**A Survey in Automatic Text Summarization and Transformers**

***Full name: Chi Nguyen Chau***

***SID: 22156675***

## Introduction

The birth of Internet brought us into the era of data. Some statistics show that approximately 328 million terabytes of data are created each day including newly generated, captured, copied and consumed data (Fabio, 2024). If we use search engines to find a topic, we will get thousands of responses. This raises a problem that how to extract key information from these mess of data while skimming, scanning, reading and understanding all of them is a difficult task for us in our busy days. For this reason, automatic text summarization (ATS) is required to support summarizing the content of documents. It can save a lot of time and effort in capturing main points and reducing possibility of skipping the key information as well as interesting texts in case of multiple documents (Gambhir & Gupta, 2017).

In definition, summary is *a* *text that is produced from one or more texts, conveys important information and no longer than half of the original text(s) as well as usually significantly less than that. Text here is used rather loosely and can refer to speech, multimedia documents, hypertext, etc* (Radev et al., 2002). Two main features of a summary text are the most important points in the original text(s) and the shrink of length which is significantly shorter than the input texts. Generating a summary automatically demands a system which can satisfy these main features. The available text summarization approaches will be discussed in the next section, but in general, the architecture (Figure 1) consists of the following tasks (El-Kassas et al., 2021):

* Pre-processing: producing a structured representation of the original text using many linguistic techniques like sentences segmentation, words tokenization, stop-words removal, stemming, etc.
* Processing: using one of the text summarization approaches by applying one or more techniques to convert the input document into the summary.
* Post-processing: solving some problems in the generated summary sentences like anaphora resolution and reordering the selected sentences before generating the final summary.

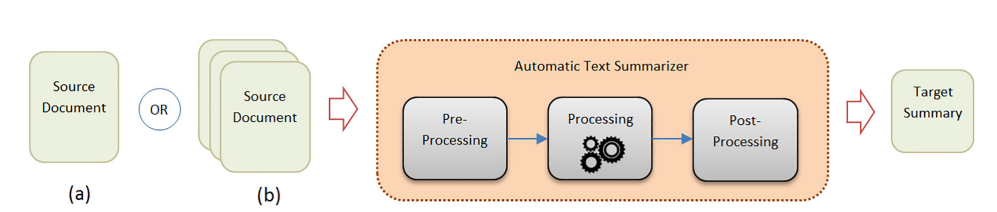


Figure 1: (a) Single-document or (b) Multi-document, automatic text summarizer. (El-Kassas et al., 2021)

To produce an effective summary, many challenges need to be handled such as identifying the most important segments in the input text to add to the summary, summarizing long documents like books, summarizing multiple documents or generating an abstract summary which contains words and sentences differing from the original texts. Moreover, the evaluation of computer-generated summary should be considered. The effort of manual evaluation is expensive. It can take a long time to finish and consume labour resources which make it difficult to conduct frequently. Therefore, we need a measurement which can evaluate the machine summarization automatically and accurately the same as human judgments. This is also a big challenge in ATS task.

## Review of earlier

The first attempt in ATS research began in 1958 with Luhn’s work that automatically excerpts abstracts of magazine articles and technical papers. Until now, several research were conducted and can be classified into three main approaches: extractive, abstractive and hybrid.

In the extractive approach, the ATS system extracts sentences in the input text to generate the summary. After pre-processing of input texts, the next steps are that creating a suitable representation to facilitate the text analysis such as n-gram, bag-of-word, graph, etc., scoring all sentences (Nenkova & McKeow, 2012), extracting high score sentences and concatenating them to generate the summary with the limitation in length as the compression rate (Nakova & McKeow, 2012). The final steps are post-processing tasks like reordering the extracted sentences, replacing pronouns with their antecedents, replacing the relative temporal expression with actual date, etc. before exporting the output. There are many methods associated with the extractive approach including statistical-based, concept-based, topic-based, sentence centrality, graph-based, semantic-based, machine learning, deep learning, optimization and fuzzy logic. The advantages of this approach are faster and simpler than the abstractive approach and producing high accuracy. The disadvantages are that it is far from the method which human experts write summaries, redundancy in some summary sentences, that extracted summary can be longer than average, temporal expressions conflicts in case of multiple documents, lack of semantics and cohesion, uncovering information confliction and the output summary may be unfair for input texts which consist of several topics.

For the abstractive approach, abstractive text summarizers generate summaries by understanding concepts in input documents using NLP methods and express them in fewer words using a clear language (El-Kassas et al., 2021). This approach makes new sentences instead of copying words from the inputs for summary generation. The architecture also consists of pre-processing, post-processing but the processing tasks include building an internal semantic representation and generating summary using natural language generation techniques instead. The outcome of this approach is better than the extractive one with higher condensation, lower redundancy, good compression rate and closer to human-generated summary. Unfortunately, in practice, generating a high-quality abstractive summary is very difficult. The current natural language generation technology is unable to adapt to the requirements of abstractive approach which needs to understand whole document before generating and able to deal with out-of-vocabulary words appropriately as well as the content beyond the representation of the input. Many methods are conveyed about the abstractive summarization such as graph-based, tree-based, rule-based, template-based, ontology-based, semantic-based and deep-learning based.

The last approach is hybrid text summarization which combine both the abstractive and extractive one. The architecture commonly consists of pre-processing, sentence extraction, abstractive summary generation and post-processing which defines dome rules ensuring the valid generated sentences like three-word minimum in length, containing-verb mandatory, ending without an article, preposition, conjunction or interrogative word. This approach inherits the advantages of both abstractive and extractive approaches, and the overall performance is improved. However, the outcome is less quality than the pure abstractive approach because it is generated based on the extracts instead of the original text. There are mainly two methods that have been used in hybrid text summarization: extractive to abstractive and extractive to shallow abstractive methods.

## Recent works

BART is a denoising autoencoder for pretraining sequence-to-sequence models corrupted text with an arbitrary noising function and learned a model to reconstruct the original text. The architecture is a combination of the bidirectional encoder as in BERT, the left-to-right decoder as in GPT and other pre-training scheme such as token deletion, sentence permutation and document rotation. It is effective in text generation, works well for comprehension tasks and achieves new state-of-the-art results on a range of abstractive dialogue, question answering and summarization tasks. (Lewis et al., 2019)

T5 is an end-to-end trained transformer model which is built based on the architectures such as BERT, GPT, etc. by utilizing text-to-text transfer learning. It can handle several tasks like translation tasks, text similarities, and text summaries. It works well in performing generating, classification, and regression tasks as well as scaling up model size and pre-training corpus (Itsnaini et al., 2023; Zhang et al., 2020).

PEGASUS is a pre-training large Transformer-based encoder-decoder models on massive text corpora with a new self-supervised objective. It processes similar to extractive summary approach that masks the importance sentences from the input document and generates them together as one output sequence. The performance on low-resource summarization surpasses previous state-of-the-art results and the output summaries achieve human performance on multiple datasets (Zhang et al., 2020).

## Introduction to Transformers

Architecturally, a transformer contains two main stacks: encoder and decoder. The encoder receives a sequence of embedded tokenized inputs and produces a sequence of vectors called hidden states which are the transformation of inputs through self-attention mechanism and feed-forward neural network. Meanwhile, the decoder receives the sequence of hidden states outputted from the encoder, however, passes each state one by one into its operation instead of whole sequence and generates outputs one at a time. This output then is fed to the next process of decoder.

In more detail, the processing flow of transformer can be described as following. The input words are converted to vectors by *input embedding* and encoded the positional information by *positional encoding*. The encoded outputs are passed into *multi-head attention* sublayers in encoder layers for assigning “attentions” to each embedding. These attention-layer outputs are fed into *feed-forward network (FFN)* sublayers in encoder layers for transformation tasks and the transformed outputs then are transferred to the decoder stack. When it comes to decoder stack, the previous decoder-layer outputs which have been handled by *output embedding* and *positional encoding* are passed into *masked* *multi-head attention* sublayers in decoder layers for assigning “attentions” before transferring to *encoder-decoder attention sublayers*. The *encoder-decoder attention* sublayersreceive the outputs that come from the previous sublayers and the encoder, calculate attentions and feed the outputs into *FFN* sublayers for transformation. A notice that outputs from sublayers of encoder and decoder are always applied *residual connections* and *normalizations* before transferring to next step. The last step in decoder layers is to convert the FFN outputs and produce the token prediction using linear transformation and softmax function.

In the above description, some scratches need to be explained more clearly, including:

**Embeddings** are used to convert the input tokens and output tokens into vectors of dimensions

( in the original work).

**Positional encoding** is a method used to add the relative or absolute positional information of tokens in the sequence. The model can make use of these information to determine the order of the sequence. The original paper of Transformer chose sine and cosine functions to encode the positions into vectors of the same dimension as the embeddings.

**Attention mechanism** is a mechanism which allows neural network algorithm to assign a different amount of weight or *attention* to each element in a sequence. In the sequence-to-sequence architecture, the encoder generates a single last hidden state containing all states of the sequence and transfers to the decoder. This is a challenge for long sequences because it has potential loss of information in each time compress the states. Moreover, the single last hidden state is a huge input for the decoder. Fortunately, the attention has come to solve this challenge when allowing the decoder to control which states need to be accessed through the term of attention so far. In the Transformer architecture, the self-attention mechanism is utilized. The “self” prefix implies that each output embedding is put attentions on all input elements or in other words, whole sequence is used to compute a weighted average of each output embedding.

**Encoder** includes several layers in which each layer contains two sublayers are multi-head attention and feed-forward network.

* ***Multi-head attention*** is a mechanism that processes several head attention layers running in parallel in which each layer transforms linearly the embeddings into a set of queries, keys and values with lower dimensions and applies *scaled dot-product attention* to this set, concatenates all outputs and projects again resulting in the final values
* ***Scaled dot-product attention*** is an algorithm of weight calculation which receives queries, keys of dimension and values of dimension as input, compute the dot products of the query with all keys, divide each by and apply a softmax function to obtain the weights on the values.
* ***Position-wise feed-forward network*** is a fully connected feed-forward network placed in each of layers in the encoder and decoder and consists of two linear transformations with a ReLU activation in between. It is applied to each embedding independently instead of processing whole sequence. The input and output dimensions are and the inner-layer dimension is .
* ***Add & Norm*** are techniques that Add refers to residual connection used to handle vanishing gradient problem and Norm refers to layer normalization which can support training model easier, hence improve the performance and training time.

**Decoder** includes several layers in which each layer contains three sublayers are masked multi-head attention, encoder-decoder multi-head attention and feed-forward network. The principle of operation in the decoder is similar to in the encoder. The difference is that the decoder utilizes two multi-head attention layers for handling the previous layer output and the encoder output.

Even though the transformer architecture was originally designed for sequence-to-sequence tasks, in practice, the transformer models can be based on only encoder stack, decoder stack or both of them. For instance, BERT and its variants (RoBERTa, DistilBERT), XLM, ALBERT and ELECTRA are encoder-only models which are well suited for tasks like text classification, sentiment analysis or named entity recognition. The family of GPT is the decoder-only models which are suitable for text generation or text completion tasks. T5, BART, PEGASUS, M2M-100 and BigBird are encoder-decoder models which can handle text summarization, translation or question and answering tasks.

In comparison to the RNN-based encoder-decoder architecture, the transformer operates self-attention in both encoder and decoder which make decoder able to access to all states at the same layer and uses feed-forward neural network instead. These make transformer able to train much faster than RNN models.

## Performance metrics

Performance evaluation is a challenging task in the field of automatic text summarization. It demands metrics that can judge the predicted summary automatically in the same manner as the human experts did. Unfortunately, designing metrics adapt to this requirement is an extremely difficult task because human evaluation is very complex, time-consuming and includes several criteria such as fluency, overall quality, informativeness, relevance, grammaticality, naturalness, coherence, accuracy, correctness, readability and so on. Even though there still haven’t been any proper metric, some automatic metrics have been used widely such as BLEU, METEOR and ROUGE. However, these automatic metrics are underinformative and do not correlate well with human evaluation. They mostly rely on N-gram overlap and cannot cover lexicality, semantics or syntactics. (van der Lee et al., 2021)

**BLEU**

*BLEU* is a metric for evaluating a translation task that is quick and inexpensive, language-independent and correlates highly with human evaluations. It is calculated by the product of brevity penalty BP and geometric mean of n-gram precision scores with n from 1 to 4.

*BP* is a penalty ratio applied on translation which is shorter than the reference corpus and computed as follows:

where c is the length of translation and r is the length of the reference corpus.

**ROUGE**

*ROUGE* is a metric for determining the quality of summary by comparing it to other summaries created by human. While the BLEU metric focuses on precision score, ROUGE concentrates on recall score which means that how much the words (and/or n-grams) in the human references appear in the candidate model outputs.

There are four different ROUGE measures including ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S.

*ROUGE-N* measures the number of matching n-grams between the model-generated text and a human-produced reference.

*ROUGE-L* is based on the longest common subsequence (LCS) between the model output and reference. The LCS is unnecessary consecutive but in order instead. A longer shared sequence indicates more similarity between the two sequences.

where is the length of a longest common subsequence of X and Y, m is length of X and n is length of Y and is or set to a very big number in DUC.

*ROUGE-W* is similar to ROUGE-L but more considering in consecutive subsequence. Output with the same LCS but higher consecutive subsequence is better.

*ROUGE-S* measures the overlap of skip-bigrams which are any pair of words in their sentence order allowing for arbitrary gaps between a candidate translation and a set of reference translations.

where is number of skip-bigram matches between X and Y, m is length of X and n is length of Y and controlling the relative importance of and .

The implementation of these metrics for BART, T5 and PEGASUS models as follows:

A white background with text

Description automatically generated

A close-up of a computer screen

Description automatically generated

Evaluating on the CNN daily mail dataset shows that PEGASUS model achieved a higher BLEU and ROUGE scores than BART and T5.

|  |  |
| --- | --- |
| BLEU | ROUGE |
| A screenshot of a computer  Description automatically generated | A screenshot of a graph  Description automatically generated |

## Conclusion and Outlook

In conclusion, automatic text summarization is a challenging task in the demand of generating an effective summary like human generated. It is hard to extract exactly the key points in the original texts and combine or paraphrase them into a summary. When a summary is generated, another challenge is that how to evaluate automatically the performance of that outcome because the manual evaluation consumes a lot of time and effort. The BLUE and ROUGE metrics are used widely but still are criticized underinformative and that do not correlate with human evaluation. These challenges require more studies to find the better models and measure metrics.

Transformer is an emerging model and beneficial a lot in sequence-to-sequence tasks. Its architecture can handle text classification, sentiment analysis, named entity recognition, translation, question and answering, text generation, completion or summarization efficiently. However, there are some challenges with scaling due to the bottleneck in infrastructure, training cost, dataset curation, model evaluation and deployment. The attention mechanism itself has a computational bottleneck when required a time and memory complexity at where n is the length of the sequence.

Using text-only to train language models has a limitation that it ignores other aspects of a real-world problem such as human bias, common sense and facts. Recently, several models are developed to train text along with vision (VisualQA, LayoutLM, DALL·E, and CLIP), table (TAPAS), speech (ASR, wav2vec 2.0). These approaches promise a bright future of using deep-learning models to address many complex real-world problems.

**REFERENCE**

El-Kassas, W. S., Salama, C. R., Rafea, A. A., & Mohamed, H. K. (2021). Automatic text summarization: A comprehensive survey. *Expert systems with applications*, *165*, 113679.

Fabio, C. (2022, Jan 11). Two minutes NLP — Learn the BLEU metric by examples. Medium. https://medium.com/nlplanet/two-minutes-nlp-learn-the-bleu-metric-by-examples-df015ca73a86

Fabio, C. (2022, Jan 19). Two minutes NLP — Learn the ROUGE metric by examples. Medium. <https://medium.com/nlplanet/two-minutes-nlp-learn-the-rouge-metric-by-examples-f179cc285499>

Fabio, D. (2024, Jun 13). Amount of Data Created Daily (2024). Exploding Topics. <https://explodingtopics.com/blog/data-generated-per-day>

Goldstein, J., Mittal, V. O., Carbonell, J. G., & Kantrowitz, M. (2000). Multi-document summarization by sentence extraction. In *NAACL-ANLP 2000 workshop: automatic summarization*.

Gambhir, M., & Gupta, V. (2017). Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, *47*(1), 1-66.

Itsnaini, Q. A. Y., Hayaty, M., Putra, A. D., & Jabari, N. A. (2023). Abstractive Text Summarization using Pre-Trained Language Model" Text-to-Text Transfer Transformer (T5)". *ILKOM Jurnal Ilmiah*, *15*(1), 124-131.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., ... & Zettlemoyer, L. (2019). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out* (pp. 74-81).

Nenkova, A., & McKeown, K. (2012). A survey of text summarization techniques. *Mining text data*, 43-76.

Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics* (pp. 311-318).

Radev, D., Hovy, E., & McKeown, K. (2002). Introduction to the special issue on summarization. *Computational linguistics*, *28*(4), 399-408.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., ... & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, *21*(140), 1-67.

Soren, D. (2023, Feb 13). Understanding Transformer model architectures. Practical Artificial Intelligence. https://www.practicalai.io/understanding-transformer-model-architectures/#:~:text=The%20Encoder-only%20architecture%2C%20on%20the%20other%20hand%2C%20is,classifier%20or%20a%20regressor%20to%20make%20a%20prediction.

Tunstall, L., Von Werra, L., & Wolf, T. (2022). *Natural language processing with transformers*. " O'Reilly Media, Inc.".

van der Lee, C., Gatt, A., van Miltenburg, E., & Krahmer, E. (2021). Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language*, *67*, 101151.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

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.