Survey on Automatic Text Summarization and Transformer Models Applicability

Wang Guan
ITMO University
Saint Petersburg, Russia
wagu0809@gmail.com

Ivan Smetannikov
ITMO University
Saint Petersburg, Russia
ismetannikov@itmo.ru

Man Tianxing
ITMO University
Saint Petersburg, Russia
mantx626@gmail.com

ABSTRACT

This survey talks about Automatic Text Summarization. Information explosion, the problem caused by the rapid growth of the internet, increased more and more necessity of powerful summarizers. This article briefly reviews different methods and evaluation metrics. The main attention is on the applications of the latest trends, neural network-based, and pre-trained transformer language models. Pre-trained language models now are ruling the NLP field, as one of the main down-stream tasks, Automatic Text Summarization is quite an interdisciplinary task and requires more advanced techniques. But there is a limitation of input and context length results in that the whole article cannot be encoded completely. Motivated by the application of recurrent mechanism in Transformer-XL, we build an abstractive summarizer for long text and evaluate how well it performs on dataset CNN/Daily Mail. The model is under general sequence to sequence structure with a recurrent encoder and stacked Transformer decoder. The obtained ROUGE scores tell that the performance is good as expected.

CCS CONCEPTS

• **Computing methodologies → Natural language generation**; **Neural networks**; **Supervised learning**.

KEYWORDS

Automatic Text Summarization, Natural Language Processing, Pre-trained language model

ACM Reference Format:
Wang Guan, Ivan Smetannikov, and Man Tianxing. 2020. Survey on Automatic Text Summarization and Transformer Models Applicability. In *2020 International Conference on Control, Robotics and Intelligent System (CCRIS 2020), October 27–29, 2020, Xiamen, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3437802.3437832

1 INTRODUCTION

With the rapid growth of enormous information generated each day on the internet, people are overwhelmed by this great amount of information. And it is invaluable source of knowledge that needs to be effectively summarized so that it can be used easily and rapidly by people in different domains. Therefore, retrieving useful information from many textual sources is quite a challenging task. Summarizing techniques are becoming more and more popular and needed under the context of the information era for this task.

A summary is the short version text produced from a single source or multiple sources while it conveys the main points of the original texts. The purpose of automatic text summarization is to create methods to produce this summary efficiently and precisely.

Although the interest in summarization has been growing since Natural Language Processing technologies are rapidly improving and new approaches are being developed almost all the time, this task of automatic text summarization is still under a mysterious cover with some questions need to be answered, e.g. how to achieve a deeper understanding, how to handle long text documents, how to improve the quality of summaries and how to improve the evaluation metrics. Answering these questions will help to get this research field deeper and further.

In this survey, we are going to review this field and mainly pay attention to the current trends of transformer-based pre-trained language models.

In the end, we propose an abstractive text summarizer to give a try on eliminating the length limitation issue of transformer-based summarizers motivated by the application of recurrent mechanism in Transformer-XL. We novelly applied XLNet [39]. on Automatic Text Summarization under a general sequence to sequence structure with a recurrent encoder and stacked transformer decoder. This is the first trial of the application of XLNet on Automatic Text Summarization. In the end, the model was evaluated on CNN/Daily Mail [25] dataset by calculating the ROUGE [16] scores and compared it with some other earlier summarizers. The results showed that this model obtained competitive results to the SOTA results and showed a big potential for improvement on both encoder and decoder parts. There are also two future works proposed.

In 1990s, as NLP techniques developed following the development of machine learning, cognitive psychology [18] and linguistics analysis also came to a new stage and were considered for usage in producing text summarizations. Also, methods for automatic text summarization were created based on the machine learning idea of training summarizers with sets of data. With machine learning methods, the summarization task could be formulated as a bi-classification problem. The sentence or passage would be classified as included in the summary or not included [26, 33].

Some machine learning methods were proposed to improve the quality of summaries, like Naïve Bayes, HHM, etc. Most popular methods among them are neural network-based methods. Kaikhah [14] proposed a machine learning approach that used neural networks to produce summaries of arbitrary length news articles. The neural network is trained on a corpus of articles and then modified, through feature fusion, to produce a summary by selecting top-ranked sentences.

2 SUMMARIZATION TECHNIQUES

Automatic Text Summarization is a process of producing a concise and meaningful summary of text from either single or multiple text resources such as books, news articles, blog posts, research papers, emails, tweets, etc. It is an interdisciplinary research field related with computer science as well as cognitive psychology and linguistics, etc. There were many problems addressed during the development of Text Summarization and researchers made a lot of the efforts on creating various methods considering factors of different fields to solve problems and improve the quality of summaries.

In terms of different eras of automatic text summarization, methods are categorized in this survey into several categories, early statistical methods, classic Machine Learning methods, Neural network-based methods, and Transformer-based methods. In this chapter, we talked about the first three categories, in the next chapter, we will mainly review the Transformer based methods.

2.1 Early works

Early works mentioned above in background had created a solid base for Automatic Text Summarization. They mainly considered physical features of text to identify the important parts using statistical methods and based on these features they could produce the summary.

Many NLP statistical techniques were used in Automatic Text Summarization to calculate upon physical features like term weight (term/word frequency), title similarity, sentence position, sentence length, thematic words, cue words, sentence to sentence similar, etc.

Term weight (term/word frequency). H. P. Luhn created the first summarization system counting word frequencies. He used word frequencies to extract words to calculate weight for each sentence counting the frequent words the sentence contained. Title Similarity. Words in titles potentially carry the main topic information of the whole text, if a sentence contains more words similar to the words in the title, this sentence would be assigned a higher score than those containing less similar words.

Sentence Position. Sentences are always evaluated by their position in the text. Baxendale took the advantages of tables of contents and sentence positions (beginning and ending parts of paragraph) to form a summary. They investigated on 200 paragraphs and the result showed that in 85 percent of them the topic sentences were at the beginning part of the paragraphs, and only in 7 percent of them at the ending part.

Sentence length. Sometimes, sentence length is also considered, when eliminating the text by deleting sentences that contain less relevant information are not expected to be included in the summary. H. P. Edmundson also proposed a hybrid method combining word frequencies, cue words, title words and sentence position importance together and evaluated on 400 technical documents with golden summaries. Since then, researchers had published many works on Automatic Text Summarization.

TF-IDF. Most statistical methods considered only internal features and needed to delete the stop words, TF-IDF (term frequency-inverse document frequency) helps to consider not only features in one document but calculates the distribution across multiple documents and also could control the effect of stop words.

TF-IDF evaluates how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus. Equations (1), (2) and (3) define the TF-IDF.

$$TF(t) = \frac{\text{Number of times term t appears in a document}}{\text{Total number of terms in a document}} \tag{1}$$

$$IDF(t) = log\frac{\text{Total number of documents}}{\text{Number of documents with term t in it}} \tag{2}$$

$$TF - IDF(t) = TF(t) \cdot IDF(t) \tag{3}$$

TF-IDF is always used in multi-document summarization (see, e.g., [4, 5, 9, 23]).

Centroid-based summarization. Dragomir [23] proposed a multi-document TF-IDF representation-based summarizer. First, they detected the topics and used a tracking system to generate several clusters, summaries were generated using cluster centroids. Modified TF-IDF was used to produce clusters of news articles on the same event ranging from 2 to 10 documents and then the centroids of clusters were used to identify which sentences are central not only to the topic of one article, but to topics of all the articles. To meet this goal, they defined CBRU to calculate the relevance of a particular sentence to the general topic and CSIS to measure sentence redundancy. In the end, top-scored sentences were selected to form summaries.

2.2 Classic Machine Learning methods

As machine learning developed and became more and more popular since the 1990s, many techniques were applied into the NLP field to help researchers improve the summary quality. The idea behind machine learning is to train the model for a specific task like classification, or regression, when it is given an existing relevant dataset.

In Automatic Text Summarization, it is mostly viewed as a sequence classification problem. Sequences are classified into two categories, summary sentence or non-summary sentence. The training set helps the model recognize such sequences in a supervised way.

Naïve Bayesian classifier. Naïve Bayesian classifier method is a supervised learning method that can also be applied as a language model, when only individual word features out of any other features (dictionaries, URLs, email addresses, phrases and so on) used in Bayes classifier and all words in the text are used. Bayes model can be viewed as a set of class-specific unigram language models and for each class it assigns a probability to each word $P(w|c)$.

Also, each sentence can be classified as a summary sentence or not a summary sentence. Sentence length feature, fixed-phrase feature, paragraph feature, thematic word feature and uppercase word feature were considered. These features F are jointly considered in the model. The probability of each sentence being included in the summary S was calculated using Bayes rule as:

$$P(s \in S|F_1, F_2, ...F_k) = \frac{\prod_{j=1}^{k} P(F_j|s \in S)P(s \in S)}{\prod_{j=1}^{k} P(F_j)} \tag{4}$$

where $P(s \in S)$ is a constant, $P(F_j|s \in S)$ and $P(F_j)$ are estimated from the training data by counting the occurrence. Each sentence was assigned a score and sentences with top scores were selected as summary.

Hidden Markov Model. Hidden Markov Model is based on Markov Chain, enhanced. It calculates the probabilities of that sequences from random variables of word or symbol sets can represent anything, word or sentence set, in Automatic Text Summarization. The HMM considers not only the states that we can observe (like words, phrases) like in Markov Chain, but also the states that we cannot observe (hidden states like tags of words).

Conroy and O'leary [19] presented the first Hidden Markov Model use in Automatic Text Summarization to produce summaries. There were two kinds of states, same as the two labels in the classification task, a sentence would be assigned a label as True, which indicates the sentence should be included in the summary, and otherwise False. Five features, position of the sentence in the document, position of sentence within its paragraph, number of terms in the sentence, how likely the terms are given a baseline of terms, how likely the sentence terms are given the document terms, were considered in the development of a Hidden Markov Model for summarization.

The model was given the set of features and an a-posterior probability that each sentence was either a summary sentence, or not was calculated. It does not assume that the probability that sentence i is in the summary, is independent of whether sentence i-1 is in the summary. They compute the probability that a sentence is a summary sentence by summing $\gamma_t(i)$ over all even values of i.

Graph-based methods. Graph-based methods are always used to calculate the importance of sentences within the summary by constructing a graph network and calculating the weight for each sentence. Each article is mapped into a graph in which sentences are vertices and edges between sentences indicate the similarity of these sentences. Then a ranking algorithm is applied to sort them according to their similarity and sentence weight, the summary is formed by selecting the top sentences. The generally used ranking algorithm is TextRank [24]. The idea of TextRank is inspired by PageRank of Google. PageRank is used to rank the websites, if a webpage is linked by many other pages, then it is considered important. Textrank is a graph-based algorithm. In TextRank, each sentence in the graph has a relationship with other N sentences in front of it and N sentences behind it. Unlike PageRank, TextRank is undirected graph constructed from natural language text, and may include multiple or partial links between sentences extracted from the text.

2.3 Neural network-based methods

The beforementioned methods mostly produce summaries in the extractive way. With the blooming development of deep neural networks, NLP is also having a rapid improving opportunity for both Extractive Summarization and Abstractive Summarization. For Automatic Text summarization, mainly Abstractive Summarization, is improved following the development of neural networks.

There are many works using neural networks and also attention mechanism and most of them have a very common procedure, 1) create a tokenizer or look-up table to transform the tokens from the source text (including words and special symbols) into a continuous vector (word embedding), 2) the vector is processed by an encoder and outputs a fixed-length vector called representations, 3) the representations are fed into an extractor (extractive summarization) or a generator (abstractive summarization) to form a summary.

ABS [21] introduced three different encoders based on separately Bag of words, Convolutional Neural Network and Attention to encode the input sequence and generate proper representations. They used Bag of Words to produce word embeddings and simply summed them up.

For CNN encoder, they utilized a standard time-delay neural network (TDNN) architecture, alternating between temporal convolution layers and max pooling layers allowing local interactions between words while also not requiring the context while encoding the input. The attention-based encoder derives from [8] and produces a single representation for an entire input sequence of time step t.

The decoder is a feedforward neural network-based language model (NNLM) for estimating the contextual probability distribution of the generation for each time step.

At the training stage, the encoder and decoder are trained jointly with mini-batch stochastic gradient descent given local condition constraints and estimate the parameters to minimize the negative log-likelihood loss with parameters Θ of a training set.

Chopra [28] further extended [21] by using a recurrent neural network instead of feedforward network and outperforms [21] on multiple data sets. They also improved the convolutional encoder to an attentive convolutional encoder.

For the decoder, they replace the feedforward neural network of [21] separately with Elman RNN and LSTM. At the training step, given training set $S = (s_1, y_1), \ldots, (s_n, y_n)$ of S source-summary pairs, they use stochastic gradient descent to minimize the negative conditional log-likelihood with parameter Θ. The experiments showed that, their work outperformed the RAS-Elman architecture at that time on Gigaword corpus.

Nallapati in [25] addressed some problems in summarization including keywords modeling, sentence-to-word hierarchy structure capturing and OOV (out of vocabulary) issue. The OOV issue was caused because the vocabulary of the language model size is fixed, and obviously it cannot contain every one of the words in the world. Normal methods always use a special token like <UNK> to replace the out of vocabulary word, which can result in a summary with unknown tokens. To deal with OOV issue, the normal way is to use a pointer mechanism to point to the location of the word in the source text.

They also proposed three models separately to solve the corresponding problem, 1) Feature-rich encoder with the use of TF and IDF, POS, NER tags, which are always used in extractive methods to capture keywords, this is a novel use of these features in deep learning for abstractive summarization, 2) Switching pointer generator to deal with the out of vocabulary words. They add a 'switch' in the decoder to decide to use generator or pointer, 3) Hierarchical Attention to capture the hierarchy document structure. It is designed to extract information from both word and sentence level using two bidirectional Recurrent Neural Networks each with an attention weight.

Another contribution of this work is that they proposed a new dataset to CNN/Daily Mail, which contains news articles and multi-sentence summary pairs. The previous datasets either are too small for neural networks or contain only one sentence. Pointer-Generator Network [32] addressed two shortcomings of previous work, 1) they mostly are easy to generate inaccurate factual details, 2) they tend to generate some tokens or phrases repeatedly.

To solve these two issues, they proposed two models, 1) a baseline model in which the encoder is a single layer bi-directional LSTM, and the decoder is a single layer unidirectional, 2) a pointer generator model which combines a pointer network into the first common seq2seq model, the pointer switch is very similar to Nallapati [25] to control whether to copy from the source text or generate from the vocabulary. Moreover, a novel coverage mechanism also was proposed to solve the repetition problem.

ELMO [20], embeddings from language models, proposed a pre-trained language model with two bidirectional LSTM layers and it was pre-trained on large corpus in semi-supervised way. It proposed two questions, 1) how to model complex word characteristics, 2) how the representations change across the context. To answer them, they implemented the model with bidirectional LSTM layers, different from the general language models the distribution is calculated from the direction left to right, they also calculated the distribution from right to left, in the end, they maximize the log-likelihood of both directions.

ELMO is important to automatic text summarization, since it is a task that relies more on token representations, especially for abstractive summarization. Edunov [30] proved that making use of the representations from pre-trained ELMO can help improving the performance of the model for a specific task by applying ELMO separately as encoder and decoder.

2.4 Transformer models

In 2017, in "Attention is all you need" [6], Transformer was proposed and proved that many self-attention mechanisms could be used in machine translation to learn text representations. It opened the era of Transformer.

As stated in the title of [6], the traditional CNN and RNN are abandoned in the Transformer, and the entire network structure is entirely composed of the Attention mechanism. To be more precise, the Transformer consists of only self-Attention and feed-forward neural networks. A trainable neural network can be built by stacking the Transformer. The experiment is to build 12 layered model, 6 layers for each of encoder and decoder, and with this model they achieved new top BLEU scores in machine translation tasks, this proved the feasibility of the Transformer. Following it, many researchers turned their attention to Transformer since its success in machine translation.

Actually, only new language models cannot make them so successful as today, one reason is that training a large transformer-based language model in a specific domain is too challenging, so another technique is needed which and what most recent successful models are using, which is pre-training, a concept from Transfer Learning, a training strategy of Machine Learning.

Deep learning had two problems on training data sets, 1) high dependence on large scale data set, 2) lack of large scale labeled data set. Transfer learning eliminates the assumption that the training data must be independent and identically distributed (i.i.d) from the test data. In transfer learning, the training data and test data need not to be i.i.d. There is no need to train the model from scratch in a specific target field, and it can significantly reduce the need for large scale training data and enormous training time. ELMO, OpenAI-GPT [3] and ULMFiT [12] are classic models pre-trained on large scale corpus and made very big contributions.

OpenAI-GPT is a generative language model pre-trained on various large unlabeled text corpora. They achieved top scores on many NLP tasks just by fine-tuning the pre-trained model with a small specific field labeled data set. OpenAI-GPT2 is the successor to GPT, it is a large transformer-based language model with 1.5 billion parameters, trained on a dataset of 8 million web pages. The training objective is very simple, just to predict the next word give all its previous contextual information. As its autoregressive manner, it is always used in text generation tasks as a decoder.

2018, it was the year of NLP because of the publication of BERT [11]. BERT made the full use of Transformer and Transfer Learning. BERT is also a bidirectional language model, but different from ELMO, ELMO separately trained two directions, BERT makes use of contextual information from both directions simultaneously and also it proved that deep language model that can more precisely model the tokens to representations. Also different from GPT and GPT2, the GPTs only use the previous contextual information, BERT used contextual information from both directions, also the training objective is different, BERT trained the model on two objectives, 1) predict the masked tokens, parallelly, on contextual information from both direction, 2) predict the next right sentence.

At the training stage of Transformer, the context length is fixed and for fine-tuning, the model cannot obtain context dependencies that are longer than this fixed length. While encoding long text, the common way for Transformer to fit the length is to truncate it to the max length setting, this results in a context fragment issue in Transformer. To solve this problem, Google published Transformer-XL [37] in 2019 to solve this problem, in this work, they proposed to use recurrent mechanism with memory to form the long-term dependencies between segments. Following Transformer-XL, XL-Net [39] was published based on Transformer-XL to replace the position of BERT. XLNet trained the model in the autoregressive way, different from the autoencoding way of BERT, this way is more suitable for generative model, so on many tasks especially generative tasks, XLNet outperforms BERT.

BERT started a new research field, which is called BERTology, people analyze each part of BERT like the influence of the number of attention heads, how does BERT perform so well, how does each component contribute to the whole mode, etc., many researchers also try to make some operations on it like distilling, inheriting, imitating, etc. Many BERT-like works were published during the recent two years [34, 35, 38].

UniLM [15] is a pre-trained model for both Natural Language Understanding and text generative tasks. They jointly pre-trained this model using three types tasks, 1) unidirectional objective (separately left-to-right and right-to-left) in which the model can only attend context from one direction (from left or right) to predict the masked tokens, 2) bidirectional objective in which the model can attend both direction at the same time and 3) sequence-to-sequence prediction.

UniLM can be applied to Natural Language Understanding and text generative tasks. Results showed that, they outperform the previous works on multiple tasks including abstractive summarization on CNN/Daily Mail.

2.5 Transformer on Summarization

Since the transformer-based models have achieved great advanced improvement for multiple NLP tasks, especially on Machine Translation, which is a generative task, researchers of Automatic Text Summarization also turned their attention to them, especially pre-trained language models from neural networks.

Zhang [10] firstly applied BERT on Automatic Text Summarization and evaluate their model on CNN/Daily Mail and NYT datasets. With the help of BERT and their two-stage processing, they get the state-of-the-art result at that time. They applied BERT as an encoder to produce representations and feed the representations into a transformer decoder to generate a draft, and then the draft was fed into the BERT encoder as input to create the draft representation by masking each token of the draft text, the draft representation will be fed into the decoder again along with the input sequence to form the final summarization.

T5 [7] explored the limits of Transfer Learning and proposed a unified framework that can generalize every language problem into a text-to-text format, when given a sequence of text as input and generating a new sequence of text as output. They scaled up the model size (up to 11 billion parameters) and pre-trained the model with their new proposed larger scale dataset C4 (“Colossal Clean Crawled Corpus”), which consists of hundreds of Gigabytes text from the Internet. Results showed that they achieve SOTA result on many NLP downstream tasks, including summarization.

BART [17] proposed a Transformer-based denoising autoencoder for pre-training seq-to-seq models and pre-trained by randomly corrupting text with an arbitrary noising function and learning to reconstruct the original text. It is also a model using the sequence to sequence structure with a bidirectional encoder and single direction (from left to right) decoder. The objective is reconstructing the original text corrupted by various noising functions. They evaluate the model on various datasets of different downstream NLP tasks, and results showed that they achieved SOTA results including summarization on CNN/Daily Mail.

PEGASUS [13] explored the objectives while pre-training for Automatic Text Summarization. They proposed pre-training large Transformer-based encoder-decoder models on massive text corpora with a new self-supervised objective. Different from other pre-trained models, this model’s target is especially for Automatic Text Summarization, but not for whole NLP task set. During training, they used their new proposed objective, gap-sentences generation, in which they masked or removed not tokens, but sentences and trained the model on generating the masked or re-moved sentences as one sentence output.

They evaluate this model on 12 datasets covering different type of articles of fields like news, short stories, emails, etc. Results showed that they outperformed all other recent works. It remained the SOTA result on CNN/Daily Mail in 2109. ProphetNet [36] is also a sequence-to-sequence model pre-trained on a new pro-posed objective, which they call as future n-gram prediction with their new proposed n-stream self-attention mechanism derived from two-stream self-attention of XLNet. They also used n-step ahead optimization strategy instead of the commonly used 1-step ahead optimization to optimize the model. The model was pre-trained on one base scale dataset (16GB) and a large-scale dataset (160GB) respectively. They also fine-tune the model on CNN/Daily Mail and achieved SOTA results in 2020.

3 OUR WORK

The aforementioned works are all based on Transformer models, which is the new trend in NLP field, however, there is a problem that the input and context length are fixed. Pre-training models is the new solution to solve NLP downstream task, however, there are not many pre-trained language models yet. Creating pre-trained language models remains huge improving space. Classic pre-trained language models are BERT, BERT-like models, recent XLNet, etc. For summarization, researchers also apply the pre-trained models on summarization under the sequence to sequence structure. Most of them fine-tune BERT as the encoder and make full use of the representations produced by BERT. As BERT is a transformer-based language model, it also has the fixed length issue. In most cases, the length is fixed as 512 or 1024 and they just truncate the input sequence to fit the input length, in this way, a lot of information that can potentially improve the quality of representations is lost especially when meeting long text.

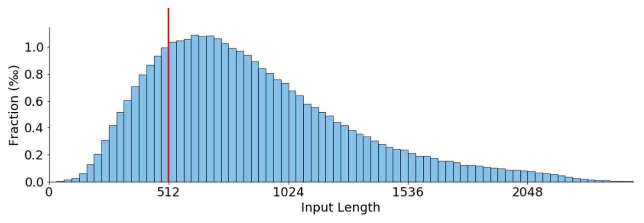
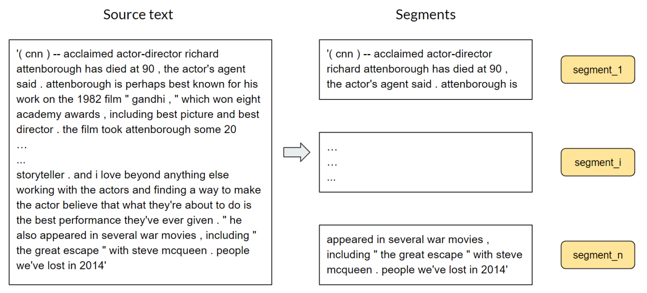


Figure 1: The input length and target length distribution of CNN/Daily mail dataset. Most articles have the length more than 512

For CNN/Daily Mail dataset, only a small part of articles has the length of less than 512, which means only this small part of the articles can be completely encoded, the article length distribution is shown in Figure 1 and we can see that most articles have a length of more than 512. In Table 1. It shows the statistic of different datasets including CNN/Daily Mail on which we will evaluate our model.

Different from other NLP downstream tasks, automatic text summarization requires more deep understanding of the words, sentences, and paragraphs across the whole article. To make use of the complete information of the whole articles, in this paper, we propose a model under the general sequence to sequence structure with a recurrent XLNet encoder and a stacked Transformer decoder.



'(cnn) -- acclaimed actor-director richard attenborough has died at 90 , the actor's agent said . attenborough is perhaps best known for his work on the 1982 film " gandhi , " which won eight academy awards , including best picture and best director . the film took attenborough some 20
...
...
storyteller . and i love beyond anything else working with the actors and finding a way to make the actor believe that what they're about to do is the best performance they've ever given . " he also appeared in several war movies , including " the great escape " with steve mcqueen . people we've lost in 2014'

'(cnn) -- acclaimed actor-director richard attenborough has died at 90 , the actor's agent said . attenborough is

...
...
...

appeared in several war movies , including " the great escape " with steve mcqueen . people we've lost in 2014'

Figure 2: The whole article is chucked into segments; each segment has the length same as the input sequence length setting

Theoretically, this model can encode any article in arbitrary length. We showcase how XLNet can be applied on automatic text summarization to encode the complete information across the whole article.

The encoder is a simple XLNet and we fine-tuned it during the training stage. It encodes the whole article recurrently until the last part of the article, it produces not only the representation but also the memory in which the content is the information from the previous time step, in this way, each part can obtain more contextual information than information only inside the current sequence.

To do this, we firstly break the whole article into several segments with the same length according to the max length setting shown in Figure 2. The length setting is limited by the ability of GPU, especially by its memory. For each article, the number of segments is different due to the different lengths of articles.

During the training stage, we make the full use of the memory mechanism of XLNet, the segments will be fed into the model one by one to produce a hidden state and a memory. The memory will be fed into the model again for the next time step together with the next segment. In this way, it helps creating long-term dependencies between different segments and the segments are tied together again as a whole article. The model structure is shown in Figure 3.

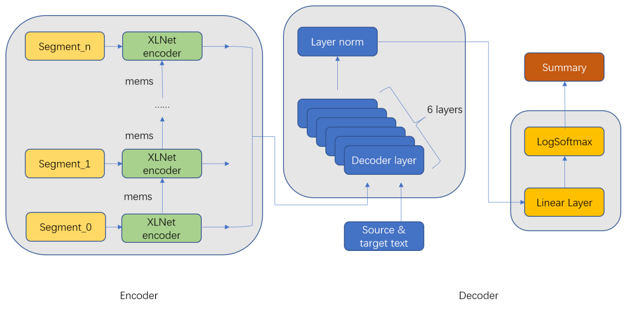


Figure 3: XLNet encoder takes each segment recurrently and produces hidden state and memory for the next time step input. Decoder take the sum of hidden states from encoder and Source and Target text to generate summaries

The decoder is composed of 6 stacked identical transformer layers. Each layer contains 2 sub-layers with multi-head self-attention, context-attention and followed by position-wise fully connected feed-forward network, respectively.

The obvious problem is that the encoder is based on pre-trained language model and the decoder is needed to be trained from scratch. If the model trained jointly with the same initialization of learning rate, parameters, optimization functions, the encoder might overfit while the decoder got underfit. The assumption is that jointly training of encoder and decoder should be initialized with different learning rate and individual optimization function. Otherwise, the decoder will not have enough ability to decode the representations from the encoder. So, we separately initialize the encoder and decoder so that the decoder can converge while fine-tuning the encoder. We use two Adam optimizers on both encoder and decoder each with the parameter $\beta_1 = 0.9$ and $\beta_1 = 0.999$, the schedule for learning rate is also separately initialized, for encoder we set warm-up steps with the learning rate initialized as $2e^{(} - 3)$, the schedule is:

$$lr_e = \widetilde{lr_e} \cdot min(step^{-0.5}, step \cdot warmup_e^{-1.5}) \tag{5}$$

where $\widetilde{lr_e}$ is the initial learning rate. For decoder, we set up different initial learning rate and warm-ups:

$$lr_d = \widetilde{lr_d} \cdot min(step^{-0.5}, step \cdot warmup_d^{-1.5}) \tag{6}$$

where $\widetilde{lr_d}$ is the initial learning rate 0.1.

3.1 Evaluation methods

Nowadays the evaluation of summarization is still a problem, and many people are working on it, trying to improve the metrics to make it fair. Even though there are some good metrics that are accepted by most people, however the used algorithms are not particularly ideal.

The easiest way to evaluate a summary is to invite several experts of this field to evaluate manually. In this way, more aspects are considered, and the results are relatively reliable. But the cost of manual checking is relatively highly time consuming and not suitable for big sets of summaries and various domains. Therefore, automatic evaluation metrics are highly demanded.

Automatic evaluation is efficient, consistent, and saves much more time for human, but it is not as accurate. The automatic methods are divided into two categories, 1) Intrinsic methods: referenced summaries are provided and used as benchmarks to evaluate the automatically generated summaries. The more the generated summaries match the referenced ones, the higher the scores are, 2) Extrinsic methods: referenced summaries are not provided. The intrinsic methods are the most used methods for automatic evaluation. The most famous intrinsic methods nowadays are ROUGE [16] and Pyramid [2].

Most papers report their result evaluated by ROUGE. It is a recall-oriented evaluation method, and it evaluates summaries based on the co-occurrence information of n-grams in the summaries. The basic idea is to generate manual summaries by multiple experts to form a standard summary set, compare the generated summaries with the manually generated standard summaries, and count the basic overlapped units to evaluate the quality of the summaries. The ROUGE criterion is composed of a series of evaluation methods, including ROUGE-N (N = 1, 2, 3, 4 respectively representing models based on 1 to 4 grams), ROUGE-L, etc.

In the research of automatic text summarization, the appropriate ROUGE method is generally selected according to the specific research content. ROUGE-1, 2, and L are the most used methods.

Pyramid is another widely used metric, different from ROUGE, Pyramid requires multiple referenced summaries for one article. In the pyramid method, some content units (words, phrases, sentences) are divided into tiers, the higher the tier is the more important it is, so the content units in high tiers show more importance than other content units. While evaluating the generated summaries, content units in high tiers contribute more to the summaries. Summary containing more high tier content units should be rewarded with more scores.

Different evaluating metrics are developed and try to behave similar to human evaluating, however, they are using different algorithms to calculate considering different aspects, and still, none can provide very persuasive results. Different kinds of summary types could also be bias on different evaluating metrics.

3.2 Datasets

Following the increasing of interest in NLP field, especially the interest in Automatic Text Summarization, datasets for this task were released to train summarization models. In terms of the type of the referenced summaries, there are One sentence summary and Multi-sentence summary. One-sentence summary data sets are usually used for topic aware summarization. All observed datasets stats are shown in table 1.

Gigaword [27] is a dataset produced by Linguistic Data Consortium and ISBN. There are 314 files containing 4,111,240 documents from 7 different sources. The summaries are one-sentence summaries in it. Xsum [31] is a large-scale data set containing BBC articles with corresponding one-sentence summaries. Each article has one extreme summary, which is created manually mostly by its author. It contains 226,711 BBC articles including 204,045 training samples, 11,332 validating samples and 11,334 testing samples. This data set is designed for an extreme summarization task, which mainly calls for abstractive summarization.

Table 1: Table shows the size of datasets, average of article length on word and sentence and average summary length

Dataset	Articles	Avg. article length		Avg. summary length
		Word	Sentence	
CNN	92K	760	34	46
Daily Mail	219K	653	29	55
NYT	655K	800	35	46
XSum	230K	431	20	23
arXiv	215K	4938	-	220

CNN/Daily mail [25] is a data set that comprises the multi-sentence summaries. It is modified from an existing data set for question-answering task. The summaries are all manually written in the CNN and Daily Mail websites. It contains 286,817 training pairs, 13,368 validating pairs and 11,487 testing pairs. NYT [29] is a dataset published by New York Times. It contains 1.8 million articles and summaries written by authors and editors in newsrooms of 38 major news publications. It combined both extractive and abstractive strategies. arXiv [1] is a scientific paper set. It collected from arXiv.org. The main reason for collecting scientific papers is that papers normally contain very long text and follow a certain discourse structure. The most important reason for this is that papers have abstracts as ground truth summaries already. Figures and tables in papers were removed and extremely long and short ones are also removed.

Despite the aforementioned datasets, there are still many other datasets for this task like PubMed, WikiSum, Chinese character dataset LCSTS, BigPatent, etc.

3.3 Implementation and experiment

We evaluated the model on the dataset CNN/Daily Mail, it contains news articles, and the targets are all multi-sentence summaries. We split the samples following its original splits like training/validating/testing as 90,266; 1,220; 1,093 for CNN and 196,961; 12,148; 10,397 for Daily Mail.

For implementation, we use Pytorch, OpenNMT and 'xlnet-base-cased' version of XLNet from Transformers of Hugginface. All tokens are tokenized using XLNet tokenizer. All source codes are shared on Github[1].

Here are the settings of the model, dropout with rate of 0.1 and label smoothing with the factor 0.1 are used. Each decoder contains 12 heads and 768 hidden units and for the feedforward layer is 3072. For generating, beam search with size 5 is used. For the out of vocabulary problem and repetition issue, we did not apply anything to forbid these issues, but thanks to the tokenizer pre-trained language, we rarely met these issues.

The model was trained on RTX 2060 mobile with gradient accumulation every 10 steps. As the limited ability of GPU and various of numbers of segments, we can only use batch size 1. Finding a way to implementation of more batch size remains to be explored. The training process lasted 7 days for 200,000 steps. We used top 3 checkpoints from evaluating phase to test.

Another problem occurred during training; it is about using batches. Since the length varies across the whole dataset, this resulted in various number of segments. If we want to process in batches, each batch should contain the same number of segments. This problem will be talked about in discussion.

3.4 Obtained results

The metric we use to evaluate the model is ROUGE [16] and the results are reported on unigram (ROUGE-1), bigram (ROUGE-2) and Longest Common Subsequence (ROUGE-L). In Table 2, it shows my results on the CNN/Daily Mail dataset. My work was the first to apply XLNet on summarization and obtained very competitive results.

As XLNet is still a very new published model, there is not any previous work applied XLNet on Automatic Text Summarization. My model is just under a very general encoder-decoder structure with applying any other algorithms for like Out Of Vocabulary problem or Repetition issue. Even though, my model still obtained

[1]https://github.com/wagu0809/XLNetSummarization

very competitive results when compared to other recent works. In this case, there is still space for application of XLNet. Also, the decoder is just an ordinary transformer decoder and trained from scratch. There are some options for decoder like GPT, GPT-2 and XLNet also can be a decoder.

Table 2: ROUGE results on CNN/Daily Mail. Results for comparison are directly taken from their papers

Models	ROUGE-1	ROUGE-2	ROUGE-3
BERT			
Two-Stage + RL	41.71	19.49	38.79
BERT-ext + abs + RL + rerank [22]	41.90	19.08	39.64
BertSumExtAbs	42.13	19.60	39.18
UniLM	43.33	20.21	40.51
Text-to-Text Transfer Transformer (T5)	43.52	21.55	40.69
BART	44.16	21.28	40.90
PEGASUS	44.17	21.47	41.11
ProphetNet	44.20	21.17	41.30
XLNet			
XLSum (this work)	43.36	20.68	40.52

4 CONCLUSION

In this survey, we reviewed the task of Automatic Text Summarization. We introduced the background including its history from its beginning to the current state, application, and its taxonomy. The current famous classification is Extractive and Abstractive Summarization, since the neural network improved the generative ability. After Transformer published, it became the new trend in NLP field. More and more researchers turn their attention to transformer. This resulted in the powerful pre-trained language models like GPT, BERT, XLNet, etc. Pre-trained models nowadays are becoming new trend in NLP field. Many works using pre-trained language models on Automatic Text Summarization and proved the huge compact.

During our own work, there are two issues we marked as the future works, 1) decoder alternatives, 2) batch size implementation.

Another future work is to implement the use of batch size. As mentioned above in Implementation part, since the length varies across the whole dataset, this results in various numbers of segments. Using batch size requires the same length of batch items, that means each batch item should contain the same number of segments.

In this paper, we also proposed a model based on XLNet for the first trial to explore the application of XLNet on summarization. Base on the experience from previous works using transformer-based language models, the models using pre-trained language model can effectively improve the performance on the task of Automatic Text Summarization. Also, we can see that there is still huge space for the application of XLNet on summarization. Also, the decoder can be advanced by replacing other more powerful ones like GPT, GPT2, etc. than general Transformer Decoder.

ACKNOWLEDGMENTS

This work is financially supported by the Government of the Russian Federation, grant 08-08.

REFERENCES

[1] Cohan A et al. 2018. 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*. 615–621.
[2] Nenkova A, Rebecca P, and Kathleen M. 2007. The pyramid method: Incorporating human content selection variation in summarization evaluation. *ACM Transactions on Speech and Language Processing* 4, 2 (2007).
[3] Radford A, Narasimhan K, Salimans T, and Sutskever I. 2018. Improving language understanding by generative pre-training. www.cs.ubc.ca/~amuham01/LING530/papers/radford2018improving.pdf
[4] Rasim M A, Ramiz M A, and Nijat R I. 2013. Multiple documents summarization based on evolutionary optimization algorithm. *Expert Systems with Applications* 40, 5 (2013), 1675–1689.
[5] Rasim M A, Ramiz, M A, Makrufa SH, and Chingiz AM. 2011. MCMR: Maximum coverage and minimum redundant text summarization model. *Expert Systems with Applications* 38, 12 (2011), 14514–14522.
[6] Vaswani A amd Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, and Polosukhin I. 2017. Attention is all you need. *Advances in neural information processing systems* (2017), 5998–6008.
[7] Raffel C et al. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv:1910.10683* (2019).
[8] Bahdanau D, Cho K, and Bengio Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473* (2014).
[9] Gunes E and Dragomir R. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence* 20, 1 (2004), 457–479.
[10] Zhang H, Jingjing C, Jianjun, and Ji W. 2019. Pretraining-Based Natural Language Generation for Text Summarization. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. 789–797.
[11] Devlin J, Ming-Wei C, Kenton L, and Kristina T. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186.
[12] Howard J and Sebastian R. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 328–339.
[13] Zhang J, Zhao Y, Saleh M, and Liu P J. 2019. PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *arXiv:1912.08777* (2019).
[14] Kaikhah K. 2004. Text summarization using neural networks. In *Proceeding of second conference on intelligent system*. 40–44.
[15] Xu K, Jimmy B, Ryan K, Kyunghyun C, Aaron C, Ruslan S, Rich Z, and Yoshua B. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International conference on machine learning*. 2048–2057.
[16] Chin-Yew L. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of ACL Workshop "Text Summarization Branches Out"*. 8.
[17] M. Lewis et al. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv:1910.13461* (2019).
[18] Chandra M, Gupta V, and Paul S K. 2011. A statistical approach for automatic text summarization by extraction. In *Proceedings of 2011 International Conference on Communication Systems and Network Technologies*. 268–271.
[19] Conroy J M and O'leary D P. 2001. Text Summarization via Hidden Markov Models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 406–407.
[20] Peters M, Mark N, Mohit I, Matt G, Christopher C, Kenton L, and Luke Z. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2227–2237.
[21] Rush A M, Sumit C, and Jason W. 2015. A neural attention model for abstractive sentence summarization. *arXiv:1509.00685* (2015).
[22] Vinyals O, Meire F, and Navdeep J. 2015. Pointer networks. *Advances in neural information processing systems* (2015), 2692–2700.
[23] Dragomir R R, Hongyan J, Małgorzata S, and Daniel T. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management* 40, 6 (2004), 919–938.
[24] Mihalcea R and Paul T. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*. 404–411.
[25] Nallapati R, Bowen Z, Caglar G, and Bing X. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv:1602.06023* (2016).

[26] Oak R. 2016. Extractive techniques for automatic document summarization: a survey. *International Journal of Innovative Research in Computer and Communication Engineering* 4, 3 (2016), 4158–4164.

[27] Parker R, Graff D, Cong J, Chen K, and Maeda K. 2011. English Gigaword. https://catalog.ldc.upenn.edu/LDC2011T07

[28] Chopra S, Auli M, and Rush A M. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 93–98.

[29] Evan S. 2008. The New York Times Annotated Corpus. https://catalog.ldc.upenn.edu/LDC2008T19

[30] Edunov S, Alexei B, and Michael A. 2019. Pre-trained language model representations for language generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. 4052–4059.

[31] Narayan S, Shay BC, and Mirella L. 2018. Pretraining-Based Natural Language Generation for Text Summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 1797–1807.

[32] Peter J L See A and Christopher D M. 2017. Get to the point: Summarization with pointer-generator networks. *arXiv:1704.04368* (2017).

[33] Gupta V and Lehal G S. 2010. A Survey of Text Summarization Extractive Techniques. *Journal of Emerging Technologies in Web Intelligence* 2, 3 (2010), 258–268.

[34] Sanh V, Lysandre D, Julien C, and Thomas W. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arxiv.org/pdf/1910.01108* (2019).

[35] Liu Y, Myle O, Naman G, Jingfei D, Mandar J, Danqi C, Omer L, Mike L, Luke Z, and Veselin S. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv:1907.11692* (2019).

[36] Yan Y, Qi W, Gong Y, Liu D, Duan N, Chen J, Zhang R, and Zhou M. 2020. ProphetNet: Predicting Future N-gram for Sequence-to-Sequence Pre-training. *arXiv:2001.04063* (2020).

[37] Dai Z, Zhilin Y, Yiming Y, Carbonell G J, Quoc L, and Ruslan S. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2978–2988.

[38] Lan Z, Mingda C, Sebastian G, Kevin G, Piyush S, and Radu S. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv:1909.11942* (2019).

[39] Yang Z, Zihang D, Yiming Y, Jaime C, Russ R S, and Quoc V L. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* (2019), 5754–5764.