Hou SL, Huang XK, Fei CQ *et al.* A survey of text summarization approaches based on deep learning. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 36(3): 633–663 May 2021. DOI 10.1007/s11390-020-0207-x

A Survey of Text Summarization Approaches Based on Deep Learning

Sheng-Luan Hou[1,2], Xi-Kun Huang[2,3,4], Chao-Qun Fei[1,2], Shu-Han Zhang[1,2], Yang-Yang Li[3,4], Qi-Lin Sun[2,3,4] and Chuan-Qing Wang[2,3,4]

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China*

[2] *University of Chinese Academy of Sciences, Beijing 100049, China*

[3] *Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing 100190, China*

[4] *Key Laboratory of Management, Decision and Information System, Chinese Academy of Sciences, Beijing 100190, China*

E-mail: houshengluan1989@163.com; huangxikun14@mails.ucas.ac.cn; {feichaoqun, zhangshuhan}@ict.ac.cn
{liyangyang12, sunqilin13, wangchuanqing15}@mails.ucas.ac.cn

Received December 6, 2019; accepted December 24, 2020.

Abstract Automatic text summarization (ATS) has achieved impressive performance thanks to recent advances in deep learning (DL) and the availability of large-scale corpora. The key points in ATS are to estimate the salience of information and to generate coherent results. Recently, a variety of DL-based approaches have been developed for better considering these two aspects. However, there is still a lack of comprehensive literature review for DL-based ATS approaches. The aim of this paper is to comprehensively review significant DL-based approaches that have been proposed in the literature with respect to the notion of generic ATS tasks and provide a walk-through of their evolution. We first give an overview of ATS and DL. The comparisons of the datasets are also given, which are commonly used for model training, validation, and evaluation. Then we summarize single-document summarization approaches. After that, an overview of multi-document summarization approaches is given. We further analyze the performance of the popular ATS models on common datasets. Various popular approaches can be employed for different ATS tasks. Finally, we propose potential research directions in this fast-growing field. We hope this exploration can provide new insights into future research of DL-based ATS.

Keywords automatic text summarization, artificial intelligence, deep learning, attentional encoder-decoder, natural language processing

1 Introduction

Automatic text summarization (ATS) plays an increasingly important role in addressing how to acquire information and knowledge in a fast, reliable, and efficient way. ATS aims to produce a condensed representation while keeping the salient elements from one or a group of topic-related documents. ATS is a potential research area and receives considerable attention from academia to industries. For decades, various kinds of approaches have been proposed, including graph-based methods [1, 2], lexical chain based methods [3], constraint optimization based methods [4, 5], shallow machine learning based methods [6] and deep learning based methods [7–9]. However, ATS is still one of the most challenging problems because of the complexity of input document(s).

Deep learning (DL) has emerged as a powerful machine learning tool and produces state-of-the-art prediction results in many fields, such as natural language processing (NLP) and computer vision (CV) [10]. Rush *et al.* [7] first brought DL-based ATS into prominence by proposing an attention-based summarization system. After that, a variety of variants have been developed. The underlying framework for these models is usually a deep neural network that consists of an encoder and a decoder. Many researchers have further proposed additional improvements over encoder-decoder models, such as incorporating attention mechanism, copying mechanism and coverage mechanism into the sequence-to-

Survey

The work was supported by the National Key Research and Development Program of China under Grant No. 2016YFB1000902 and the National Natural Science Foundation of China under Grant Nos. 61232015, 61472412, and 61621003.

sequence learning framework, and leveraging external knowledge from text classification.

As more and more research interest focuses on DL-based ATS, an overview is needed to understand better what has been achieved in this field. To fill this gap, this work focuses on generic ATS tasks and aims to provide insights into some of the problems that inherently arise with current approaches. We compare the popular approaches for various ATS tasks. We also provide the performance analyses of most of the ATS models discussed in this paper, in terms of both ROUGE [11] and human evaluation. The main contents of this paper are summarized as follows.

- We provide a comprehensive overview of DL-based ATS methods specifically in the context of encoder-decoder models.
- The large-scale datasets used for DL-based ATS are listed and elaborated in this paper.
- Most of DL-based ATS work focuses on single-document summarization (SDS), which contains two parts: abstractive models and extractive models. We elaborate these two parts and give a comparison of them. Moreover, we summarize various DL-based multi-document summarization (MDS) approaches, including adapting methods from the models for SDS, encoder-decoder methods, and neural ranking models.
- We quantitatively compare the performance of different approaches in the context of ROUGE and human evaluation. Some excellent conclusions are obtained (see Section 7).

The remainder of this paper is structured as follows. In Section 2, we introduce preliminaries. Section 3 details the datasets commonly used in DL-based ATS methods. Section 4 introduces the DL methods for abstractive SDS. Section 5 shows the extractive SDS approaches. Section 6 details the MDS methods. Section 7 is the performance analysis. Finally, Section 8 concludes with research directions for future work.

2 Preliminaries

2.1 Abstractive and Extractive Summarization

Given one or more documents, ATS aims to produce a condensed text that captures the salient meaning of the input document(s). Let us suppose the input $\boldsymbol{X}$ consists of a sequence of m words $(\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m)$ from a fixed vocabulary $\mathcal{V}$ of size $|\mathcal{V}| = V$. A summarizer outputs a shortened sequence with length $n < m$.

Definition 1 (Abstractive Summarization). *A summarizer is abstractive if it tries to generate the optimal sequence* $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n)$, *where* $\boldsymbol{y}_i \in \mathcal{V}$.

Definition 2 (Extractive Summarization). *A summarizer is extractive if it transfers words, clauses, or sentences from the input to generate the optimal sequence* $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n)$, *where* $\boldsymbol{y}_i \in \boldsymbol{X}_{(o_1, o_2, \cdots, o_n)}$, $o_j \in \{\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m\}$.

2.2 Deep Neural Networks

Deep neural networks (DNNs) leverage sophisticated mathematical models for data processing in complex ways.

2.2.1 Feed-Forward Neural Network

A feed-forward neural network (FFNN) treats all input features as unique and independent of one another. The commonly-used situation is the end layer for probability outputs. Note that the multi-layer perceptron (MLP) is a class of FFNN, which consists of one or more hidden layers.

2.2.2 Recurrent Neural Network

Recurrent neural networks (RNNs) are a class of DNNs with cycles in them, which can be thought of as multiple copies of the same network. Supposing the input sequence is $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m)$, where $\boldsymbol{x}_t$ is the input vector at time step t, the RNN will output hidden states $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_m)$.

The Vanilla RNN has the simple network structure. At time step t, the hidden layer is $\boldsymbol{h}_t = \sigma_h(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{U}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$, and the output layer is $\boldsymbol{o}_t = \sigma_o(\boldsymbol{W}_o \boldsymbol{h}_t + \boldsymbol{b}_o)$, where σ_h and σ_0 are both activate functions.

Long Short Term Memory (LSTM) [12] was designed capable of learning long-term dependencies. The key to LSTM is the cell state, along which information flows: "forget gate" decides what information should be thrown away; "input gate" decides what new information should be stored; the "tanh" layer creates a vector of new candidate values of x_t, and then the cell state can be updated; "output gate" decides what should be outputted.

The gated recurrent unit (GRU) [13] is much simpler than LSTM, which has two gates: "update gate" controls how much information from the previous hidden state will be carried; "reset gate" is used to decide how much of the past information will be forgotten.

A bidirectional RNN consists of a forward RNN and a backward RNN, which was formulated to include the information of both preceding and following elements.

When the RNN is LSTM (GRU), the bidirectional RNN becomes BiLSTM (BiGRU).

2.2.3 Convolutional Neural Network and Graph Convolutional Network

In a convolutional neural network (CNN)[14], the hidden layers typically consist of a series of convolutional filters. After applying a filter and a max-over-time pooling operation, the significant features can be identified. Features of multiple filters are concatenated together as the final representation.

Graph convolutional networks (GCNs)[15] are a type of graph-based neural networks to calculate the probability for each node label in the graph. A GCN is modeled as a function $f(\boldsymbol{X}, \boldsymbol{A})$ to encode the graph structure, where $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ (N is the number of nodes in the graph, and C is the number of input channels) is the matrix of node features, and $\boldsymbol{A}$ is the adjacency matrix.

2.2.4 Recursive Neural Network

Recursive neural networks (ReNNs)[16] are generalizations of RNNs on tree structures. The representation of a parent node is: $\boldsymbol{p}_{l,r} = f(\boldsymbol{W}[\boldsymbol{c}_l, \boldsymbol{c}_r] + \boldsymbol{b})$, where $[\boldsymbol{c}_l, \boldsymbol{c}_r]$ denotes the concatenation of children representations. ReNNs have been successfully applied in NLP, such as sentiment analysis[16] and rhetorical parsing[17].

There are some other types of DNN. S-LSTM[18] is an extension of LSTM to tree structures, in which a memory cell can reflect the history memories of multiple child cells in a recursive process. RCNN[19] was proposed by employing a recurrent structure to capture contextual information and a max-pooling layer to capture the key components.

2.3 Encoder-Decoder Model

The encoder-decoder model[13, 20] addresses the sequence-to-sequence nature where input sequences may differ in length from output sequences. It learns to encode an input sequence $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_m)$ into a fixed-length internal representation, and decodes it into an output sequence $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n)$.

The encoder is responsible for encoding the entire sequence into a fixed-length vector $\boldsymbol{c}$. The decoder is responsible for mapping $\boldsymbol{c}$ to the target sequence. The encoder-decoder model is also called the sequence-to-sequence model, or the Seq2Seq model for short.

2.4 Attention Mechanism

The attention mechanism was proposed to pay selective attention to the inputs and relate them to the output sequence[21]. The attentional encoder-decoder model requires access to the outputs from the encoder:

$$\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_m = Encoder(\boldsymbol{X}). \quad (1)$$

In the decoder, the probability for each output $\boldsymbol{y}_t$ is: $p(\boldsymbol{y}_t|\{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{t-1}\}, \boldsymbol{X}) = g(\boldsymbol{y}_{t-1}, \boldsymbol{s}_t, \boldsymbol{c}_t)$, where $g(\cdot)$ is potentially an MLP, $\boldsymbol{s}_t$ is the hidden state in time step t, and the context vector $\boldsymbol{c}_t$ is computed as

$$\boldsymbol{c}_t = \sum_{i=1}^{m} \boldsymbol{\alpha}_{ti}\boldsymbol{h}_i,$$

where the weight $\boldsymbol{\alpha}_{ti}$ determines how much attention should be paid to that input word

$$\boldsymbol{\alpha}_{ti} = \frac{\exp(\boldsymbol{e}_{ti})}{\sum_{j=1}^{m}\exp(\boldsymbol{e}_{tj})} = \mathrm{softmax}(\exp(\boldsymbol{e}_{ti})), \quad (2)$$

where $\boldsymbol{e}_{ti} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$ is an alignment model that reflects the importance of $\boldsymbol{h}_i$ with respect to the previous hidden state $\boldsymbol{s}_{t-1}$ in generating $\boldsymbol{y}_t$. According to its different implementations, the attention mechanism has different classifications, such as additive attention vs multiplicative attention, soft attention vs hard attention, global attention vs local attention.

Additive attention[21] computes the alignment model $a(\cdot)$ using an MLP with a single hidden layer

$$a(\boldsymbol{s}_j, \boldsymbol{h}_i) = \boldsymbol{v}_a^{\mathrm{T}} \tanh(\boldsymbol{W}_a[\boldsymbol{h}_i, \boldsymbol{s}_j]), \quad (3)$$

where $\boldsymbol{v}_a^{\mathrm{T}}$ and $\boldsymbol{W}_a$ are attention parameters. Multiplicative attention[22] simplifies $a(\cdot)$ as

$$a(\boldsymbol{s}_j, \boldsymbol{h}_i) = \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{W}_a\boldsymbol{s}_j.$$

Multiplicative attention is faster as it can be implemented by using matrix multiplication.

Soft attention[21, 23] is the same as additive attention. Hard attention[23] only focuses on one hidden state of the input sequence, which is decided by treating i as an intermediate latent variable, and a multinoulli distribution is assigned. However, it is non-differentiable and is complicated to train.

Global attention[22] computes $a(\cdot)$ using three different alternatives, which are simplifications and generalizations of the above models.

$$a(\boldsymbol{s}_j, \boldsymbol{h}_i) = \begin{cases} \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{s}_j, & \text{dot},\\ \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{W}_a\boldsymbol{s}_j, & \text{general},\\ \boldsymbol{v}_a^{\mathrm{T}}\tanh(\boldsymbol{W}_a[\boldsymbol{h}_i, \boldsymbol{s}_j]), & \text{concat}. \end{cases} \quad (4)$$

Local attention [22] only focuses on a small window of the hidden states, which generates an aligned position $\boldsymbol{p}_t$ for each target word, and the context vector $\boldsymbol{c}_t$ is computed as a weighted average over the source hidden states within the window $[\boldsymbol{p}_t - D, \boldsymbol{p}_t + D]$, where D is empirically selected. Unlike (2), the attention weight here is computed as $\boldsymbol{\alpha}_{ti} = \boldsymbol{e}_{ti} \exp(-\frac{(i-\boldsymbol{p}_t)^2}{2\sigma^2})$, where $\boldsymbol{e}_{ti} = a(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$ is computed by (4).

The above attention mechanisms, in each decoding step, only look at the previous hidden state $\boldsymbol{s}_{t-1}$ and the source hidden states $\boldsymbol{H}$, may result in: 1) repeating words or phrases; 2) missing some salient parts. Self-attention and temporal attention are two attempts to tackle these problems [22].

Self-attention [12, 24], also known as intra-attention, was proposed to represent a sequence as a matrix. With the outputs of encoder, the self-attention mechanism outputs a weight matrix $\boldsymbol{A} = \text{softmax}(\boldsymbol{W}_{s2} \tanh(\boldsymbol{W}_{s1} \boldsymbol{H}^{\mathrm{T}}))$, where $\boldsymbol{W}_{s2}$ and $\boldsymbol{W}_{s1}$ are parameters, and $\boldsymbol{H} = (\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_m)$ is from (1). The resulting matrix is $\boldsymbol{M} = \boldsymbol{A}\boldsymbol{H}$. This allows the decoder to keep track of its progress and reduces the generation of repeated information.

Temporal attention [25] uses the historical information to modulate the subsequent attention. Each attention distribution is divided by the sum of the previous steps, which effectively dampens a repeating problem.

2.5 Copying Mechanism and Coverage Mechanism

To reduce the computational cost, a common approach is to limit the output vocabulary to a shortlist: only top-K most frequent words and UNK (a symbol for all other unknown words). In this sense, the UNKs are also called OOV (out-of-vocabulary) words. However, two problems will be caused: 1) some of the words in the shortlist occur less frequently in the training set and thus are difficult to learn a good representation; 2) some important information is missing by mapping different words to UNK and there exists a chance to see UNK at test time. Copying mechanism, such as pointer network or CopyNet, is a solution by choosing certain segments in the input sequence and putting them at proper places in the output sequence. Another problem for Seq2Seq models is repetition, i.e., the generated summaries tend to repeat themselves. The coverage mechanism had been proved effective for solving repetition problems [26].

2.5.1 *Pointer Network*

The pointer network [8, 27] leverages the additive attention to produce an output sequence consisting of elements from the input sequence. At time step j, the output probability is: $p(\boldsymbol{y}_j | \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_{j-1}\}, \boldsymbol{X}) = \text{softmax}(a(\boldsymbol{s}_j, \boldsymbol{h}_i))$, where $\boldsymbol{X}$ is the input sequence, $\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_n$ is a sequence of output indices, and $a(\boldsymbol{s}_j, \boldsymbol{h}_i)$ is the output of addictive attention from (3). The pointer network can also be used to tackle the problems where the size of the output dictionary is variable, such as sorting variable sized sequences.

2.5.2 *CopyNet*

CopyNet [28] predicts words based on a model with two modes: generate mode and copy mode, where the former follows the regular way of word generation and the latter picks words from the source sequence.

$$\begin{aligned} & p(\boldsymbol{y}_t | \boldsymbol{s}_t, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t, \boldsymbol{M}) \\ = & \ p(\boldsymbol{y}_t, \boldsymbol{g} | \boldsymbol{s}_t, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t, \boldsymbol{M}) + p(\boldsymbol{y}_t, \boldsymbol{c} | \boldsymbol{s}_t, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t, \boldsymbol{M}), \end{aligned}$$

where $\boldsymbol{M}$ denotes the output of the encoder (see (1)), and $\boldsymbol{g}$ denotes the generate mode while c denotes the copy mode. Due to a shared normalization term, the two modes are basically competing through a softmax function. With this approach, the UNKs can be inferred from neighboring words.

2.5.3 *Coverage Mechanism*

See *et al.* [26] adapted the coverage model of Tu *et al.* [29] for solving repetition in ATS, in which a coverage vector is maintained to keep track of the attention history. (3) is modified as

$$a(\boldsymbol{s}_j, \boldsymbol{h}_i) = \boldsymbol{v}_a^{\mathrm{T}} \tanh(\boldsymbol{W}_a [\boldsymbol{h}_i, \boldsymbol{s}_j, \boldsymbol{c}_i^j]),$$

where $\boldsymbol{c}^j = \sum_{j'=0}^{j-1} \boldsymbol{\alpha}_{j'}$ is a (unnormalized) distribution over the source document words that represents the degree of coverage.

2.6 Transformer

Transformer [30] is a parallelized attention architecture based solely on the multi-head self-attention mechanism. Essentially, attention can be described as mapping a query and a set of key-value pairs to an output. We suppose the dimension of queries and the dimension of keys are both d_k, and the dimension of each value is d_v.

2.6.1 Multi-Head Self-Attention

Multi-head attention jointly attends to information from different representation subspaces

$$\begin{aligned}&\textit{Multi-head attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})\\ &= \textit{concat}(\boldsymbol{head}_1,\boldsymbol{head}_2,\cdots,\boldsymbol{head}_h)\boldsymbol{W}^{\mathrm{O}},\end{aligned}$$

where $\boldsymbol{head}_i = \textit{Scaled dot-product attention}(\boldsymbol{QW}_i^Q, \boldsymbol{KW}_i^K, \boldsymbol{VW}_i^V)$, parameter matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{\text{model}}\times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{\text{model}}\times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{\text{model}}\times d_v}$ and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v\times d_{\text{model}}}$, where h is the number of heads and d_{model} is the dimension of input and output embedding.

$$\begin{aligned}&\textit{Scaled dot-product attention}(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V})\\ &= \text{softmax}\left(\frac{\boldsymbol{QK}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V},\end{aligned}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ are matrices packed by a set of queries, keys and values respectively. These three matrices are created from the encoder's input sequence $\boldsymbol{X}$: $\boldsymbol{Q} = \boldsymbol{XW}^Q, \boldsymbol{K} = \boldsymbol{XW}^K, \boldsymbol{V} = \boldsymbol{XW}^V$.

2.6.2 Architecture of Transformer

Transformer follows the encoder-decoder architecture. The encoder consists of a stack of six identical layers, each of which has two sub-layers. The first one uses a multi-head self-attention mechanism, and the second is an MLP. Residual connections are employed around each of the sub-layers followed by layer normalization. Different from the layers in the encoder, there is a third sub-layer in the decoder, which is designed to prevent positions from attending to subsequent positions.

To summarize, FFNNs treat features as independent (each layer has different parameters), CNNs focus on relative location and proximity, and RNNs and ReNNs both have shared memories. Attention mechanism and Transformer can grab context about an element.

2.7 Reinforcement Learning

Reinforcement learning (RL) is a way of training an agent to interact with a given environment to maximize a reward [24]. A sequential Markov decision process is considered in RL, which has been used to solve problems where the metric to optimize is not differentiable, such as ROUGE [24]. Taking extractive summarization as an example, at each time step t, the agent is in a state which includes the document and the previous extractions. The agent would take an action that decides whether a textual unit is extracted or not. After that, the agent may receive an immediate reward that shows how good the action is. The reward can also be delayed. When the agent finishes extractions from the document, it will receive a final reward that indicates the performance of the entire action sequence.

3 Datasets for ATS

Identifying large, high-quality resources for ATS has called for creative solutions: the summaries in DUC/TAC are created by linguists, news headlines are adopted as summaries of the first sentences in Gigaword, concatenated bullet points are summaries in CNN/DM, librarian archival summaries are used in NYT, etc. Table 1 shows the statistics of the popular datasets for ATS approaches based on DL. Gigaword has the biggest scale with the shortest summaries. Compared with CNN/DM, the summaries in NYT are shorter.

3.1 DUC/TAC

Document Understanding Conference (DUC)[1] is an evaluation workshop for a long-term evaluation effort in summarization from 2001. A set of English documents was published every year for participants to evaluate their summarization systems. DUC has become a summarization track of TAC since 2008.

The documents are from the news agency. Table 2 is the detailed statistics for MDS, from which we can see that there are multiple versions of ground-truth MDS summaries per cluster. For SDS, there are four datasets. In DUC'01 and DUC'02, there is a 100-word ground-truth abstractive summary for each article in a cluster. There are very short (10-word) ground-truth summaries in the DUC'03 dataset. The DUC'04 dataset has 75 bytes (roughly 14 words) ground-truth summaries.

3.2 Gigaword

Gigaword contains news articles from various news agencies. For ATS, Rush *et al.*[7] paired the headline (title) of each article with its first sentence to create a text-summary pair. Gigaword is processed by lower-casing, replacing all digit characters with special character, replacing unseen words with "UNK", and re-

[1] http://duc.nist.gov/, June 2020.

Table 1. Statistics of Large-Scale Datasets

Dataset	Avg. Length of Source	Avg. Length of Summary	#Pairs in Training Set	#Pairs in Validation Set	#Pairs in Testing Set
Gigaword [7]	31.3	8.5	3 800 000	189 000	1 951
CNN/DM [25]	≈ 800.0	≈ 60.0	286 722	13 362	11 480
NYT [31]	549.0	40.0	589 284	32 736	32 739
WikiSum [32]	>35 000.0	139.4	1 579 360	38 144	38 205
WikiCatSum [33]	800.0	≈ 100.0	≈ 162 000	≈ 9 000	≈ 9 000
Newsroom [34]	658.6	26.7	995 041	≈ 105 760	≈ 105 760

Note: "#" means "number of".

Table 2. Statistics of DUC/TAC Datasets for MDS[2]

Dataset	Output Type	#Clusters	#Documents in Total	#Ground-Truth Summary	Summary Length
DUC'01	Abstractive	60	607	3 per cluster	50, 100, 200, 400 words
DUC'02	Abstractive	59	567	128	10, 50, 100, 200 words
DUC'02	Extractive	59	567	128	200, 400 words
DUC'03	Abstractive	30	624	4 per cluster	100 words
DUC'04	Abstractive	50	10 per cluster	4 per cluster	665 bytes
DUC'05	Abstractive	50	1 593	4 per cluster	250 words
DUC'06	Abstractive	50	25 per cluster	4 per cluster	250 words
DUC'07	Abstractive	45	25 per cluster	4 per cluster	250 words
TAC'10	Abstractive	46	10 per cluster	8 per cluster	100 words
TAC'11	Abstractive	44	10 per cluster	8 per cluster	100 words

Note: "#" means "number of".

moving all duplicate pairs with the texts in the DUC dataset.

Rush *et al.* [7] provided script[3] to do the above filtering. This script can generate about 3.8 million training pairs, 189 000 validation pairs, and an equal number of testing pairs. Note that the commonly used test set is a randomly held-out 2 000 pairs.

ATS on the Gigaword dataset is also called headline generation since this task generates a headline from the first sentence of a document. However, the task is more akin to sentence paraphrasing as only the first sentence is used to predict another sentence (i.e., the headline).

3.3 CNN/DM

Nallapati *et al.* [25] proposed an abstractive summarization dataset CNN/DM by modifying a question-answering dataset [35] of news articles paired with story highlights from CNN and Daily Mail (DM). There are two versions of the CNN/DM dataset.

1) *Anonymous Version*[4] [25, 36]. For each pair, the text and summary have both been pre-processed to replace each named entity with its own unique identifier.

2) *Non-Anonymous Version*[5] [26]. The original text and summary constitute each pair.

Different from DUC/TAC, there is only one version of reference summary for each article. Table 3 is the statistics of the CNN/DM dataset [36]. In the training set, the average number of sentences (words) per document is about 30 (800), and there are about 3 or 4 sentences (60 words) in each reference summary. Note that some systems [26, 36] use CNN/DM for training, whereas others [8, 25, 36] use only the DM subset.

Table 3. Statistics of CNN/DM Dataset [25]

	#Pairs in Training Set	#Pairs in Validation Set	#Pairs in Test Set	Total Number
CNN subset	90 165	1 215	1 084	92 464
DM subset	196 557	12 147	10 396	219 100
Total	286 722	13 362	11 480	311 564

Note: "#" means "number of".

[2] http://duc.nist.gov/, June 2020.

[3] https://github.com/facebook/NAMAS, June 2020.

[4] https://github.com/deepmind/rc-data, June 2020.

[5] https://github.com/abisee/cnn-dailymail, June 2020.

3.4 NYT

The NYT dataset[31] is a large collection of articles published between 1987 and 2007 with article metadata provided by the New York Times. Most articles in the NYT dataset are manually summarized by library scientists. This collection contains over 650 000 article-summary pairs that are used for the development and evaluation of SDS algorithms.

Paulus *et al.*[24] preprocessed this dataset for ATS, which results in an average of 549 (40) words of document (summary) per document-summary pair when limiting the length of document (summary) to 800 (100) words. Note that the NYT50 test set[37] is constructed by removing the document-summary pairs whose summaries are shorter than 50 words from the NYT test set, which includes 3 452 test examples.

In the NYT dataset, the summaries are written by library scientists. Some researchers revealed that the data are somewhat biased toward extractive strategies, making it particularly useful as an extractive summarization dataset. Despite this, limited work has used this dataset for ATS[24, 38, 39].

3.5 WikiSum and WikiCatSum

WikiSum[32] was created as a large-scale dataset for MDS with hundreds of thousands of Wikipedia articles. In Wikipedia, the lead section introduces the entity (e.g., a country, or a kind of food) that the article is about, pointing out important facts associated with it. The lead section is a summary of multiple documents related to the entity. Since many articles have few citations, Google search results are taken as the supplementation. Based on this premise, the authors[32] proposed the MDS task of generating the lead section from the set of documents cited in Wikipedia articles or returned by Google (using entity names as queries). The input contains entity name (title) of a Wikipedia article, a collection of documents (cited sources of Wikipedia articles and Web search result articles from Google using the titles as queries). The output is the lead section.

The authors[40] provided code and scripts[6] to crawl documents. Liu and Lapata[40] crawled about 78.9% of the original documents since some URLs have become invalid. Table 1 includes the detailed statistics. The summaries in WikiSum are multiple sentences and sometimes multiple paragraphs, written in a fairly uniform style as encouraged by the Wikipedia manual of style. The total number of words is orders-of-magnitude larger than those of previous datasets.

WikiCatSum[33] is a domain-specific dataset that can be regarded as a subset of WikiSum. It includes about 180 000 pairs with more than five source documents. WikiCatSum includes three domains: Companies, Films, and Animals. Table 4 shows the dataset statistics. The ROUGE-1 recall denotes the average ROUGE-1 recall between the source document and the ground-truth Wikipedia lead section. Articles in WikiCatSum are associated with a set of categories by querying the DBPedia knowledge-base[7].

Table 4. Statistics of WikiCatSum Dataset[33]

Category	#Source-Target Pairs	#Topics	ROUGE-1 Recall
Company	62 545	40	55.1
Film	59 973	20	55.9
Animal	60 816	30	54.1

Note: "#" means "number of".

3.6 Newsroom

Newsroom[34] was released by the Connected Experiences Laboratory that contains 1.3 million news articles and various meta-data information, such as title and summary. The dataset is crawled and extracted from 38 major publishers. The summaries in Newsroom are written by authors and editors.

The Newsroom dataset is collected using social media and search engines. The sites are collected from Alexa.com. After excluding sites that have few or no articles with summary metadata available, a set of sites is obtained. Over 100 million HTML pages and metadata of these sites are then crawled from Archive.org, after which the newswire articles and meta-data such as title and summary are identified.

3.7 Others

LCSTS[41] is a Chinese SDS dataset, which consists of 2 400 591 Chinese short Weibo texts. The average length of source documents and target summaries is 104 and 18 respectively. XSum[42] is a novel dataset created for extreme, abstractive summarization. The task is to reduce a news article to a short, one-sentence summary. The training set contains about 204 000 article-summary pairs and the test set contains 11 000 pairs. Multi-News[43] is the first large-scale MDS news

[6] https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/data_generators/wikisum, June 2020.

[7] https://wiki.dbpedia.org/downloads-2016-10, June 2020.

dataset, which contains 56 216 articles-summary pairs. News articles and human-written summaries of them are obtained from newser.com.

4 Single-Document Abstractive Approaches

According to Definition 1, abstractive ATS may generate new words not present in the original text, which involves natural language generation techniques. Encoder-decoder models have become a popular basic architecture to solve the single-document abstractive summarization problem. With the research of attention mechanisms, the encoder-decoder models equipped with them have gained better results.

4.1 Methodologies

It is common for models developed for abstractive SDS to output a probability distribution over each word in the out vocabulary $\boldsymbol{W}$, which will be then transformed into a final sequence of words as summary. The final layer generally has one neuron with softmax activation function for each word in $\boldsymbol{W}$ and outputs a likelihood of each word being the next word in the summary.

4.1.1 ABS Model and Its Extensions

Motivated by the developments in neural machine learning, Rush *et al.*[7] proposed an attention-based summarization (ABS) model that combines a feed-forward neural language model (FFNLM) with a contextual input encoder for headline generation from one sentence. ABS utilizes an attention-based contextual encoder to construct a representation based on the input and current context. The FFNLM is parameterized as a neural network, which consists of a standard language model and the encoder term. When generating summaries, the beam search strategy[44] is leveraged as a compromise between exact and greedy decoding to reduce the search space. Moreover, the authors experimented with tuning a very small set of additional features that trade off the abstractive/extractive tendency of the system, which is referred to as ABS+. ABS is trained on the Gigaword dataset, and ABS+ is further trained on the DCU'03 dataset.

RAS (recurrent attentive summarizer)[45] is an extension of the ABS model. Improved attention models are leveraged in the RAS model. The decoder is modeled using two options: Vanilla RNN and LSTM. The convolutional attention-based encoder convolves over the full embeddings (the embedding of each word is the addition of its word embedding and position embedding) of consecutive words for attention distribution computation. The RAS model is the novel convolutional attention-based conditional RNN model.

Another extension of the ABS model is ABS+AMR[46]. AMR (abstract meaning representation) is a rooted, directed, and acyclic graph that encodes the meaning of a sentence. The information presented in AMR, such as predicate-argument structures and named entities, is used since it provides effective clues to produce summaries. The AMR for an input sentence is acquired by a transition-based AMR parser[47]. The results obtained from the AMR parser are encoded by using a modified version of attention-based Tree-LSTM encoder[48] as additional information of the ABS model.

4.1.2 Encoder-Decoder Models with Various Attention Mechanisms

In Lopyrev's model[49], two deep stacks of four LSTM units are utilized in the encoder and the encoder respectively. The decoder takes as input the hidden layers generated after feeding in the last word of the input text. Two different attention mechanisms were tested. The first one is the dot attention. The other one consists of a small set of neurons for computing the attention weights, which makes it easier to study the function of the network. Experiments on the Gigaword dataset showed the better performance of the simplified attention. Note that the "teacher forcing" strategy[50] is effective since the expected word in the actual summary is fed in when generating the next word. To overcome the disconnect between training and testing, the author in [49] randomly fed in a generated word at 10% of the time.

Chen *et al.*[51] incorporated a coverage mechanism ("distraction" in their paper) into their model. The BiGRU is adopted as the encoder. They proposed a novel attention mechanism that not only considers specific regions and content of input documents, but also distracts them to traverse between different content of a document. In the decoding process, three different types of distraction are enforced: Kullback-Leibler (KL) divergence[52] of attention weights, distraction on the attention-primed input content vector, and distraction on the hidden output vector.

The above models often generate repetitive and incoherent phrases when trained on longer documents. To mitigate this problem, Paulus *et al.*[24] proposed an intra-attention based encoder-decoder architecture.

The encoder is a BiLSTM and the decoder is a single layer LSTM. An intra-temporal attention function is used to attend over specific parts of the encoded input sequence in which multiplicative attention is adopted. The attention weights are then normalized with a temporal attention function, which penalizes the input tokens that have obtained high attention scores in past decoding steps. An intra-decoder attention mechanism is introduced to further prevent the model from attending over the same parts of the input on different decoding steps. This is the first end-to-end model for abstractive summarization on the NYT dataset.

Inspired by the graph-based approaches, Tan *et al.*[53] introduced a graph-based attention mechanism in the attentional encoder-decoder framework, in which the attention score of a sentence is determined by a hidden state graph. This hidden state graph is composed of representations of all sentences of an input document. A hierarchical encoder framework is adopted, where a word encoder encodes the words of a sentence into the sentence representation, and a sentence encoder encodes the sentences of a document into the document representation $\boldsymbol{d}$. Two different LSTMs are used as the word encoder and the sentence encoder respectively. The LSTM-based decoder receives $\boldsymbol{d}$ as the initial state and predicts the sentence representations sequentially.

A bottom-up attention[54] is proposed to better conduct summary content selection. A data-efficient content selector is employed to over-determine salient phrases in a source document. The selector decides relevant aspects of the source document, which is formulated as a word-level sequence labeling problem, to identify summary words. At inference time, the content selector computes selection probabilities for each word in a source document. The selection probabilities are used to modify the copy attention distribution to only include words identified by the selector.

Hsu *et al.*[55] proposed a unified model combining the strength of both extractive and abstractive summarization. The probability output of each sentence from the SummaRuNNer[36] model was treated as sentence-level attention. Then the word-level dynamic attention from the pointer-generator (PG) model[26] was modulated with sentence-level attention such that words in less attended sentences are less likely to be generated. A novel inconsistency loss function was proposed to penalize the inconsistency between two levels of attention.

Fan *et al.*[56] designed a user preference-oriented model to enable users to specify high-level attributes such as source style and entities of interest. An encoder-decoder model was proposed that the encoder and the decoder are both CNNs. The multi-hop attention (i.e., attention is applied at each layer of the decoder) was used to connect the encoder and the decoder. Self-attention was also adopted in the decoder to enable the model to refer back to previously generated words. To control the length, this model first quantizes summary length into discrete bins. Then the input vocabulary is expanded with special word types to indicate the length bin of the desired summary. To enable entity-centric summaries, entities are anonymized by replacing all occurrences of a given entity in a document by the same token. To enable remainder summarization, the model first aligns summaries to full documents and then matches each reference summary sentence to its best matching sentence based on ROUGE-L.

Narayan *et al.*[42] employed CNNs in the encoder-decoder model. The core of this model is a convolutional block structure that computes intermediate states on a fixed number of input words. The convolutional encoder applies this structure across the document and repeats these operations in a stacked fashion to get a multi-layer hierarchical representation. Each layer in the decoder determines useful source context by attending to the encoder representation. The output of the top layer is passed to a softmax layer to predict a distribution over the target vocabulary. In addition, the LDA topic model[57] is used to get word and document topic distributions, which is the additional input of the proposed network structure. Instead of generating multi-sentence summary, this method aims to a novel ATS task called extreme summarization: to create a short, one-sentence news summary answering the question "What is the article about?".

RC-Transformer (RCT), proposed by Cai *et al.*[58], is an extension of Transformer with an additional RNN-based encoder to capture the sequential context representations. Thus the model has two encoders and one decoder. The two encoders are capturing contextual semantic representations and modeling sequential context respectively. RCT can not only learn long-term dependencies, but also address the inherent shortcoming of Transformer. To extract salient information effectively, the authors[58] further constructed a convolution module to filter the sequential context with local importance. This model owns an advantage in speed over the classical Seq2Seq models.

4.1.3 Methods for Tackling UNKs and Repetition Problems

Gulcehre *et al.*[59] proposed a method to handle UNKs based on the idea that some of the words in the generated summary also appear in the input text. Their model can learn to point a word in the input sentence and copy it to the summary. At each time step, this model first determines whether to take a word from the predefined shortlist or to copy one from the source sentence. To achieve this goal, two softmax output layers are employed. A BiGRU is used for the encoder and a GRU is applied for the decoder. The large vocabulary trick (LVT)[60] is also introduced to reduce the size of the soft-max layer of the decoder such that the decoding can be efficiently done.

Gu *et al.*[28] used their proposed CopyNet to mitigate UNKs. A BiRNN was used as the encoder. The decoder is a vanilla RNN that reads $\boldsymbol{M}$ and predicts the summary sentences with copying and generation. They tested their model on the LCSTS dataset and achieved better results than Hu *et al.*'s model[41].

To overcome UNKs and repetition problems, See *et al.*[26] augmented the attentional encoder-decoder model with a hybrid PG network. They used a BiLSTM as the encoder and an LSTM as the decoder. In the PG network, the generation probability is calculated from the context vector, decoder state and decoder input. Then they used the coverage mechanism to keep track of what has been summarized, which discourages repetition. This approach is considerably different from Gulcehre *et al.*'s approach[59] and Nallapati *et al.*'s pointing mechanism[25] in that: 1) these methods train their pointer components to activate only for OOV words or named entities whereas See *et al.*'s method freely learns them when to use the pointer; 2) these methods do not mix the probabilities from the copy distribution and the vocabulary distribution.

4.1.4 Inspirations by Human Summarizers

Inspired by human summarizers which read the text multiple times before generating summaries, Zeng *et al.*[61] proposed a "Read-Again" model that first reads the text and then does a second read where to pay special attention to the words that are relevant to generate the summary. This is implemented by reading the input text twice and using the information acquired from the first read to bias the second read representation, and thus allows the intermediate hidden word vectors to capture the meaning appropriate for the input text. This idea can be seamlessly plugged into LSTM and GRU. Additive attention is used in this model.

DRGN (deep recurrent generative decoder)[62] was proposed as a component of attentional encoder-decoder model, which considers the latent structure information implied in the target summaries. For latent structure modeling, historical dependencies on the latent variables of VAEs were added. This model has two parts: inference (variational-encoder) and generation (variational-decoder). BiGRU was employed as the modeling component for the encoder. The discriminative deterministic decoding is an improved attention modeling based recurrent sequence decoder. Then the standard GRU-based discriminative deterministic decoder and the recurrent generative decoder were integrated into a unified decoding framework.

Inspired by human summarizers who first skimmed the document and delete unnecessary materials, Li *et al.*[63] extended the encoder-decoder framework by adding an information selection layer to model the information selection process. The information selection layer was designed as a gated global information filtering network, which consists of two parts: gated global information filtering used to remove the unnecessary information, and local sentence selection used to select salient sentences from a document sequentially to produce a summary.

Inspired by how humans summarize long documents, Chen and Bansal[64] combined extractive and abstractive summarization with RL. Salient sentences are first selected and then be rewritten. In the sentence extraction network, temporal convolutional model[14] is applied to compute the representation of each sentence. A BiLSTM is employed to further incorporate the global context of the document. Another LSTM is added to train a pointer network to extract sentences recurrently. The copying mechanism is used to help directly copy some OOV words from the input document to the generated summary.

4.1.5 Incorporating Additional Features

To address the problems that classical attentional encoder-decoder architecture often suffers, Nallapati *et al.*[25] adopted BiGRU as the encoder and GRU as the decoder. Firstly, to identify the key concepts and key entities, additional linguistic features (POS tags, named entity tags, and TF-IDF scores of the words) are captured for each word, along with original word embedding to concatenate into a single long vector as the novel word representation. Then the decoder is equipped with a switch to decide whether to use the generator or a pointer at every time-step to handle UNKs. Moreover, to capture the hierarchy of sentence-to-word structure, the attention mechanism is operated at both levels simultaneously. The same first author [25] then proposed a neural extractive approach, which will be detailed in Section 5.

To improve informativeness, Jiang *et al.* [65] incorporated topical keyword information from the original document into a PG network via a new attention mechanism. TextRank [2] was leveraged to extract topical keywords. The sum of word embeddings for d topical keywords was used as a part of input for the attention distribution. In this way, a topic-oriented summary can be generated in a context-aware manner with guidance.

Cohan *et al.* [66] took scientific papers as an example of long documents with discourse information, where their abstracts were used as ground-truth summaries. The proposed model includes a hierarchical encoder, capturing the discourse structure of the document and a discourse-aware decoder that generates the summary. To capture the discourse structure, the word-level BiLSTM encodes word sequence in a section into vector representations, whose outputs are fed into the section-level BiLSTM to get the document representation. Discourse-aware attention that uses the discourse-related information to modify the word-level attention was proposed. The copying mechanism and a coverage model were also adopted to deal with OOV words and self-repeat problems respectively. The authors crawled more than 340 000 long and structured scientific papers from arXiv.org and PubMed.com as their experimental dataset.

4.1.6 Others

To optimize the performance metric directly, Ranzato *et al.* [44] borrowed ideas from RL [67] and introduced Mixed Incremental Cross-Entropy Reinforce (MIXER). Their generative model is a conditional vanilla RNN where the conditioning vector is computed by a convolutional attentive encoder similar to the one in the ABS model. MIXER leads to significant improvements over existing supervised learning methods on the Gigaword dataset. This method requires an additional deep neural network to predict the expected reward and stabilize the objective function gradients; thus the computational costs are huge.

A deep generative auto-encoding sentence compression (ASC) model [68] was proposed to model the joint distribution of sentence-summary pairs. A discrete variational auto-encoder (VAE) was employed for inference. The objective of ASC is to perform Bayesian inference for the posterior distribution of summaries conditioned on the observed utterances. The ASC model consists of four RNNs: an encoder, a compressor, a decoder, and a language model. To further boost the performance, a supervised forced-attention sentence compression model (FSC) was presented and trained on labeled data to teach the ASC model.

Pasunuru and Bansal [69] proposed an RL approach with two novel reward functions: ROUGESal and Entail, on top of a coverage-based baseline. The ROUGESal reward is based on the ROUGE metric by upweighting the salient phrases/words detected via a keyphrase classifier. The Entail reward gives high (length-normalized) scores to logically-entailed summaries using an entailment classifier. This multi-reward approach optimizes multiple rewards simultaneously in alternate mini-batches and has achieved competitive results on the CNN/DM dataset.

Celikyilmaz *et al.* [39] presented deep communicating agents [70] in an encoder-decoder architecture to address the challenges of representing a long document. Each agent encodes a paragraph using two stacked encoders: a local encoder (a BiLSTM), whose output is fed into the contextual encoder, and a contextual encoder broadcasting their encoding to others. In this way, the agents can share global context information about different sections of the document. By passing new messages through multiple layers, the agents can coordinate and focus on the important aspects of the input text. The decoder is an LSTM in which additive attention was used over the agents to integrate information from multiple agents smoothly at each decoding step.

Liu *et al.* [71] proposed an adversarial process in which a generative model and a discriminative model were simultaneously trained. The generator was taken as an agent of RL and was implemented following See *et al.*'s PG network [26], in which the encoder is a BiLSTM, and the additive attention-based decoder is an LSTM. The discriminator attempts to distinguish the generated summary from the ground-truth summary in the training set, which was implemented as a text classifier that learns to classify the generated summaries as machine- or human-generated.

Based on the idea that the salient information should be logically entailed by the input document, Guo *et al.* [72] proposed a multi-task learning framework that incorporates the question generation and entailment generation tasks. The former task teaches the summarization model how to look for salient questioning-worthy details, and the latter task teaches the model how to rewrite the final summary. Experiments on Gigaword and CNN/DM both achieved significant improvements over the previous models.

4.2 Summary

Abstractive summarization is the ultimate goal of the ATS research, but previously it is less investigated due to the immaturity of text generation techniques. Fortunately, we have seen the emerging and development of abstractive SDS models. After the success ABS and ABS+ models [7], the attentional encoder-decoder models become the mainstream architectures. Further efforts consider many factors to improve performance. The first type of variants is improved attention mechanisms, such as fused attention [55], bottom-up attention [54], hierarchical attention [66], graph-based attention [53], intra-attention [24], and Transformer [58]. Many improvements are argumented with other features, such as additional linguistic features [25, 46, 73], inherent structures [62, 66], keyword information [65], entity information [74], and pretrained language model [75–78]. Inspired by human summarizers, there are the "Read-Again" model [61], the "Rewrite" model (which first selects salient sentences and then rewrites them abstractively) [64] and information selection model [63]. Adversarial process [68, 71] is also effective, in which a generative model and a discriminative model were simultaneously trained. To handle UNKs, Gulcehre *et al.* [59] used two softmax output layers to determine whether to take a word from the vocabulary or to copy one from the source sequence. Gu *et al.* [28] proposed CopyNet and achieved better results on the LCSTS dataset. Since then, the copying mechanism [26, 28, 59] has become an ideal solution for dealing with UNKs. Pointer-generator + Coverage model [26] is a further solution for overcoming UNKs and repetition problems, which has proved the coverage mechanism is remarkably effective for eliminating repetition problems. The coverage mechanism was further extended by Chen *et al.* [51] with a new attention mechanism. To process long documents, deep communicating agents [39] were utilized, each of which encodes a paragraph using two stacked encoders. Moreover, there are the user preference-oriented model [56], soft template-guided model [79], CNN-based Seq2Seq model [42, 80], multi-task learning model [72], etc.

For the model training, a significant number of parallel article-summary pairs are needed. During training, the loss for a document d is generally the sum of negative log-likelihood (NLL) of all target words:

$$L_d(\boldsymbol{W}, \boldsymbol{b}) = -\sum_{t=0}^{n} \log P(\boldsymbol{w}_t^*),$$

where $\boldsymbol{w}_t^*$ is the target word at time step t of decoding. The overall loss is $L(\boldsymbol{W}, \boldsymbol{b}) = \sum_{i=0}^{N} L_d(\boldsymbol{W}, \boldsymbol{b})$, where N is the number of documents in a training batch. To speed up training, the LVT [60] was often employed to reduce the size of the soft-max layer of the decoder. Once the model is trained, the process of generating a summary could be improved by using the beam search to find a reasonably good output sequence.

Despite the performance improvement of abstractive SDS approaches, one of the shortcomings of such models is that they often suffer from two common problems [24]: exposure bias [44] and inconsistency between train/test measurement. Leveraging methods from RL [24, 44, 69] has emerged in addressing these two problems. These approaches directly optimize the metric (i.e., ROUGE in the ATS task) used at test time.

5 Single-Document Extractive Approaches

Despite the emergence of abstractive methods, extractive approaches are still attractive as they are less complex, less expensive, and generate grammatically and semantically correct summaries most of the time. As defined in Definition 2, extractive approaches extract salient textual units directly from the original text. A few recent approaches conceptualize extractive SDS as a sequence classification problem.

5.1 Methodologies

5.1.1 *NLL-Based Sentence Salience Scoring*

Auto-Regressive Models. Cheng and Lapata's NN-SE model [8] is based on the encoder-decoder architecture, where the encoder learns the representation of each sentence and the entire document while the decoder classifies each sentence with the help of the pointer network. The hierarchical document reader first computes each sentence representation using a CNN over word embedding matrix, and then the document representation is obtained by an LSTM over the sentence representations. The sentence extractor uses another LSTM to label summary sentences. In the sentence extractor, the pointer network is used to apply attention to directly extract summary sentences after reading them. The ground-truth extractive training data is constructed by a rule-based system, which will be depicted in Subsection 5.2.

SummaRuNNer[36] is a two-layer BiGRU model, whose first layer is a BiGRU that runs at word level, and the second layer is another BiGRU that runs at the sentence level. The representation of the entire document is modeled as a non-linear transformation of the average pooling of the concatenated hidden states of the sentence-level layer. Each sentence is revisited sequentially to make a binary decision about whether it belongs to the summary by a sigmoid layer. The authors proposed two training strategies: 1) extractive training using an unsupervised approach to convert the abstractive summaries to extractive labels; 2) abstractive training that couples the SummaRuNNer model with an RNN decoder that models the generation of abstractive summaries at training time only.

Based on the idea that summary sentences often contain important keywords, SWAP-NET (Sentences and Words from Alternating Pointer Networks)[81] was proposed as a two-level pointer network-based architecture that models the interaction of keywords and salient sentences. In SWAP-NET, an attention-based mechanism, similar to that of pointer networks, is used to learn important words and sentences. A switch mechanism is used to select between words and sentences during decoding and the final summary is generated using a combination of selected sentences and words.

Non-Auto-Regressive Models. Based on the idea that the gist of the newswire article may lie in side information (e.g., the title or the image captions in a document), Narayan *et al.*[82] developed an attentional encoder-decoder framework composed of a hierarchical document encoder and an attention-based sentence decoder with attention over side information. The document encoder is a hierarchical one: an LSTM-based document encoder followed by a CNN-based sentence encoder. The sentence decoder is another LSTM that labels each sentence in the document by implicitly estimating its relevance in the document. The attention mechanism they used is additive attention and directly attending to the side information for importance cues, which is different from previous approaches.

Isonuma *et al.*[83] proposed a framework that identifies salient sentences from a document using external knowledge (i.e., subject) from text classification. This is a multi-task learning framework that contains two components: one for sentence extraction and the other for document classification. Document classification supports sentence extraction by learning common feature representations of salient sentences for summarization. Sentence embedding is obtained by CNN from word embedding matrix. The sentence extraction component is an LSTM-based encoder-decoder architecture. By feeding back the error of document classification to the sentence extraction component, this approach learns to extract summary sentences related to the document subject. For multi-task learning, the curriculum learning strategy was adopted.

NeuSum[84] learns to identify the relative importance of sentences using the gain over previously selected sentences. NeuSum consists of two parts: the document encoder and the sentence extractor. The document encoder has a hierarchical architecture: the BiGRU-based sentence-level encoder is based on another BiGRU-based word-level encoder. The sentence extractor reads the representation of the last extracted sentence, and then produces a new sentence extraction state and uses it to score the relative importance of the rest sentences. A GRU followed by a two-layer MLP was used to calculate the sentence salience.

Instead of using RNN or CNN, HIBERT[85] leverages hierarchical Transformer as the document encoder. The document representation is obtained by two encoders: a sentence encoder for transforming each sentence into a vector and a document encoder for learning context-sensitive sentence representations. The extractive SDS is modeled as a sequence labeling problem in which the salience of a sentence can be estimated using an additional linear projection and a softmax layer. For the model training, there are three stages. The two pre-training stages are conducted to predict a sentence using all sentences on both its left and right in a document. The fine-tuning stage is conducted to predict extractive sentence labels on the CNN/DM dataset.

BertSum[86] uses BERT for extractive summarization. To better train BERT for sentence embedding, the authors[86] modified the model by using multiple [CLS] symbols to get features for sentences ascending the symbol. Moreover, the interval segment embeddings were leveraged to distinguish multiple sentences within a document. Then several summarization-specific layers stacked on top of the BERT outputs were built to capture document-level features for extracting summaries. This model has achieved state-of-the-art results on the CNN/DM dataset.

5.1.2 RL-Based Methods

Narayan *et al.*[87] took extractive summarization as a sentence ranking task using RL. In their method, a novel training algorithm that globally optimizes the ROUGE evaluation metric through an RL objective was proposed. Their proposed summarization architecture resembles the previous models: hierarchical encoder (CNN-based sentence encoder, LSTM-based document encoder), and LSTM-based sentence extractor with softmax output. The summarization architecture is viewed as an agent that interacts with an environment consisting of input documents. The agent would be given a reward commensurate with how well the score resembles the ground-truth summary. The reward function is the mean $F1$ score of ROUGE-1, ROUGE-2, and ROGUE-L. The agent is updated using the REINFORCE algorithm[67].

RNES[88] incorporates coherence into the neural extractive model via RL. NES was first built, in which CNN is applied at the word level and BiGRU is employed at the sentence level to obtain sentence representations. The document is represented by a non-linear transformation of the mean over all sentence representations. The probability of each summary sentence is computed as an MLP. RNES learns to optimize coherence and informative importance of the summary simultaneously using the REINFORCE algorithm based on the pre-trained NES. The reward function considers both coherence score and ROUGE score.

BanditSum[89] treats extractive summarization as a contextual bandit (CB) problem. In a CB, a context (i.e., a document) is sampled and shown to the agent at each trial that the rewards (computed based on the ground-truth abstractive summary using the mean $F1$ score of ROUGE-1, ROUGE-2, and ROGUE-L) yielded by the actions (each ordered subset of a document's sentences is a different action) may depend on. Therefore, the agent can quickly learn which actions are favorable in which contexts. The model is an encoder-decoder architecture: the BiLSTM-based encoder encodes each sentence into a vector representation in the context of the document and the MLP-based decoder maps each sentence through a final sigmoid function to derive sentence salience score. The agent is also updated using the REINFORCE algorithm.

5.1.3 Others

Zhang *et al.*[90] proposed a latent variable approach, where sentences are viewed as latent variables, and sentences with activated variables are used to infer ground-truth abstractive summaries. The extractive framework is similar to the NN-SE model. The assumption is that there is a latent variable for each sentence indicating whether it should be a summary sentence. The extractive model was used to produce probability distributions for latent variables and obtain them by sampling. To avoid random label sequences during sampling, a pre-trained extractive model was used to initialize the latent model.

Inspired by the observation that a human often reads an article multiple times to fully understand and summarize its contents, ITS (iterative text summarization)[91] consists of an iteration mechanism and a selective reading module. There is one encoder, one decoder, and one iterative unit in each iteration. The iterative unit is used to update the document representation with the newly constructed sentence representations using GRU. ITS adopts positional encoding[92] as the sentence encoding method. The document representation is computed using a non-linear transformation of the average pooling of the concatenated hidden states of sentence representations. In the decoder, in each iteration, a BiGRU is used to output hidden states. The final extracting probability for each sentence is calculated by concatenating the hidden states of all decoders in all iterations and applying an MLP to them.

Xu and Durrett[38] proposed JECS (joint extractive and compressive summarizer), based on joint sentence extraction and syntactic compression. The sentence encoder encodes words using a BiLSTM and applies multiple convolution layers and max-pooling layers to extract the representation of each sentence. Similarly, the document encoder aggregates these sentence representations into a document representation with a similar BiLSTM and CNN combination. The decoding process resembles pointer network-style approaches used in NeuSum. In the sentence compression module, the ELMo[93] is employed to compute contextualized word representations. Then CNN is utilized to encode the sentence and the candidate compression. For the model training, the authors[38] constructed oracle extractive-compressive summaries, and then learned both of two components jointly with this supervision.

Kedzie *et al.*[94] investigated how content selection is performed and took the summarization task as a sequence tagging problem. This model contains two main design decisions: the selection of sentence encoder and the choice of sentence extractor. The former component maps each sentence to its vector representation whereas the latter maps a sequence of sentence embeddings to a sequence of extraction probabilities. The sentence encoder was experimented with three architectures: word averaging, RNN encoder, and CNN encoder. The sentence extractor was evaluated with four kinds: NN-SE's extractor, SummaRuNNer's extractor, BiGRU-based extractor, and Seq2Seq extractor. After their experiments on different datasets, several results have been revealed: 1) for summarization of news reports, sentence position bias dominates the learning signal; 2) for sentence embedding, simple word embedding averaging is as good as or better than RNNs or CNNs; 3) pre-trained word embeddings are better than learned embedding in most cases; 4) Seq2Seq extractor performs generally better than other extractors.

5.2 Training Strategies

To train the extractive model, ground-truth sentence-level labels are needed. However, most datasets only contain ground-truth abstractive summaries. Many training strategies were proposed based on the ground-truth abstractive summaries.

5.2.1 *Training with Automatic Generated Extractive Labels*

The intuitive solution is to convert the abstractive summaries to extractive summary sentence labels. The authors of NN-SE model [8] designed a rule-based system that determines whether a document sentence matches a highlight. The position of the sentence in the document, the unigram and bigram overlap between document sentences and highlights, and the number of entities appearing in the highlight and the sentence were taken into account. The weights of the rule were adjusted on 9 000 documents with manual sentence labels. Approximately 30% of the sentences in each document were deemed summary-worthy.

The basic idea of creating datasets for training SummaRuNNer [36] is that the selected sentences from the document should be the ones that maximize the ROUGE score with respect to ground-truth summaries. A greedy approach was adopted since finding a globally optimal subset of sentences is computationally expensive, that is, adding one sentence at a time incrementally to the summary such that the ROUGE score of the current set of selected sentences is maximized with respect to the entire ground-truth summary. Many subsequent efforts [82, 84, 85] follow this work with small changes. For instance, the dataset used in NeuSum [84] was constructed by maximizing the ROUGE-2 $F1$ score. A greedy approach was also employed.

The training of the JECS model [38] needs two types of ground-truth labels: sentence extractive labels and sentence compression labels. The former was identified using a beam search procedure similar to Maximal Marginal Relevance (MMR) [95] from the first 30 sentences. For each additional sentence to add, a heuristic cost equal to the ROUGE score of a given sentence with respect to the reference summary was computed. To get the sentence compression binary labels, for each compression option, the authors [38] assessed the value of it by comparing the ROUGE score of the sentence with and without this phrase. Any option that increases ROUGE was treated as a compression that should be applied.

NLL is the commonly used loss function. At training time, what to minimize is

$$L(\boldsymbol{W},\boldsymbol{b}) = -\sum_{d=1}^{N}\sum_{j=1}^{N_d}\Big(\boldsymbol{y}_j^d \log P(\boldsymbol{y}_j^d = 1|\boldsymbol{r}_j^d) + (1-\boldsymbol{y}_j^d)\log(1-P(\boldsymbol{y}_j^d = 1|\boldsymbol{r}_j^d))\Big),$$

where N is the number of documents in a training batch, and N_d represents the number of sentences in document d. $\boldsymbol{r}_j^d$ denotes the representations of documents and the current sentence. $\boldsymbol{y}_m^n$ denotes the binary summary label of document n's m-th sentence.

5.2.2 *Training with Abstractive Ground-Truth Directly*

The authors of the SummaRuNNer model [36] also proposed an abstractive training strategy to train the extractive model on human-generated abstractive reference summaries alone. An RNN decoder was adopted to model the generation of abstractive summaries, which uses the output of SummaRuNNer as context and equips with a softmax layer to emit a word at each time step. The NLL of the words in the reference summary was minimized as

$$L(\boldsymbol{W},\boldsymbol{b}) = -\sum_{k=1}^{N_s}\log(P(\boldsymbol{w}_k)),$$

where N_s is the number of words in the reference summary. It is worth noting that at the test time, the RNN decoder will be uncoupled and only the sentence-level extractive probabilities will be emitted.

Furthermore, in Zhang *et al.*'s Latent model [90], sentences are viewed as latent binary variables. Sentences with activated variables (i.e., 1 s) are used to infer ground-truth abstractive summaries. The latent variables are predicted with an extractive model, and the training loss comes from gold summaries directly.

5.2.3 Training with Reinforcement Learning

RL has been proposed as a way of training Seq2Seq models to directly optimize the metric used at test time, such as ROUGE in ATS.

In Narayan *et al.*'s method [87], a novel training algorithm that globally optimizes the ROUGE evaluation metric through an RL objective is proposed. The agent is initialized randomly at first. It reads a document and predicts a relevance score for each sentence. The agent is then given a reward commensurate with how well the extract resembles the gold-standard summary and is updated using the REINFORCE algorithm [67]. Since there is a large number of possible extracts, the authors approximated the expected gradient using a single sample from model output for each training example in a batch.

Different from the above method that uses an approximation of a policy gradient method to train their model, the BanditSum model [89] samples directly from the true action space, and uses exact policy gradient parameter updates. In the RNES model [88], the reward function considers both the coherence score and the ROUGE score. The ROUGE score reward is the same as the previous models. The neural coherence model can identify the appropriate next sentence to compose a coherent sentence pair, whose output is the coherence reward. Moreover, the agent is initialized by the pre-trained NES model by minimizing the NLL.

The objective of RL is to discriminate among sentences with respect to how often they occur in high scoring summaries, which is the main difference with NLL-based training. However, RL methods are known for sometimes being unstable during training. There are also many challenges in current RL research. For instance, it is often too memory-expensive to store values of each state, since the problems can be pretty complex.

5.3 Summary

Despite recent progress in abstractive SDS, extractive approaches still achieve strong performance. Most of the extractive models focus on extracting and ordering full sentences.

Attentional encoder-decoder frameworks are also favorable for extractive SDS. Hierarchical encoder is often used, one for word-level (which encodes words in each sentence), and the other for sentence-level (which aggregates the sentence representations into a document representation). Each hierarchy is an RNN or a CNN, such as CNN-based sentence encoder and LSTM-based document encoder [8, 82, 83, 87], and both the sentence encoder and the document encoder are based on the Bi-GRU model [36, 84] and the hierarchical Transformer [85]. LSTM [8, 82, 83, 87, 90], FFNN [36, 85, 88, 89], GRU followed by MLP [34, 84], and modified GRU [91] were adopted as the decoder. Moreover, pointer networks and their extensions were employed in extractive models [8, 81]. Side information [82] or subject from text classification [83] was used as external knowledge to further improve the performance. The iterative document representation approach [91] is a novel idea inspired by the observation that a human often reads an article multiple times to fully understand and summarize its contents. BERT has also been adopted into extractive summarization and has achieved good performance [86].

For the model training, NLL is the commonly used loss function. There are mainly two solutions for dealing with the absence of ground-truth extractive summaries: training with rule-based automatic generated extractive labels and training with abstractive ground-truth directly. Learning to rank sentences through an RL objective is an intuitive solution for tackling the pitfalls of training with NLL. Many RL-based methods [87–89] have been proposed. Incorporating coherence into the final reward performs better than using the ROUGE reward alone. Moreover, the self-supervised learning approach [96] outperforms previous models.

The above approaches remain extractive. One of the shortcomings is that they often generate verbose contents with unnecessarily long sentences and redundant information. Despite encouraging results, summarizing a large quantity of texts still requires sophisticated abstraction capabilities. Taking extracting salient sentences as the first step, then using techniques such as generalization, paraphrasing, and sentence fusion for further summarization may be a good solution. The JECS model [38] is a representative example.

6 Multi-Document Summarization Approaches

As the research progressed, MDS approaches emerged and were applied to clusters of news articles on the same topic, aiming at producing a summary. Neural SDS utilizing the encoder-decoder architecture has shown promising results but it did not prompt much advance on MDS. This is mainly due to:

1) the lack of large MDS datasets needed to train the computationally expensive encoder-decoder model;

2) the inadequacy of RNNs to capture the complex relations across multiple documents;

3) the natural characteristics of multiple documents about the same topic.

6.1 Methodologies

6.1.1 *Extractive Approaches*

Central to extractive MDS is the notion of similarity between sentences. To evaluate different compositions, Kågebäck *et al.*[97] proposed two sentence embedding methods based on word embedding. The first one is the simple vector addition without regarding word orders. The second one takes into account the word order and grammar in which an unfolding recursive auto-encoder[98] is used over sentence syntactic trees. For summarization, the similarity between two sentences is measured by cosine similarity and Euclidean distance. Then different combinations of word embeddings, sentence embeddings, and similarity measures are fed into Lin and Bilmes's submodular optimization summarizer[99]. Experimental results on the Opinosis dataset show that the combination of CBOW word embedding, vector addition sentence representation, and cosine similarity achieves the highest performance.

To achieve redundancy reduction, Yin and Pei[100] applied CNN to obtain sentence representation and then selected sentences by minimizing the cost based on their "prestige" and "diverseness". They first proposed CNNLM, a CNN-based network language model to project sentences into vector representations. CNNLM was trained in an unsupervised scheme, which resembles the CBOW scheme in Word2vec[101]. NCE (noise-contrastive estimation)[102] was employed to compute the cost: the model learns to discriminate between true next words and noise words. A sentence adjacent graph based on the cosine similarity was built for the next phase. And then a diversified selection process (DivSelect) to select salient sentences was constructed. DivSelect formulates the selection process as an optimization problem. In this way, the proposed DivSelect+CNNLM can produce a diversified top-K ranking list which facilitates the sentence selection for summary generation without extra steps.

Based on ReNN, Cao *et al.*[103] proposed R2N2 to formulate the sentence ranking as a hierarchical regression process. The input to R2N2 is a set of syntactic parsing trees of input documents. A projection layer is then added to transform raw features into hidden states. These hidden states are then fed into ReNN to compute all the upper node representations. Finally, different kinds of regression are conducted over the parsing tree. R2N2 contains two widely-used sentence selection methods: greedy algorithm (GA) and integer linear programming (ILP). In the GA method, the TF-IDF cosine similarity is employed to control the redundancy with the most salient sentences. The ILP-based sentence selection method considers both word and sentence scores. The dataset used in R2N2 contains DUC'01, DUC'02, and DUC'04. The R2N2 model was evaluated by three-fold validation, i.e., being trained on a two-year dataset and tested on the other year's dataset.

Later, to mitigate the problem of lacking sufficient training data and diverse categories of documents of MDS, Cao *et al.*[104] proposed TCSum, which leverages plentiful text classification data to improve the performance. Firstly, a text classification model is trained using a CNN, which projects a document into the vector representation and adds a softmax layer to predict a category of it. Then the summarization model shares the same projection procedure with the text classification model to generate document representations. TCSum transforms the document embedding to the summary embedding to maximize the match to the reference summaries. To make the summary embedding sensitive to different summary styles, the transformation matrices are computed as a category-specific according to the predicted categories. Then the sentence saliency is predicted by the cosine similarity between the summary embedding and the sentence embedding, where the sentence embedding is derived by a CNN. Finally, the sentences were ranked according to their saliency scores.

Christensen *et al.*'s G-Flow model[8] [105] demonstrated the importance of considering discourse relations among sentences in MDS. The approximate discourse graph (ADG), a multi-document discourse graph, is the core in G-Flow. ADG has proved that graphs can conveniently capture the relationships between textual units within a document. Moreover,

[8] G-Flow is an extractive MDS model, which is based on relations across sentences on the basis of indicators including discourse cues, deverbal nouns, and co-reference.

ADG can be easily constructed under the assumption that text spans represent graph nodes and edges are semantic links between them. Yasunaga *et al.* [106] extracted summary sentences in two procedures: sentence salience estimation and summary sentence selection. A sentence relation graph was built in the first procedure, where interacting sentence nodes are connected by edges. Each sentence embedding was represented as the last hidden state of GRU. Three different methods were employed to represent the relationship between two sentences (i.e., TF-IDF cosine similarity, ADG from G-Flow [105], and their proposed Personalized Discourse Graph (PDG)). In the second procedure, a GCN was applied over the sentence relation graph. The final sentence embedding s_i that incorporates the graph representation of sentence relationships was obtained through multiple layer-wise propagation. Additionally, another GRU was applied to encode the entire document cluster to an embedding C. Then the salience of each sentence was computed as $salience(s_i) = \mathrm{softmax}(\boldsymbol{v}^{\mathrm{T}} \tanh(W_1 C + W_2 s_i))$. Finally, a greedy heuristic was adopted to extract salient sentences while avoiding redundancy.

The existing efforts usually model sentence ranking (evaluate importance) and sentence selection (evaluate redundancy) in two separate processes. RASR (Redundancy-Aware Sentence Regression) [107] models sentence importance and redundancy simultaneously by directly evaluating the relative importance $f(s|S)$ of a sentence s given a set of already selected sentences S. To obtain $f(s|S)$, RASR considers the minimum relative importance of sentence s with respect to each sentence in S. The algorithm starts with the first selected sentence. A new sentence was added to the summary that results in the maximum relative increase. The algorithm terminates when the summary length constraint is reached. Two groups of features, including length, content overlap, and embedding, were employed in the training episodes. The dataset they used contains DUC'01, DUC'02, and DUC'04. The model was trained on two years' dataset and tested on the other year's dataset. However, their model of measuring the redundancy only considers the redundancy of the sentence that has the maximal score, which lacks the modeling of all the selection history.

CRSum (contextual relation-based summarization) [108] takes advantage of contextual relations among sentences to improve the performance. Based on the assumption that sentence importance also depends on contextual relations, the authors [108] first used sentence relations with a word-level attentive pooling CNN to construct sentence representations. Then, they used contextual relations with a sentence-level attentive pooling RNN to construct context representations. Finally, CRSum automatically learns useful contextual features by jointly learning representations of sentences and similarity scores between a sentence and sentences in its context. Using a two-level attention mechanism, CRSum can pay attention to important contents, i.e., words and sentences, in the surrounding context of a given sentence. The authors also evaluated the combination of CRSum with surface features (SF), such as sentence position, and TF-IDF-based features. The sentence selection was conducted using a constraint-based approach. The experimental results showed that CRSum+SF significantly outperforms CRSum. The same as RASR, the dataset includes DUC'01, DUC'02, and DUC'04. The model was trained on two years' dataset and tested on the other year's dataset.

Li *et al.* [109] proposed an unsupervised framework, which contains two parts, latent semantic modeling and salience estimation. A neural generative model called variational auto-encoders (VAEs) was employed in the first part to describe the observed sentences and their latent representations. Note that the neural variational inference was used for the posterior inference of the latent variables. In the sentence salience estimation module, an unsupervised data reconstruction framework that jointly considers the reconstruction for latent semantic space and observed term vector space was proposed. In this way, the salience of sentences from these two different and complementary vector spaces can be captured. Experimental results on the DUC/TAC dataset showed that this framework achieves better performance than the previous models.

Cho *et al.* [110] further proved that optimization-based methods can obtain competitive results when training on a relatively small-scale dataset. They proposed to adopt determinantal point processes (DPPs) to select a set of most representative sentences from the given source documents as the summary while maintaining a high diversity among summary sentences. To better capture the lexical and syntactic variations in sentences, a novel similarity measure inspired by capsule networks [111] was first proposed to measure pairwise sentence (dis)similarity. Then DPP was leveraged to obtain a set of diverse summary sentences. One of the advantages of this method is that DPP can be trained on small data.

6.1.2 Abstractive Counterparts

Wang and Ling[112] employed an encoder-decoder model to effectively produce short abstractive summaries for opinionated text. The model architecture is simple: BiLSTM-based encoder, LSTM-based decoder, and additive attention. The output is a one-sentence abstractive summary. The key point for this work is that the input consists of multiple separate text units. An importance-based sampling method was designed to allow the encoder to integrate the information from an important subset of input. The importance score was defined for each document. During training, K candidates were sampled from a multinomial distribution. Note that the model is still able to learn from more than K text units since the training process goes over the training set multiple times. Top-K candidates with the highest importance scores collapsed in descending order were taken as the input in the test step. In the word embedding, additional features such as POS tag, TF-IDF score, dependency relation were mapped into word representation via lookup tables. The authors[112] crawled two datasets: movie reviews from Rotten Tomatoes[9] and arguments from Idebate[10] on controversial topics with ground-truth abstracts, which contains about 248 000 "documents" with about 4 400 "topics", for training, validation, and testing.

For MDS, to train an encoder-decoder model with millions of parameters, Zhang *et al.*[113] adopted Tan *et al.*'s hierarchical encoder-decoder SDS framework[53] to the abstractive MDS task. Since MDS should take all the input documents as input, they proposed a different encoder model. The multi-document encoder first uses Tan *et al.*'s document encoder model[53] to encode each document into a vector representation, then takes all the document vectors in a document set as input, and produces a novel document set vector. The document set vector is the weighted sum of all document vectors whereas the weight for a document is determined based on the document itself and its contribution to the representation of the overall document set. The decoder is also a hierarchical one but with different attention mechanisms. Their proposed attention distribution was computed by conducting a topic-sensitive PageRank algorithm on all input sentences. To solve the problem that the attention distribution will be too disperse and even when the number of sentences is large, the model was restricted that only the top-K ranked sentences can have attention weights. This full model was pre-trained on the CNN/DM dataset and the decoders were tuned on the DUC/TAC dataset. The authors also revealed that neural abstractive summarization models did not transfer well on a different dataset.

PG-MMR[114] is another adaption of the encoder-decoder model to MDS. PG-MMR exploits MMR to select representative sentences from multi-documents, and leverages PG to fuse disparate sentences to an abstractive summary. PG-MMR is an iterative framework. At each iteration, PG-MMR follows the MMR principle to select the top-K source sentences, which serve as the basis for PG to generate an abstractive summary. The PG attention weights were dynamically adjusted at test time to allow the PG system to effectively focus on these K sentences. This model, which requires no MDS training data, was trained on the CNN/DM dataset with the same hyper-parameters as See *et al.*[26] did and fine-tuned on DUC/TAC datasets.

The authors of the WikiSum dataset[32] proposed a two-stage model that first coarsely extracts salient texts from source documents and then uses a decoder-only architecture (that can attend to very long sequences) to generate the final summaries. In the extractive stage, three methods (i.e., TextRank[2], TF-IDF, and SumBasic[115]) from the summarization literature, along with a trivial and cheating method, assess the importance of paragraphs. The input to the second stage is the concatenation of K best ranked paragraphs (up to 7.5k tokens) in importance order, and prefixed with the title. The Transformer architecture was modified to only consist of a decoder, which performs better in the case of longer input sequences compared with RNNs. Experimental results on the WikiSum dataset demonstrated that the decoder-only architecture can scalably attend to sequences much longer than typical encoder-decoder architectures used in sequence transduction. However, this approach still considers the multiple input documents as a concatenated flat sequence, which ignores the hierarchical structures and the relations that might exist among documents.

Liu and Lapata's abstractive MDS system[40] first ranked source paragraphs and the top-k ones serve as input to an encoder-decoder model. In paragraph ranker, an LR model was employed to determine where a paragraph should be selected as the input of the encoder-decoder model. The feature vectors are from two LSTMs (one for title and the other one for source

[9] https://www.rottentomatoes.com/, June 2020.

[10] https://idebate.org/, June 2020.

paragraph) and two max-pooling operations. Instead of treating the selected paragraphs as a very long sequence, they represented inter-paragraph relationships via several local and global Transformer layers which can be stacked freely. In the decoding procedure, beam search and length penalty [116] were adopted to generate more fluent and longer summaries. This model was trained on the WikiSum dataset.

Perez-Beltrachini *et al.* [33] proposed a neural model that is guided by the topic structure of target summaries. The topic structure contains the way that content is organized into sentences and the type of content these sentences discuss. On the WikiCatSum dataset, this model takes as input a set of ranked paragraphs that were concatenated to form a flat input sequence. The model adopts a CNN-based encoder-decoder architecture, in which the convolutional encoder [117] was used to obtain a sequence of hidden states H given an input sequence of words. A hierarchical convolutional decoder generates the target sentences based on H. To further render the document-level decoder topic-aware, the authors annotated the sentences of ground-truth summaries with topic templates and forced the model to predict them. The LDA topic model was trained to discover topic templates from summaries, especially to obtain sentence-level topic distributions by treating sentences as documents.

Based on the observation that humans tend to choose content from one or two sentences and merge them into a single summary sentence, Lebanoff *et al.* [118] proposed to rank sentence singletons and pairs together in a unified space. They exploited BERT [119] and traditional vector space models (VSMs) to characterize singletons and pairs based on the amount of summary-worthy content it conveys. The MMR principle was employed to select a set of the highest important scoring and non-redundant instances, where important score was obtained by BERT or VSM, and redundancy score was obtained by the cosine similarity between the instance and partial summary. The PG network was adopted to compress/fuse sentences into summary sentences and trained on ground-truth instances.

Sequence models perform well when parallel data are abundant. However, datasets of large, paired document-summary instances are rare. To side-step these difficulties, MeanSum [9] aims to abstractively summarize multiple product or business reviews in an unsupervised way. MeanSum consists of two main components: an auto-encoder that learns representations for each review and a summarizer that learns to generate summaries semantically similar to each of the input documents. The auto-encoder module was implemented as a Seq2Seq model to reconstruct the original reviews. The summarization component takes the outputs of the former component as input. MeanSum then re-encoded the summary and computed a similarity loss that further constrains the summary to be semantically similar to the original reviews. The final loss includes the sum of reconstruction loss and average cosine distance. MeanSum was applied to the publicly available datasets: Yelp and Amazon reviews.

The authors of the Multi-News dataset [43] proposed an end-to-end hierarchical MMR-attention pointer-generator (Hi-MAP) model for abstractive MDS. The PG network [26] was expanded as a hierarchical network, in which the sentence-level MMR ranking scores [95] were integrated to adapt the attention weights to the word level. This model performs competitively on the Multi-News dataset and the DUC'04 dataset with regard to ROUGE scores.

6.2 Summary

Compared with SDS, MDS received less attention since the paucity of suitable data for the application of DL. The same as SDS methods, MDS approaches can also be categorized into extractive methods and abstractive counterparts. Existing extraction-based systems were mostly implemented by developing a selection model to choose sentences from a candidate set.

Kågebäck *et al.* [97], and Yin and Pei [100] mapped sentences to a continuous vector space which is used for similarity measurement to reduce the redundancy in the generated summaries. Generally, extractive MDS has the following two components, which are often modeled in two separate processes.

1) *Sentence Scoring/Ranking.* Sentence scoring/ranking gives an informative score to every sentence to measure its importance.

2) *Sentence Selection.* This component selects sentences to generate a summary based on the ranked sentences in the context of redundancy.

In extractive MDS, one critical issue is to represent the semantic meanings of the sentences and documents. Word vector addition [97], CNN [100, 104, 108], ReNN [103], GRU [106] and VAEs [109] were used for sentence embedding, among which CNN is the intuitive and mostly adopted one since it has the ability to extract salient n-gram features. On the other hand, sentence ranking was conducted under a single relation assumption in most extractive publications. To tackle insufficient training data, Cao *et al.*[104] leveraged knowledge from text classification. The surface features such as length and position[108], the relative importance of sentences[106–108] are also demonstrated beneficial for extractive MDS. In the training step, NLL is also a widely adopted loss function. On the other hand, optimization-based approaches[110] have also obtained competitive results.

The cost to create the ground-truth MDS dataset can be prohibitive. The existing MDS datasets, such as DUC/TAC, are too small to be used for training encoder-decoder models with millions of parameters without overfitting. Wang and Ling[112] first tried to apply the attentional encoder-decoder model for producing short abstractive summaries from multiple opinionated texts in which the multiple texts were sampled and concatenated into one "fat" text based on importance order. Then the model transfer methods[113, 114] were proposed, where a Seq2Seq model was pre-trained on the SDS dataset and fine-tuned on DUC/TAC benchmarks. However, these models do not transfer very well on a different dataset. Many efforts focus on summarizing Wikipedia articles, which model this task as a two-stage: extracting salient texts and feeding them into deep neural models to generate abstractive summaries[32, 33, 40, 118]. Besides, Hi-MAP[43] is a novel model that integrates MMR into PG architecture for MDS.

Moreover, unsupervised MDS approaches[100, 120] were proposed to mitigate the problems of lacking sufficient training data. These approaches can also be applied to other datasets directly.

7 Performance Analysis

This section shows the performance analysis of the popular ATS models on common datasets. Since full-data manual evaluation is time-consuming and unrealistic, a variety of automatic evaluation metrics have been proposed, such as METEOR[121], and BLEU[49], among which ROUGE[11] is currently the defacto standard evaluation metric. Human evaluation is an important complementation of the ROUGE metric since it is coherence-insensitive.

7.1 ROUGE

ROUGE[11] is a recall-oriented ATS evaluation method, which measures the n-gram overlap between the system generated and ground-truth summaries. ROUGE has the advantage that it can compare the system-generated summaries with one or more ground-truth summaries as the DUC/TAC dataset has. There are several alternatives of ROUGE, including ROUGE-N (n-grams), ROUGE-L (the longest common sequence), and ROUGE-SU (skip-bigrams and unigrams). As an example, the computation of ROUGE-N is[11]

$$ROUGE\text{-}N = \frac{\sum_{gram_n} Count_{\rm match}(gram_n)}{\sum_{gram_n} Count(gram_n)},$$

where $Count_{\rm match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and a ground-truth summary, and n is the length of n-gram. ROUGE-N recall (precision) can be obtained when $Count_{match}(gram_n)$ is the total number of n-grams occurring in ground-truth (candidate) summary. ROUGE-N $F1$ is their harmonic mean.

7.1.1 *SDS on Gigaword and DUC'04*

The SDS task on the Gigaword dataset is generating an abstractive headline from the first sentence of an article. The expectation is for a summary of 75 bytes (roughly 14 words). We compare the following models.

1) ABS and ABS+[7] are both the encoder-decoder models with soft attention. ABS was trained on the Gigaword dataset, and ABS+ extracts additional hand-crafted features and was further trained on the DUC'03 dataset.

2) ASC+FSC[68] draws a latent summary sentence from a background language model, and then draws the observed sentence conditioned on this latent summary.

3) Gulcehre *et al.*'s model[59] is the first attempt to handle UNKs using a pointing and copying mechanism.

4) RAS-LSTM and RAS-Elman[45] both consider words and word positions as input and use convolutional encoders to handle the source information.

5) Nallapati *et al.*'s model[25] incorporates additional linguistic features into attentional encoder-decoder models, and utilizes LVT[60] to control the vocabulary size.

6) DRGD[62] is a Seq2Seq model equipped with a deep recurrent generative model.

7) RCT[58] is an extension of Transformer with an additional RNN-based encoder.

Table 5 shows the experimental results on the Gigaword dataset of abstractive methods, from which we can see that RCT achieves the best summarization performance on all the ROUGE metrics. The above models were also evaluated on the DUC'04 dataset. Table 6 shows the ROUGE recall at 75 words on this dataset. The ABS+ model obtained slightly better results than the ABS model since it was fine-tuned on the DUC'03 dataset. The results of the ABS+AMR model indicate that the performance can be improved when incorporating syntactic and semantic features. The generative models ASC+FSC and DRGD are better than the ABS+ model. Evaluation results on both datasets show that vanilla RNN performs better than LSTM in the RAS model.

Table 5. Experimental Results (ROUGE *F*1) of Abstractive Methods on Gigaword

System	ROUGE-1	ROUGE-2	ROUGE-L
ABS [7]	29.55	11.32	26.42
ABS+ [7]	29.76	11.88	26.96
ASC+FSC [68]	34.16	15.94	31.92
Gulcehre *et al.* [59]	35.19	16.66	32.51
RAS-LSTM [45]	32.55	14.70	30.03
RAS-Elman [45]	33.78	15.97	31.15
Nallapati *et al.* [25]	35.30	16.64	32.62
DRGD [62]	36.27	17.57	33.62
RCT [58]	37.27	18.19	34.62

Table 6. Experimental Results (ROUGE Recall) of Abstractive SDS Methods on DUC'04

System	ROUGE-1	ROUGE-2	ROUGE-L
ABS [7]	26.55	7.06	22.05
ABS+ [7]	28.18	8.49	23.81
ABS+AMR [46]	28.80	7.83	23.62
RAS-Elman [45]	28.97	8.26	24.06
RAS-LSTM [45]	27.41	7.69	23.06
Nallapati *et al.* [25]	28.61	9.42	25.24
DRGD [62]	31.79	10.75	27.48
RCT [58]	33.16	14.70	30.52

7.1.2 SDS on CNN/DM and DUC'02

Both extractive and abstractive SDS models can be trained and evaluated on the CNN/DM dataset. For extractive models, at test time, picking all sentences with probability larger than 0.5 may not be an optimal strategy. Instead, most efforts pick the top 3 high probability sentences as the final summary. We list the experimental results of the following approaches.

1) Lead-3 picks the first three sentences as the summary.

2) NN-SE [8] is an encoder-decoder architecture equipped with PG network. However, this model was trained and evaluated on the DM part of the CNN/DM dataset. We only report the results on the DUC'02 dataset.

3) SummaRuNNer [36] adopts BiGRU for sentence representation and a sigmoid layer to make a binary decision, which contains two training strategies: SummaRuNNer denotes the extractively trained model, and SummaRuNNer-abs is the abstractively trained model.

4) RNES [88] is another RL-based model to optimize coherence and informative importance simultaneously.

5) Banditsum [89] treats extractive SDS as a contextual bandit problem.

6) JECS [38] is a joint extractive and compressive summarizer, which first extracts salient sentences and then compresses them using the NeuSum model.

7) Nallapati *et al.*'s abstractive SDS method [25] was described as in Table 5.

8) Tan *et al.*'s model [53] incorporates graph-based attention into the Seq2Seq model.

9) Abstract-ML [24] is an intra-attention based encoder-decoder architecture. Abstract-ML+RL was trained using RL.

10) Chen and Bansal [64] combined extractive and abstractive summarization with RL, which first selects salient sentences and then rewrites them abstractively.

11) Fan *et al.*'s model [56] is a user preference-oriented model. The control variables were set on the general settings without user input.

12) ITS [91] is an iterative document representation method with polishing for SDS. The same as NN-SE, this model was trained and evaluated only on the CNN (DM) part of the CNN/DM dataset. We only report the results on the DUC'02 dataset.

Table 7 shows the results of different systems on the CNN/DM test set (anonymized version). We can conclude that the Lead-3 is a strong baseline, mainly due to how newswire articles are written. Generally, extractive models perform better than abstractive counterparts. JECS has achieved high ROUGE scores among extractive summarization models. Some models trained on the CNN/DM dataset (DM part for NN-SE model) were also tested on the DUC'02 dataset. Following the official guidelines, Table 8 shows the ROUGE recall scores at 75 words on the DUC'02 dataset. The abstractive training strategy performs better on the CNN/DM dataset, but the performance is not better when testing on the DUC'02 dataset. The NN-SE model obtained the highest score among these five systems.

Table 7. Experimental Results (ROUGE $F1$) on CNN/DM Dataset (Anonymized Version)

System	Extraction Mode	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	Extractive	39.20	15.70	35.50
SummaRuNNer-abs [36]	Extractive	37.50	14.50	33.40
SummaRuNNer [36]	Extractive	39.60	16.20	35.30
RNES [88]	Extractive	41.25	18.87	37.75
Banditsum [89]	Extractive	41.50	18.70	37.60
JECS [38]	Extractive	41.70	18.50	37.90
Nallapati *et al.* [25]	Abstractive	35.46	13.30	32.65
Tan *et al.* [53]	Abstractive	38.10	13.90	34.00
Abstract-ML+RL [24]	Abstractive	39.87	15.82	36.90
Abstract-ML [24]	Abstractive	38.30	14.81	35.49
Chen and Bansal [64]	Abstractive	39.66	15.85	37.34
Fan *et al.* [56]	Abstractive	39.06	15.38	35.77

Table 8. Experimental Results (ROUGE Recall) of Extractive SDS Methods on DUC'02

System	ROUGE-1	ROUGE-2	ROUGE-L
Lead-3	43.60	21.00	40.20
NN-SE [8]	47.40	23.00	43.50
SummaRuNNer-abs [36]	44.80	21.00	41.20
SummaRuNNer [36]	46.60	23.10	43.03
ITS [91]	46.60	23.40	43.50

7.1.3 MDS on DUC'02 and DUC'04

Abstractive MDS on DUC'02 and DUC'04 aims to generate a summary from about 10 newswire articles. The difference is that the summary length of the former is 100 words while the latter is 665 bytes. We list the experimental results of the following MDS approaches.

1) GCN+GRU [106] adopts a GCN on the sentence relation graph in which the sentences are represented by GRU. The sentence salience score was computed using a softmax layer.

2) R2N2 [103] is a ranking framework using ReNN. The authors [103] experimented with two strategies to select summary sentences: greedy selection (R2N2_GA) and integer linear programming (R2N2_ILP).

3) TCSum [104] leverages text classification data to improve the performance of MDS.

4) RASR [107] models sentence importance and redundancy simultaneously by directly evaluating the relative importance.

5) CRSum [108] takes advantage of contextual relations among sentences. CRSum+SF is the combination of CRSum with surface features.

Table 9 is the ROUGE recall scores on DUC'02 and DUC'04 of abstractive MDS methods. Considering TCSum is not supplemented with any hand-crafted features, its performance is very promising. Compared with other models, the GCN+GRU model performs equivalent to TCSum. In R2N2, the ILP-based summary sentence selection achieved better results than the GA-based strategy. The performance of the CRSum+SF model proved the effectiveness of surface features in MDS.

Table 9. Experimental Results (ROUGE Recall) of MDS Methods on DUC'02 and DUC'04

System	DUC'02		DUC'04	
	ROUGE-1	ROUGE-2	ROUGE-1	ROUGE-2
GCN+GRU [106]	–	–	38.23	9.84
R2N2_GA [103]	36.84	8.52	38.16	9.52
R2N2_ILP [103]	37.96	8.88	38.78	9.86
TCSum [104]	36.90	8.61	38.27	9.66
RASR [107]	37.80	9.61	39.60	10.57
CRSum [108]	37.10	9.29	38.19	9.66
CRSum+SF [108]	38.90	10.28	39.53	10.60

7.2 Human Evaluation

One of the limitations of the ROUGE metric is coherence-insensitive [122]. To ensure robustness and assess linguistic quality, many ATS models complement ROUGE results with human evaluation on relatively small samples.

Many metrics have been proposed for human evaluation.

1) Informativeness (I) [8, 9, 55, 87] indicates how well the summary captures the important parts of the article.

2) Fluency (F) [8, 9, 32, 39, 87] indicates whether the summary is written in well-formed English. This metric was previously used in DUC'05, which contains grammaticality, non-redundancy, referential clarity, focus, and structure and coherence.

3) Conciseness (CoN) [55] indicates whether the sum-

mary is clear enough to explain everything without being redundant.

4) Readability (ReD)[24,55,64] indicates how well-written (fluent and grammatical) the summary does.

5) Relevance (ReL)[24,64] indicates how well the summary captures the important parts of the article. It is based on the summary containing salient information from the input article, etc.

6) Sentiment accuracy (SA)[9] indicates how well the sentiment of the summary agrees with the overall sentiment of the original review.

7) Compactness (CoM)[112] denotes whether a summary contains unnecessary information.

Amazon Mechanical Turk (AMT)[11] is a commonly used crowd-sourcing platform for human evaluation. The selected participants are often self-reported native or proficient English speakers. In general, 3–6 participants were asked to participate in an evaluation task.

There are generally three types of human evaluation, including summary quality scoring (SQS), system ranking (SRK), and QA-based evaluation. For SQS, participants were asked to evaluate each summary by scoring each metric with a score. The final evaluation score is the average over different participants. For example, the SQS of NN-SE model[8] was conducted on AMT platform by eliciting human judgments for 20 randomly sampled DUC'02 test documents, which evaluates the summaries with level range [1, 6] with respect to informativeness and fluency. Table 10 shows the details of SQS for different systems.

In the original paper, the SQS results of Paulus *et al.*'s model[24] show that even though RL has the highest ROUGE-1 and ROUGE-L scores, it produces the least readable summaries, which confirms that optimizing for single discrete evaluation metric such as ROUGE with RL can be detrimental to the model quality. Evaluations of Tan *et al.*'s model[53] show it can generate more informative and concise summaries, which reflects the advantage of abstractive methods over extractive methods. To some extent, the fluency scores show the good ability of the abstractive model to generate fluent and grammatical sentences.

SRK was conducted by many authors. For instance, Narayan *et al.*[82] randomly selected 20 articles from the test set of the CNN part of the CNN/DM dataset and assigned this task to five annotators. Annotators were presented with an article and summaries from four different systems: Lead, NN-SE, the ground-truth, and their proposed method, and were asked to rank the summaries from best (1st) to worst (4th). Table 11 shows the details of evaluations by SRK for different systems. SRK is an ideal complementation of ROUGE results to indicate the proposed models' performance.

Furthermore, Narayan *et al.*[87] assessed the summaries following a QA paradigm. For instance, the question is "How far did the plane descend in three minutes?" for the ground-truth summary "The plane descended 28 000 feet in three minutes." The multiple fact-based QA pairs were written for each gold summary without looking at the document. Up to 71 questions in total varying from two to six questions per gold summary were created. Five participants answered the questions of each ground-truth summary. The more the questions a system can answer, the better it is at summarizing the document as a whole. In another work[42], questions were formulated so as not to reveal answers to subsequent questions. Participants read the output

Table 10. Details of Summary Quality Scoring

System	Metric	Platform	#Raters	#Evaluation Documents
NN-SE[8]	I, F	AMT	5	20 from DUC'02
Wang and Ling[112]	I, F, CoM	AMT	5	40 from movie reviews
Paulus *et al.*[24]	ReL, ReD	AMT	5	100 from CNN/DM
Tan *et al.*[53]	I, CoN, F	AMT	3	20 from CNN/DM
Hsu *et al.*[55]	I, CoN, ReD	Their own	5	100 from CNN/DM
Celikyilmaz *et al.*[39]	F	AMT	5	100 from CNN/DM
Liu *et al.*[32]	F	AMT	3	25 from WikiSum
Liu *et al.*[71]	ReD	Their own	2	50 from CNN/DM
Zhang *et al.*[113]	F	AMT	3	30 from DUC
Narayan *et al.*[87]	I, F	AMT	5	20 from CNN/DM
MeanSum[9]	SA, I, F	AMT	2	100 from CNN/DM

[11] https://www.mturk.com/, June 2020.

Table 11. Details of Evaluations by System Ranking

System	Comparison System	#Raters	#Evaluation Documents
Narayan *et al.*[82]	Lead, NN-SE, the ground-truth	5	20 from CNN/DM
Isonuma *et al.*[83]	Lead, NN-SE, LREG	6	20 from NYTAC
Fan *et al.*[56]	Pointer-generator+Coverage	5	500 from CNN/DM
Chen and Bansal[64]	Pointer-generator+Coverage	3	100 from CNN/DM
ITS[91]	Lead, the ground-truth	5	40 from CNN/DM
NeuSum[84]	NN-SE	3	50 from CNN/DM
BanditSum[89]	SummaRuNNer	5	57 from CNN/DM
HIBERT[85]	Lead, and the ground-truth, etc.	5	20 from CNN/DM

summaries and answered the questions as best as they could without access to the gold summary. Five participants answered questions for each summary. The same paradigm was adopted by Perez-Beltrachini *et al.*[33], which was tested on the WikiCatSum dataset. In this way, the QA-based human evaluation can evaluate the summary quality concerning facts.

8 Conclusions

This work provides a systematic review of currently available DL-based ATS approaches. We started with the general definition of abstractive and extractive summarization and a brief account of the related neural models. Then we listed and analyzed the large-scale datasets that have distinguishing features. Gradually the state-of-the-art techniques which conduct abstractive SDS using deep neural models, especially attentional encoder-decoder architectures were first introduced. Some advanced techniques which perform extractive SDS were later discussed. We then focused on the techniques adopted for MDS. We described the overall framework, specific model design, typical training procedures, as well as pros and cons of such techniques. Finally, this work conducted the performance analysis on large-scale datasets. The performance analysis should be useful for researchers or users to select suitable models for practical use. We hope this brief exploration can provide new insights into future research and application of ATS.

Compared with MDS, more efforts are focusing on SDS. This is mainly because of the richer research foundation and the available large-scale datasets of SDS. These approaches mostly employ Seq2Seq models where the document is fed to an encoder network, and another network learns to decode the summary. Besides RNN, CNN also performs well in SDS[117, 123]. Several models such as adaptations from SDS models, neural ranking models have been proposed for a better performing MDS task. With more and more needs of MDS, techniques about MDS might receive increasing attention in the near future. On the other hand, abstractive models can be more concise by performing generation from scratch than extractive counterparts, but they perform poorly at content selection. Extractive models learn to extract sentences from the source documents and build summaries by concatenating the extracted sentences, which are more practical since they can guarantee the grammatical correctness of the produced summary.

DL-based ATS models are promising in terms of performance when large-scale datasets are available for training. However, many challenges still remain unsolved. We discuss several interesting research directions for future work.

1) Most of the models adopt Seq2Seq models, in which CNN or RNN is employed. In the future, more DNN extensions can be used for better modeling the semantics of documents. For instance, S-LSTM and R^{2}NN can be used to tree structures. RCNN can be used to capture contextual information and the key components in texts.

2) The methods introduced so far mostly conduct ATS using only facts observed in the original text(s). In fact, there is a wide variety of external knowledge that can be incorporated to further improve the task, e.g., from rhetorical structure tree[17], lexical chain[3], and knowledge graph[124]. The investigation on incorporating additional information has just started, and might receive increasing attention in the near future.

3) The existing extractive models usually pick the most important sentences as the final summary, which may obtain incoherent or not concise results. In the future, fine-grained textual unit such as clause can be taken as the basic processing unit.

4) Most of the existing ATS methods are abstractive or extractive. With more and more specific requirements of users, other ATS tasks would be proposed.

- *User Preference Oriented Summarization.* Most of the existing methods disregard user preferences like the desired length, source-style, entities of interest, and summarize only remaining portions of a document. Fan *et al.*[56] designed a user preference-oriented abstractive summarization model.

- *Extreme Summarization.* Narayan *et al.*[42] proposed a new single-document summarization task which aims at creating a short, one-sentence news summary answering the question "What is the article about?"

Acknowledgements We are grateful to Prof. Ru-Qian Lu, a fellow of Chinese Academy of Sciences, for his encouragement. We also thank the reviewers for their valuable comments and suggestions for further improving the quality of this paper.

References

[1] Canhasi E. Graph-based models for multi-document summarization [Ph.D. Thesis]. Doktora Tezi, Ljubljana Universitesi, 2014.

[2] Mihalcea R, Tarau P. TextRank: Bringing order into text. In *Proc. the 2004 Conference on Empirical Methods in Natural Language Processing*, July 2004, pp.404-411.

[3] Hou S L, Huang Y, Fei C Q, Zhang S M, Lu R Q. Holographic lexical chain and its application in Chinese text summarization. In *Proc. the 1st Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data*, July 2017, pp.266-281. DOI: 10.1007/978-3-319-63579-8_21.

[4] Berg-Kirkpatrick T, Gillick D, Klein D. Jointly learning to extract and compress. In *Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, June 2011, pp.481-490.

[5] Gillick D, Favre B. A scalable global model for summarization. In *Proc. the NAACL HLT Workshop on Integer Linear Programming for Natural Language Processing*, June 2009, pp.10-18. DOI: 10.3115/1611638.1611640.

[6] Fattah M A. A hybrid machine learning model for multi-document summarization. *Applied Intelligence*, 2014, 40(4): 592-600. DOI: 10.1007/s10489-013-0490-0.

[7] Rush A M, Chopra S, Weston J. A neural attention model for abstractive sentence summarization. In *Proc. the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp.379-389. DOI: 10.18653/v1/d15-1044.

[8] Cheng J P, Lapata M. Neural summarization by extracting sentences and words. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.484-494. DOI: 10.18653/v1/p16-1046.

[9] Chu E, Liu P. MeanSum: A neural model for unsupervised multi-document abstractive summarization. In *Proc. the 36th International Conference on Machine Learning*, June 2019, pp.1223-1232.

[10] Young T, Hazarika D, Poria S, Cambria E. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 2018, 13(3): 55-75. DOI: 10.1109/mci.2018.2840738.

[11] Lin C. ROUGE: A package for automatic evaluation of summaries. In *Proc. the Workshop on Text Summarization Branches Out*, July 2004, pp.74-81.

[12] Cheng J P, Dong L, Lapata M. Long short-term memory networks for machine reading. In *Proc. the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.551-561. DOI: 10.18653/v1/d16-1053.

[13] Cho K, Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder decoder for statistical machine translation. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing*, October 2014, pp.1724-1734. DOI: 10.3115/v1/d14-1179.

[14] Kim Y. Convolutional neural networks for sentence classification. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing*, October 2014, pp.1746-1751. DOI: 10.3115/v1/d14-1181.

[15] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2016. https://arxiv.org/abs/1609.02907, June 2020.

[16] Socher R, Perelygin A, Wu J, Chuang J, Manning C D, Ng A, Potts C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. the 2013 Conference on Empirical Methods in Natural Language Processing*, October 2013, pp.1631-1642.

[17] Li J W, Li R M, Hovy E. Recursive deep models for discourse parsing. In *Proc. the 2014 Conference on Empirical Methods in Natural Language Processing*, October 2014, pp.2061-2069. DOI: 10.3115/v1/d14-1220.

[18] Zhu X D, Sobihani P, Guo H Y. Long short-term memory over recursive structures. In *Proc. the 32nd International Conference on Machine Learning*, July 2015, pp.1604-1612.

[19] Lai S W, Xu L H, Liu K, Zhao J. Recurrent convolutional neural networks for text classification. In *Proc. the 29th AAAI Conference on Artificial Intelligence*, January 2015, pp.2267-2273.

[20] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2014, pp.3104-3112.

[21] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[22] Luong T, Pham H, Manning C D. Effective approaches to attention-based neural machine translation. In *Proc. the 2015 Conference on Empirical Methods in Natural Language Processing*, September 2015, pp.1412-1421. DOI: 10.18653/v1/d15-1166.

[23] Xu K, Ba J, Kiros R, Cho K, Courville A, Salakhudinov R, Zemel R, Bengio Y. Show, attend and tell: Neural image caption generation with visual attention. In *Proc. the 32nd International Conference on Machine Learning*, July 2015, pp.2048-2057.

[24] Paulus R, Xiong C, Socher R. A deep reinforced model for abstractive summarization. arXiv:1705.04304, 2017. https://arxiv.org/abs/1705.04304, June 2020.

[25] Nallapati R, Zhou B W, Santos C, Gulcehre C, Xiang B. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Proc. the 20th SIGNLL Conference on Computational Natural Language Learning*, August 2016, pp.280-290. DOI: 10.18653/v1/k16-1028.

[26] See A, Liu P J, Manning C D. Get to the point: Summarization with pointer-generator networks. In *Proc. the 55th Annual Meeting of the Association for Computational Linguistics*, July 2017, pp.1073-1083. DOI: 10.18653/v1/p17-1099.

[27] Vinyals O, Fortunato M, Jaitly N. Pointer networks. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2015, pp.2692-2700.

[28] Gu J T, Lu Z D, Li H, Li V O. Incorporating copying mechanism in sequence-to-sequence learning. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.1631-1640. DOI: 10.18653/v1/p16-1154.

[29] Tu Z P, Lu Z D, Liu Y, Liu X H, Li H. Modeling coverage for neural machine translation. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.76-85. DOI: 10.18653/v1/p16-1008.

[30] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A N, Kaiser L, Polosukhin I. Attention is all you need. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2017, pp.5998-6008.

[31] Hermann K M, Kocisky T, Grefenstette E, Espeholt L, Kay W, Suleyman M, Blunsom P. Teaching machines to read and comprehend. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2015, pp.1693-1701.

[32] Nallapati R, Zhai F F, Zhou B W. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Proc. the 31st AAAI Conference on Artificial Intelligence*, February 2017, pp.3075-3081.

[33] Sandhaus E. The New York times annotated corpus. Technical Report, The New York Times Company, Research and Development, 2008. https://catalog.ldc.upenn.edu/docs/LDC2008T19/new_york_times_annotated_corpus.pdf, June 2020.

[34] Durrett G, Berg-Kirkpatrick T, Klein D. Learning-based single-document summarization with compression and anaphoricity constraints. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, August 2016, pp.1998-2008. DOI: 10.18653/v1/p16-1188.

[35] Xu J C, Durrett G. Neural extractive text summarization with syntactic compression. arXiv:1902.00863, 2019. https://arxiv.org/abs/1902.00863, June 2020.

[36] Çelikyilmaz A, Bosselut A, He X, Choi Y. Deep communicating agents for abstractive summarization. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.1662-1675. DOI: 10.18653/v1/n18-1150.

[37] Liu P J, Saleh M, Pot E, Goodrich B, Sepassi R, Kaiser L, Shazeer N. Generating Wikipedia by summarizing long sequences. In *Proc. the 6th International Conference on Learning Representations*, May 2018.

[38] Liu Y, Lapata M. Hierarchical transformers for multi-document summarization. arXiv:1905.13164, 2019. https://arxiv.org/abs/1905.13164, June 2020.

[39] Perez-Beltrachini L, Liu Y, Lapata M. Generating summaries with topic templates and structured convolutional decoders. In *Proc. the 57th Annual Meeting of the Association for Computational Linguistics*, July 2019, pp.5107-5116. DOI: 10.18653/v1/p19-1504.

[40] Grusky M, Naaman M, Artzi Y. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. arXiv:1804.11283, 2018. https://arxiv.org/abs/1804.11283, June 2020.

[41] Hu B T, Chen Q C, Zhu F Z. LCSTS: A large scale Chinese short text summarization dataset. In *Proc. the 2015 Conference on Empirical Methods in Natural Language Processing*, September 2015, pp.1967-1972. DOI: 10.18653/v1/d15-1229.

[42] Narayan S, Cohen S B, Lapata M. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.1797-1807. DOI: 10.18653/v1/d18-1206.

[43] Fabbri A R, Li I, She T W, Li S Y, Radev D R. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proc. the 57th Conference of the Association for Computational Linguistics*, July 2019, pp.1074-1084. DOI: 10.18653/v1/p19-1102.

[44] Ranzato M, Chopra S, Auli M, Zaremba W. Sequence level training with recurrent neural networks. arXiv:1511.06732, 2015. https://arxiv.org/abs/1511.06732, June 2020.

[45] Chopra S, Auli M, Rush A M. Abstractive sentence summarization with attentive recurrent neural networks. In *Proc. the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2016, pp.93-98. DOI: 10.18653/v1/n16-1012.

[46] Takase S, Suzuki J, Okazaki N, Hirao T, Nagata M. Neural headline generation on abstract meaning representation. In *Proc. the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.1054-1059. DOI: 10.18653/v1/d16-1112.

[47] Wang C, Xue N W, Pradhan S. A transition-based algorithm for AMR parsing. In *Proc. the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, May 2015, pp.366-375. DOI: 10.3115/v1/n15-1040.

[48] Tai K S, Socher R, Manning C D. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, July 2015, pp.1556-1566. DOI: 10.3115/v1/p15-1150.

[49] Lopyrev K. Generating news headlines with recurrent neural networks. arXiv:1512.01712, 2015. https://arxiv.org/abs/1512.01712, June 2020.

[50] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press, 2016.

[51] Chen Q, Zhu X D, Ling Z H, Wei S, Jiang H. Distraction-based neural networks for modeling document. In *Proc. the 25th International Joint Conference on Artificial Intelligence*, July 2016, pp.2754-2760.

[52] Inan H, Khosravi K, Socher R. Tying word vectors and word classifiers: A loss framework for language modeling. arXiv:1611.01462, 2016. https://arxiv.org/abs/1611.01462, June 2020.

[53] Tan J W, Wan X J, Xiao J G. Abstractive document summarization with a graph-based attentional neural model. In *Proc. the 55th Annual Meeting of the Association for Computational Linguistics*, July 2017, pp.1171-1181. DOI: 10.18653/v1/p17-1108.

[54] Gehrmann S, Deng Y, Rush A. Bottom-up abstractive summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.4098-4109. DOI: 10.18653/v1/d18-1443.

[55] Hsu W T, Lin C K, Lee M Y, Min K, Tang J, Sun M. A unified model for extractive and abstractive summarization using inconsistency loss. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.132-141. DOI: 10.18653/v1/p18-1013.

[56] Fan A, Grangier D, Auli M. Controllable abstractive summarization. In *Proc. the 2nd Workshop on Neural Machine Translation and Generation*, July 2018, pp.45-54. DOI: 10.18653/v1/w18-2706.

[57] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 2003, 3(1): 993-1022.

[58] Cai T, Shen M J, Peng H L, Jiang L, Dai Q. Improving transformer with sequential context representations for abstractive text summarization. In *Proc. the 8th CCF International Conference on Natural Language Processing and Chinese Computing*, October 2019, pp.512-524. DOI: 10.1007/978-3-030-32233-5_40.

[59] Gulcehre C, Ahn S, Nallapati R, Zhou B W, Bengio Y. Pointing the unknown words. In *Proc. the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp.140-149. DOI: 10.18653/v1/p16-1014.

[60] Jean S, Cho K, Memisevic R, Bengio Y. On using very large target vocabulary for neural machine translation. In *Proc. the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, July 2015, pp.1-10. DOI: 10.3115/v1/p15-1001.

[61] Zeng W Y, Luo W J, Fidler S, Urtasun R. Efficient summarization with read-again and copy mechanism. arXiv:1611.03382, 2016. https://arxiv.org/abs/1611.03382, June 2020.

[62] Li P J, Lam W, Bing L D, Wang Z H. Deep recurrent generative decoder for abstractive text summarization. In *Proc. the 2017 Conference on Empirical Methods in Natural Language Processing*, September 2017, pp.2091-2100. DOI: 10.18653/v1/d17-1222.

[63] Li W, Xiao X Y, Lyu Y J, Wang Y Z. Improving neural abstractive document summarization with explicit information selection modeling. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.1787-1796. DOI: 10.18653/v1/d18-1205.

[64] Chen Y C, Bansal M. Fast abstractive summarization with reinforce-selected sentence rewriting. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.675-686. DOI: 10.18653/v1/p18-1063.

[65] Jiang X P, Hu P, Hou L W, Wang X. Improving pointer-generator network with keywords information for Chinese abstractive summarization. In *Proc. the 7th CCF International Conference on Natural Language Processing and Chinese Computing*, August 2018, pp.464-474. DOI: 10.1007/978-3-319-99495-6_39.

[66] Cohan A, Dernoncourt F, Kim D S, Bui T, Kim S, Chang W, Goharian N. A discourse-aware attention model for abstractive summarization of long documents. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.615-621. DOI: 10.18653/v1/n18-2097.

[67] Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 1992, 8(3): 229-256. DOI: 10.1007/978-1-4615-3618-5_2.

[68] Miao Y S, Blunsom P. Language as a latent variable: Discrete generative models for sentence compression. In *Proc. the 2016 Conference on Empirical Methods in Natural Language Processing*, November 2016, pp.319-328. DOI: 10.18653/v1/d16-1031.

[69] Pasunuru R, Bansal M. Multi-reward reinforced summarization with saliency and entailment. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.646-653. DOI: 10.18653/v1/n18-2102.

[70] Sukhbaatar S, Szlam A, Fergus R. Learning multiagent communication with backpropagation. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2016, pp.2244-2252.

[71] Liu L Q, Lu Y, Yang M, Qu Q, Zhu J, Li H Y. Generative adversarial network for abstractive text summarization. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.8109-8110.

[72] Guo H, Pasunuru R, Bansal M. Soft layer-specific multi-task summarization with entailment and question generation. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.687-697. DOI: 10.18653/v1/p18-1064.

[73] Cao Z Q, Wei F R, Li W J, Li S J. Faithful to the original: Fact aware neural abstractive summarization. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.4784-4791.

[74] Amplayo R K, Lim S, Hwang S. Entity commonsense representation for neural abstractive summarization. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.697-707. DOI: 10.18653/v1/n18-1064.

[75] Kryscinski W, Paulus R, Xiong C M, Socher R. Improving abstraction in text summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.1808-1817. DOI: 10.18653/v1/d18-1207.

[76] Zhang H Y, Cai J J, Xu J J, Wang J. Pretraining-based natural language generation for text summarization. In *Proc. the 23rd Conference on Computational Natural Language Learning*, November 2019, pp.789-797. DOI: 10.18653/v1/k19-1074.

[77] Dong L, Yang N, Wang W H, Wei F R, Liu X D, Wang Y, Gao J F, Zhou M, Hon H. Unified language model pre-training for natural language understanding and generation. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2019, pp.13042-13054.

[78] Song K T, Tan X, Qin T, Lu J F, Liu T. MASS: Masked sequence to sequence pre-training for language generation. In *Proc. the 36th International Conference on Machine Learning*, June 2019, pp.5926-5936.

[79] Cao Z Q, Li W J, Li S J, Wei F R. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.152-161. DOI: 10.18653/v1/p18-1015.

[80] Wang L, Yao J L, Tao Y Z, Zhong L, Liu W, Du Q. A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. In *Proc. the 27th International Joint Conference on Artificial Intelligence*, July 2018, pp.4453-4460. DOI: 10.24963/ijcai.2018/619.

[81] Jadhav A, Rajan V. Extractive summarization with SWAP-NET: Sentences and words from alternating pointer networks. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.142-151. DOI: 10.18653/v1/p18-1014.

[82] Narayan S, Papasarantopoulos N, Cohen S B, Lapata M. Neural extractive summarization with side information. arXiv:1704.04530, 2017. https://arxiv.org/abs/1704.04530, June 2020.

[83] Isonuma M, Fujino T, Mori J, Matsuo Y, Sakata I. Extractive summarization using multi-task learning with document classification. In *Proc. the 2017 Conference on Empirical Methods in Natural Language Processing*, September 2017, pp.2101-2110. DOI: 10.18653/v1/d17-1223.

[84] Zhou Q Y, Yang N, Wei F R, Huang S H, Zhou M, Zhao T J. Neural document summarization by jointly learning to score and select sentences. In *Proc. the 56th Annual Meeting of the Association for Computational Linguistics*, July 2018, pp.654-663. DOI: 10.18653/v1/p18-1061.

[85] Zhang X X, Wei F R, Zhou M. Hibert: Document level pre-training of hierarchical bidirectional transformers for document summarization. arXiv:1905.06566, 2019. https://arxiv.org/abs/1905.06566, June 2020.

[86] Liu Y. Fine-tune BERT for extractive summarization. arXiv:1903.10318, 2019. https://arxiv.org/abs/1903.10318, June 2020.

[87] Narayan S, Cohen S B, Lapata M. Ranking sentences for extractive summarization with reinforcement learning. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.1747-1759. DOI: 10.18653/v1/n18-1158.

[88] Wu Y X, Hu B T. Learning to extract coherent summary via deep reinforcement learning. In *Proc. the 32nd AAAI Conference on Artificial Intelligence*, February 2018, pp.5602-5609.

[89] Dong Y, Shen Y K, Crawford E, van Hoof H, Cheung J C K. BanditSum: Extractive summarization as a contextual bandit. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.3739-3748. DOI: 10.18653/v1/d18-1409.

[90] Zhang X X, Lapata M, Wei F R, Zhou M. Neural latent extractive document summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.779-784. DOI: 10.18653/v1/d18-1088.

[91] Chen X Y, Gao S, Tao C Y, Song Y, Zhao D Y, Yan R. Iterative document representation learning towards summarization with polishing. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.4088-4097. DOI: 10.18653/v1/d18-1442.

[92] Sukhbaatar S, Weston J, Fergus R *et al.* End-to-end memory networks. In *Proc. the Annual Conference on Neural Information Processing Systems*, December 2015, pp.2440-2448.

[93] Peters M E, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, Zettlemoyer L. Deep contextualized word representations. In *Proc. the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2018, pp.2227-2237.

[94] Kedzie C, McKeown K, Daume III H. Content selection in deep learning models of summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.1818-1828. DOI: 10.18653/v1/d18-1208.

[95] Carbonell J G, Goldstein J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 1998, pp.335-336. DOI: 10.1145/290941.291025.

[96] Wang H, Wang X, Xiong W H, Yu M, Guo X X, Chang S Y, Wang W Y. Self-supervised learning for contextualized extractive summarization. In *Proc. the 57th Conference of the Association for Computational Linguistics*, July 2019, pp.2221-2227. DOI: 10.18653/v1/p19-1214.

[97] Kågebäck M, Mogren O, Tahmasebi N, Dubhashi D. Extractive summarization using continuous vector space models. In *Proc. the 2nd Workshop on Continuous Vector Space Models and their Compositionality*, April 2014, pp.31-39. DOI: 10.3115/v1/w14-1504.

[98] Socher R, Huang E H, Pennington J, Ng A Y, Manning C D. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. 25th Annual Conference on Neural Information Processing Systems*, December 2011, pp.801-809.

[99] Lin H, Bilmes J. A class of submodular functions for document summarization. In *Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, June 2011, pp.510-520.

[100] Yin W P, Pei Y L. Optimizing sentence modeling and selection for document summarization. In *Proc. the 24th International Joint Conference on Artificial Intelligence*, July 2015, pp.1383-1389.

[101] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781, 2013. https://arxiv.org/abs/1301.3781, June 2020.

[102] Mnih A, Teh Y. A fast and simple algorithm for training neural probabilistic language models. In *Proc. the 29th International Conference on Machine Learning*, June 2012.

[103] Cao Z Q, Wei F R, Dong L, Li S J, Zhou M. Ranking with recursive neural networks and its application to multi-document summarization. In *Proc. the 29th AAAI Conference on Artificial Intelligence*, January 2015, pp.2153-2159.

[104] Cao Z Q, Li W J, Li S J, Wei F R. Improving multi-document summarization via text classification. In *Proc. the 31st AAAI Conference on Artificial Intelligence*, February 2017, pp.3053-3059.

[105] Christensen J, Soderland S, Etzioni O. Towards coherent multi-document summarization. In *Proc. the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2013, pp.1163-1173.

[106] Yasunaga M, Zhang R, Meelu K, Pareek A, Srinivasan K, Radev D. Graph-based neural multi-document summarization. In *Proc. the 21st Conference on Computational Natural Language Learning*, August 2017, pp.452-462. DOI: 10.18653/v1/k17-1045.

[107] Ren P J, Wei F R, Chen Z M, Ma J, Zhou M. A redundancy-aware sentence regression framework for extractive summarization. In *Proc. the 26th International Conference on Computational Linguistics*, December 2016, pp.33-43.

[108] Ren P J, Chen Z M, Ren Z C, Wei F R, Ma J, de Rijke M. Leveraging contextual sentence relations for extractive summarization using a neural attention model. In *Proc. the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, August 2017, pp.95-104. DOI: 10.1145/3077136.3080792.

[109] Li P J, Wang Z H, Lam W, Ren Z C, Bing L D. Salience estimation via variational auto-encoders for multi-document summarization. In *Proc. the 31st AAAI Conference on Artificial Intelligence*, February 2017, pp.3497-3503.

[110] Cho S, Lebanoff L, Foroosh H, Liu F. Improving the similarity measure of determinantal point processes for extractive multi-document summarization. In *Proc. the 57th Conference of the Association for Computational Linguistics*, July 2019, pp.1027-1038. DOI: 10.18653/v1/p19-1098.

[111] Hinton G E, Sabour S, Frosst N. Matrix capsules with EM routing. In *Proc. the 6th International Conference on Learning Representations*, April 2018.

[112] Wang L, Ling W. Neural network-based abstract generation for opinions and arguments. In *Proc. the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2016, pp.47-57. DOI: 10.18653/v1/n16-1007.

[113] Zhang J M, Tan J W, Wan X J. Adapting neural single-document summarization model for abstractive multi-document summarization: A pilot study. In *Proc. the 11th International Conference on Natural Language Generation*, November 2018, pp.381-390. DOI: 10.18653/v1/w18-6545.

[114] Lebanoff L, Song K, Liu F. Adapting the neural encoder-decoder framework from single to multi-document summarization. In *Proc. the 2018 Conference on Empirical Methods in Natural Language Processing*, October 2018, pp.4131-4141. DOI: 10.18653/v1/d18-1446.

[115] Nenkova A, Vanderwende L. The impact of frequency on summarization. Technical Report, Microsoft Research, 2005. https://www.cs.bgu.ac.il/~elhadad/nlp09/sumbasic.pdf, June 2020.

[116] Wu Y H, Schuster M, Chen Z F *et al.* Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv:1609.08144, 2016. https://arxiv.org/abs/1609.08144, June 2020.

[117] Gehring J, Auli M, Grangier D, Yarats D, Dauphin Y N. Convolutional sequence to sequence learning. In *Proc. the 34th International Conference on Machine Learning*, August 2017, pp.1243-1252.

[118] Lebanoff L, Song K, Dernoncourt F, Kim D S, Kim S, Chang W, Liu F. Scoring sentence singletons and pairs for abstractive summarization. arXiv:1906.00077, 2019. https://arxiv.org/abs/1906.00077, June 2020.

[119] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2019, pp.4171-4186.

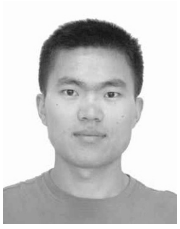
[120] Chu E, Liu P J. Unsupervised neural multi-document abstractive summarization. arXiv:1810.05739, 2018. https://arxiv.org/abs/1810.05739, June 2020.

[121] Denkowski M, Lavie A. Meteor universal: Language specific translation evaluation for any target language. In *Proc. the 9th Workshop on Statistical Machine Translation*, June 2014, pp.376-380. DOI: 10.3115/v1/w14-3348.

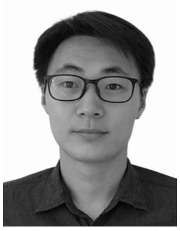
[122] Schluter N. The limits of automatic summarisation according to ROUGE. In *Proc. the 15th Conference of the European Chapter of the Association for Computational Linguistics*, April 2017, pp.41-45. DOI: 10.18653/v1/e17-2007.

[123] Zhong M, Liu P F, Wang D Q, Qiu X P, Huang X J. Searching for effective neural extractive summarization: What works and what's next. In *Proc. the 57th Conference of the Association for Computational Linguistics*, July 2019, pp.1049-1058. DOI: 10.18653/v1/p19-1100.

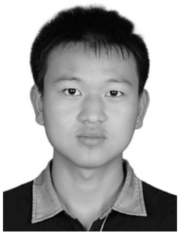
[124] Lu R Q, Jin X L, Zhang S M, Qiu M K, Wu X D. A study on big knowledge and its engineering issues. *IEEE Transactions on Knowledge and Data Engineering*, 2018, 31(9): 1630-1644.



Sheng-Luan Hou received his M.S. degree in mathematics from Beijing University of Technology, Beijing, in 2014. He is now a Ph.D. candidate in Institute of Computing Technology (ICT), Chinese Academy of Sciences, Beijing. His research interests include deep learning, automatic text summarization, and artificial intelligence.



Xi-Kun Huang received his B.S. degree in mathematics from Jilin University, Changchun, in 2014. He is now a Ph.D. candidate in Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing. His current research interests include complex network evolving model, link prediction in dynamic networks, and network representation learning.



Chao-Qun Fei received his M.S. degree in computer science from Wuhan University, Wuhan, in 2015. He is now a Ph.D. candidate in Institute of Computing Technology (ICT), Chinese Academy of Sciences, Beijing. His research interests include knowledge graph, natural language processing, and machine learning.

Shu-Han Zhang received her M.S. degree in computer science from Yunnan Normal University, Kunming, in 2015. She is now a Ph.D. candidate in Institute of Computing Technology (ICT), Chinese Academy of Sciences, Beijing. Her research interests include petri net, data mining, and artificial intelligence.

Yang-Yang Li received her B.E. degree in mathematics and computer science from Hebei University, Baoding, in 2012. She is currently pursuing her Ph.D. degree in Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing. Her research interests include pattern recognition, image processing, and machine learning.

Qi-Lin Sun received his B.S. degree from Dalian University of Technology, Dalian, in 2013. He is now a Ph.D. candidate in Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing. His research interests include knowledge graph, natural language processing, and machine learning.

Chuan-Qing Wang received his B.S. degree in mathematics from Henan University, Kaifeng, in 2015. He is now a successive M.S.-Ph.D. candidate in Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing. His research interests include knowledge-based programming and natural language processing.