq Summarization Model 1
Pretrained Seq2Seq Summarization Model 2
. . .
Pretrained Seq2Seq Summarization Model N-1
Pretrained Seq2Seq Summarization Model N

Figure 3 An ensemble method of multiple *transformer-pointer-generator (TPG)* models, referred to as *multi-transformer-pointer-generator (MTPG)* model. It is done by taking a weighted sum of the final distributions (predictions) from multiple *TPG* models. The weights of these final distributions are learned in the training process where more "accurate" *TPG* models are expected to have higher weights vs *TPG* models that are less consistent in generating "accurate" summarized titles. *MTPG* improves over individual *TPG* by reducing variance of the predictions (summarization of abstracts to generate titles).

4 Results and discussion

We use different flavors of Rouge scores to rank the candidates. Rouge-L will be the primary metrics that we use. In our case, the BLEU score is not a good candidate because of the length of the outputs. On average, the titles are 10 - 15 tokens in length, which makes it unlikely to find matching 4-grams, resulting in high probability of a 0 BLEU score, even when the predicted titles make sense.

Model	**ROUGE 1**	**ROUGE 2**	**ROUGE L**
Text Rank - raw	0.18	0.32	0.21
T5 - vanilla	0.22	0.19	0.2
T5 - fine-tuned (no pre-processing)	0.49	0.29	0.44
T5 - fine-tuned (with pre-processing)	0.44	0.24	0.39
PEGASUS - vanilla	0.24	0.21	0.27
PEGASUS - fine-tuned (no pre-processing)	0.53	0.33	0.47
PEGASUS - fine-tuned (with basic text cleaning and tokenization)	0.51	0.31	0.45
PEGASUS - fine-tuned (with academic term substitution)	0.45	0.26	0.4
BART - vanilla	0.24	0.11	0.22
BART - fine-tuned (no pre-processing)	0.55	0.33	0.49

Table 1 Baseline text summarization model performance (ROUGE) on our test set

From the result tables above, we can see that Fine-Tuned BART has the best performance across all the candidates which is quite intuitive since BART training scheme involves "Text Infilling" and "Sentence Permutation" whereas T5 training scheme is more like fill-in the blanks scenario and PEGASUS is similar in the sense that it also uses masking technique for pre-training.

In the case of PEGASUS we experimented with two pre-processing techniques, first one involved some basic cleaning of the data like removing punctuations whereas in the second technique we replaced potential academic terms with <UNK> token. With T5 we only used the first pre-processing scheme i.e. cleaning of the raw data. Interestingly in all cases the model performed worse than the fine-tuned model with raw data. Therefore, the two specific methods of text preprocessing didn't help in the case of PEGASUS and T5.

We examined the actual predicted outcomes and the ranking by Rough-L seems to align with our subjective evaluation. Below is an example:

Original Abstract	We consider the problem of learning low-dimensional representations for large-scale Markov chains. We formulate the task of representation learning as that of mapping the state space of the model to a low-dimensional state space, called the kernel space. The kernel space contains a set of meta states which are desired to be representative of only a small subset of original states. To promote this structural property, we constrain the number of nonzero entries of the mappings between the state space and the kernel space. By imposing the desired characteristics of the representation, we cast the problem as a constrained nonnegative matrix factorization. To compute the solution, we propose an efficient block coordinate gradient descent and theoretically analyze its convergence properties.
Actual Title	Identifying Sparse Low-Dimensional Structures in Markov Chains: A Nonnegative Matrix Factorization Approach
Text Rank	We consider the problem of learning low-dimensional representations for promote this structural property, we constrain the number of nonzero entries of the mappings between the state space and the kernel space.
PEGASUS Fine-Tuned	Learning Low-dimensional Representations for Large-Scale Markov Chains
T5 Fine-Tuned	Learning Low-Dimensional Representations for Large-Scale Markov Chains
BART Fine-Tuned	Constrained Nonnegative Matrix Factorization for Low-dimensional Representations in Markov

Table 2 Predicted output of our text summarization models given an abstract. PEGASUS and T5 model gave almost an identical result, while BART can generate a title that is closest to the actual title of the given example arXiv paper "Identifying Sparse Low-Dimensional Structures in Markov Chains: A Nonnegative Matrix Factorization Approach"

5 Conclusion

Compared with the actual title, the fine-tuned BART only missed a key term - "Identifying." However, reading the abstract, there are no mentions related to the "identification" process, which makes it reasonable for the model to not be able to pick that up. This reflects a limitation on our data - not all the information that people use to generate titles can be found in the abstract - it may come from the body of text, or even from our own summarizations. In this case, there is a natural ceiling of performance since we only use abstracts as our data sources to generate the title.

References

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020, November). Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In International Conference on Machine Learning (pp. 11328-11339). PMLR.

Raffel, Colin, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." J. Mach. Learn. Res. 21.140 (2020): 1-67.

See, Abigail, Peter J. Liu, and Christopher D. Manning. "Get to the point: Summarization with pointer-generator networks." arXiv preprint arXiv:1704.04368 (2017).

Luhn, H.P., The Automatic Creation of Literature Abstracts (1958)