Neural Abstractive Summarization for Long Text and Multiple Tables

Shuaiqi Liu, Jiannong Cao, *Fellow, IEEE*, Zhongfen Deng, *Member, IEEE*, Wenting Zhao, Ruosong Yang, Zhiyuan Wen, and Philip S. Yu, *Fellow, IEEE*

***Abstract*—Abstractive summarization aims to generate a concise summary covering the input document's salient information. Within a report document, the salient information can be scattered in the textual and non-textual content. However, existing document summarization datasets and methods usually focus on the text and filter out the non-textual content. Missing tabular data can limit produced summaries' informativeness, especially when summaries require covering quantitative descriptions of critical metrics in tables. Existing datasets and methods cannot meet the requirements of summarizing long text and dozens of tables in each report document. To deal with the scarcity of available datasets, we propose FINDSum, the first large-scale dataset for long text and multi-table summarization. Built on 21,125 annual reports from 3,794 companies, FINDSum has two subsets for summarizing each company's results of operations and liquidity. Besides, we present four types of summarization methods to jointly consider text and table content when summarizing reports. Additionally, we propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. Our summarization methods significantly outperform advanced baselines, which verifies the necessity of incorporating textual and tabular data when summarizing report documents. We also conduct extensive comparative experiments to identify vital model components and configurations that can improve summarization results.**

***Index Terms*—Document summarization, natural language generation, natural language processing, text summarization.**

I. Introduction

REPORT documents, like financial reports, investigative reports, and technical reports, are essential information sources. These report documents usually contain large amounts of textual and tabular content and provide rich knowledge about companies, industries, technologies, etc. Each report's salient information can be scattered in long text and multiple tables in different sections, which makes it difficult for non-specialized readers to efficiently read these report documents. A high-quality summary of each report document can help readers quickly browse key information. Automatic document summarization techniques can be utilized to produce reports' summaries. Users can flexibly adjust the input document and immediately get a summary from the automatic summarization system. Our target is to let the computer generate an informative, fluent, and non-redundant summary for the long text and multiple tables in each report document. To achieve this target, we need to deal with some challenging issues: 1) the scarcity of available datasets, 2) identifying the salient information scattered in a large amount of input content, 3) incorporating different types of content when generating summaries, and 4) models' efficiency in processing long inputs and outputs.

Previous document summarization datasets usually focus on text. Non-textual content is usually regarded as noises and filtered out. When target summaries only focus on narratives and qualitative descriptions, removing non-textual content has little effect since the document's text already contains most of the required information. When it comes to report documents, like financial reports, their summaries should cover both the narrative content and quantitative descriptions of critical metrics recorded in tables, which are essential for readers' analysis and decision-making [1]. Existing datasets cannot meet the requirements of summarizing long text and multiple tables in each report document.

To deal with the scarcity of available datasets, we propose FINDSum, the first large-scale dataset for long text and multi-table summarization.[1] FINDSum has two subsets named FINDSum-ROO and FINDSum-Liquidity for summarizing companies' results of operations and liquidity. Inputs of each example in FINDSum include tens of thousands of words and dozens of tables from a report document. Table I shows that FINDSum's target summaries usually contain more numerical values than previous datasets. Meanwhile, most numerical values in target summaries cannot be found in the corresponding input text. Only focusing on text is not enough to summarize these financial reports.

We propose a solution for long text and multi-table summarization to cope with the other three issues. As shown in Fig. 1, our solution has three main steps: data pre-processing,

Manuscript received 6 February 2023; revised 30 August 2023; accepted 30 September 2023. Date of publication 13 October 2023; date of current version 19 April 2024. This work is supported in part by the Research Institute for Artificial Intelligence of Things at PolyU, the Hong Kong Jockey Club Charities Trust under Project 2021-0369, Hong Kong RGC Theme-Based Research Scheme T41-603/20-R and Research Impact Fund R5034-18, and in part by NSF under Grants III-2106758, III-1763325, III-1909323, and SaTC-1930941. Recommended for acceptance by M.A. Cheema. *(Corresponding authors: Shuaiqi Liu; Philip S. Yu.)*

Shuaiqi Liu, Jiannong Cao, Ruosong Yang, and Zhiyuan Wen are with Hong Kong Polytechnic University, Hong Kong, China (e-mail: cssqliu@comp.polyu.edu.hk; csjcao@comp.polyu.edu.hk; rsong.yang@polyu.edu.hk; cszwen@comp.polyu.edu.hk).

Zhongfen Deng, Wenting Zhao, and Philip S. Yu are with the University of Illinois Chicago, Chicago, IL 60607 USA (e-mail: zdeng21@uic.edu; wzhao41@uic.edu; psyu@uic.edu).

Digital Object Identifier 10.1109/TKDE.2023.3324012

[1]FINDSum dataset is available for download online at: https://github.com/StevenLau6/FINDSum

TABLE I
STATISTICAL INFORMATION OF SUMMARIZATION DATASETS

Dataset	Pairs	Words (Doc)	Sents (Doc)	Words (Sum)	Sents (Sum)	Numbers (Sum)	% Covered Numbers	Cov.	Dens.
CNN/DM	312,085	810.6	39.8	56.2	3.7	0.6	78.7	0.9	3.8
PubMed	133,215	3049.0	87.5	202.4	6.8	3.3	68.2	0.8	5.8
arXiv	215,913	6029.9	205.7	272.7	9.6	0.7	53.9	0.9	3.8
FINDSum-ROO	21,125	45,566.0	1250.5	660.7	16.3	**24.3**	**26.3**	0.9	9.7
FINDSum-Liquidity	21,125	45,566.0	1250.5	1,057.6	26.7	**32.3**	**41.2**	0.9	9.6

"Pairs" is the number of examples. "Words" and "Sents" denote the average number of words and sentences in input text or target summary. "Numbers" is the average number of numerical values in target summaries, and "Covered numbers" is the ratio of the target summary's numerical values found in the input text. "Cov." and "Dens." are the extractive fragment's coverage and density [6].

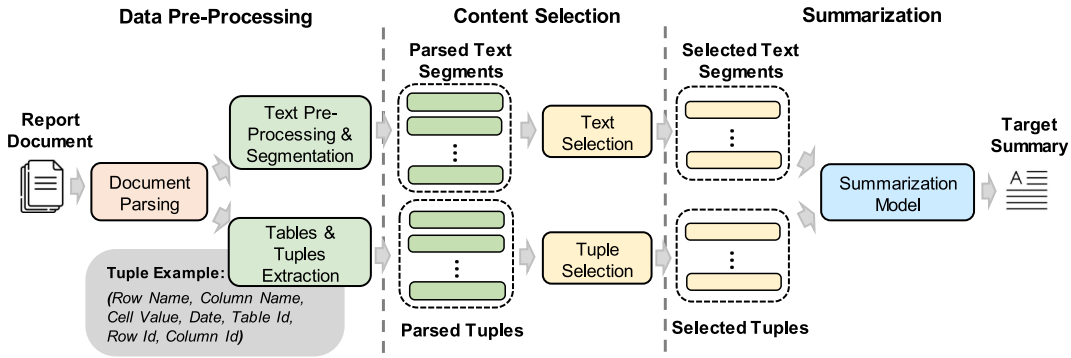


Fig. 1. Overview of our solution for long text and multi-table summarization.

content selection, and summarization. To efficiently identify the scattered key information, we add the content selection step as a rough selection over the long inputs, and then the summarization step conducts a finer selection. The content selection step aims to compress long inputs while maximizing the recall of salient content in long text and dozens of tables. Specifically, we adopt the Maximum Marginal Recall Gain (MMRG) method to select salient text segments. As for the tabular content, we transform each table cell into a tuple and regard the salient tuple selection as a binary classification problem. The summarization step should jointly consider different types of inputs. To incorporate text and tabular content, we present four types of summarization methods: generate-then-combine (GC), combine-then-generate (CG), generate-template-then-fill (GTF), and generate-combine-then-generate (GCG).

The complexity of the transformer's self-attention mechanism scales quadratically with the input length [2], which limits transformer-based models' efficiency. Thus we employ content selection methods and sparse attention mechanisms to reduce the complexity and enable fine-tuning large pre-trained models over long inputs on an off-the-shelf GPU. Besides, existing autoregressive models still have difficulty in generating long sequences [3], [4]. We employ a divide-and-conquer approach to generate summary segments in parallel and then merge them as the final summary.

We benchmark advanced extractive and abstractive summarizers as baselines on our FINDSum dataset. To compare their performance, we conduct automatic evaluation and human evaluation. In addition to the commonly used ROUGE scores [5], we propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. Experimental results show that our methods can outperform competitive baselines.

We also conduct extensive comparative experiments and a case study to compare and analyze the influence of model components and configurations on summarization results. We find the input sequence length, content selection methods, divide-and-conquer method, sparse attention mechanism, and pre-trained model can greatly affect summarization results. Experimental results also verify the effectiveness of our methods in content selection and summarization.

Our contribution is fourfold:

- We build FINDSum, the first large-scale dataset for long text and multi-table summarization.
- We present and compare four types of methods incorporating text and tables into summary generation.
- We propose evaluation metrics to assess the usage of numerical information in generated summaries.
- We find vital components and configurations of models that improve summarization results.

II. RELATED WORK

A. Automatic Document Summarization

Automatic document summarization techniques can produce concise summaries covering input documents' salient information. In recent years, both large-scale summarization datasets and advanced neural models boosted improvements in produced summaries' quality. Except for the widely studied news

summarization [6], [7], summarizing long documents received more attention. There are some datasets collected from different domains, including scientific literature [8], [9], government reports [10], and books [11]. The Financial Narrative Summarisation shared task in 2020 [12] delivered an annual report dataset from firms listed on the London Stock Exchange. These datasets only focus on the text, regard tabular data as noises, and filter them out. Previous summarization methods can be generally classified into two categories: extractive [13], [14] and abstractive methods [15], [16]. To model long inputs with limited GPU memory, Huang et al. [10] compare various efficient attention mechanisms for abstractive summarization. Liu et al. [9] identify and encode salient content in different aspects from diverse and long inputs by category-based alignment and sparse attention. However, these summarization methods only focus on text and neglect input tabular data.

B. Table Question Answering

Question answering aims to answer natural language questions based on the context (e.g., text, table, knowledge base, etc.). There are many datasets and methods for QA over tabular data. Some pre-trained models for tabular data achieved good performance. TAPAS is pre-trained from BERT with a maximum input length of 512. Its embedding setting cannot fit dozens of differently shaped input tables in each example of FINDSum. TAPEX is built on BART and uses special tokens to indicate the region of table headers and rows. We try TAPEX as a component in our GTF methods. Besides, some QA work uses both text and tables to answer questions [17], [18], [19]. QA tasks are essentially different from general summarization tasks. QA has hints from questions and just needs to find matching answers. In contrast, summarization tasks usually have no clear hints on what to look for and require the model to comprehensively identify and summarize the key content in longer inputs.

C. Table-to-Text Generation

There are some table summarization or table-to-text generation datasets, like the WEATHERGOV [20], WikiBio [21], and ROTOWIRE [22]. Some advanced methods, like hierarchical-encoder [23], macro-plan [24], and LATTICE [25], perform well on these datasets. However, existing datasets and methods are usually limited to generating short descriptions for limited cells in a few tables with similar schemas. Conversely, financial reports usually contain numerous cells in dozens of differently shaped tables. Selecting salient ones from thousands of cells can be challenging. In addition to summarizing multiple tables, we observe that human-written summaries can combine the information from both the text and multiple tables within the report document. Unstructured text and structured tabular data have different natures. It is also challenging to effectively integrate different types of input data in the summary generation. To fill in the gap between the limitations of existing work and the actual requirements of long text and multi-table summarization, we propose the FINDSum dataset and four types of summarization methods.

III. FINDSum Dataset

Financial report document summarization (FINDSum) is the first large-scale dataset for long text and multi-table summarization. This section introduces our data collection and pre-processing procedures and describes FINDSum's two subsets. We conduct descriptive statistics and in-depth analysis on FINDSum and compare it with other datasets.

A. Data Collection and Pre-Processing

Form 10-K is the annual report that comprehensively describes a company's financial performance in the prior fiscal year [1]. We collected thousands of companies' last ten 10-K forms' HTML files from the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system.[2] The U.S. Securities and Exchange Commission (SEC) makes companies' 10-K forms publicly available through the EDGAR system. The SEC stipulates the 10-K form's format and required content. It usually has four parts and sixteen items [26]. The item "Management's Discussion and Analysis of Financial Condition and Results of Operations" (MD&A) contains the management's summary of the company's results of operations and liquidity [27]. FINDSum uses the text in MD&A's two sections: "results of operations" and "liquidity and capital resources" as target summaries and the remaining content of each report document as the input.

After collecting tens of thousands of 10-K forms' HTML files, we parse them and split text and tables. To keep tables' positional information and align tables and text, we add a special token containing each table's index to concatenate the text before and after the table. Extracted text and tables are stored in separate files. Text and tabular data require different pre-processing procedures, considering their different natures. Our text pre-processing procedures include: removing noises (e.g., cover pages and special characters composing a style) and dividing text in different parts of 10-K form into text segments. To pre-process tabular data, we extract table content (e.g., names of rows and columns, cell content), remove noises in table content, and transform each cell into a tuple: (row name, column name, cell value, date, table index, row index, column index). The cell value in the tuple concatenates the original cell value and the rounding result with an ampersand. Besides, we remove duplicate samples and outliers with too-short input text, truncate too-long input text, split the training (80%), validation (10%), and test (10%) sets. Considering that the same company's annual reports in different years usually have duplicate content, we split these three sets by company to minimize their overlaps.

B. Dataset Description

FINDSum dataset is built on collected report documents. It has two subsets: FINDSum-ROO and FINDSum-Liquidity.

FINDSum-ROO is the subset focusing on each company's results of operations (ROO). In the ROO section of MD&A, the company's management usually compares and explains critical items of revenue and expense in the current and prior period [26].

[2] www.sec.gov/edgar/searchedgar/companysearch.html

TABLE II
PROPORTION OF NOVEL N-GRAMS IN TARGET SUMMARIES

Dataset	% of novel n-grams in target summary			
	unigrams	bigrams	trigrams	4-grams
CNN/DM	19.50	56.88	74.41	82.83
PubMed	18.38	49.97	69.21	78.42
arXiv	15.04	48.21	71.66	83.26
FINDSum-ROO	17.79	50.59	72.13	81.66
FINDSum-Liquidity	26.45	59.63	80.43	88.48

This section's text can be regarded as the target summary written by experts. Table I exhibits that the average number of numerical values in FINDSum-ROO's target summaries is dozens of times larger than that of previous datasets. However, nearly three-quarters of these numerical values cannot be found in these reports' remaining text. A lot of critical numerical information is only recorded in tables. Therefore, we use the remaining parts' text and all the tables in each report as inputs for each example.

FINDSum-Liquidity focuses on summarizing each company's liquidity and capital resources. The "liquidity and capital resources" section in MD&A mainly analyzes the company's ability to generate and obtain cash [27]. This section's text can be used as the target summary. Most of the numerical values in target summaries are not included in the remaining parts' text. FINDSum-Liquidity's inputs include the remaining text and all the tables in each report.

C. Dataset Analysis

We conduct statistics and analysis on FINDSum's two subsets. Table I shows that both the input documents and target summaries of these two subsets are much longer than those of previous summarization datasets. These two subsets' target summaries contain much more numerical information, while most of them cannot be found in the input text.

To measure how abstractive FINDSum's target summaries are, we count the percentage of summaries' novel n-grams not appearing in inputs. Table II shows that FINDSum-Liquidity's target summaries have more novel n-grams and are more abstractive. The abstractiveness of FINDSum-ROO's target summaries is similar to that of other datasets.

We also adopt three measures defined by Grusky et al. [6] to assess the extractive nature of summarization datasets. Given a document $D = [d_1, d_2, ..., d_n]$ consisting of a sequence of tokens d_i and its summary $S = [s_1, s_2, ..., s_m]$, extractive fragments $F(D, S)$ is the set of shared token sequences in D and S. In Equation (1a), extractive fragment coverage measures the percentage of summary words that are part of an extractive fragment from the input document. Equation (1b) calculates the extractive fragment density assessing the average length of the extractive fragment to which each summary word belongs. Besides, the compression ratio is the word ratio between the articles and their summaries, as shown in Equation (1c). We report the extractive fragment coverage and density in Table I. Two measures' distributions are visualized using kernel density estimation in Fig. 2. FINDSum's density is higher than those

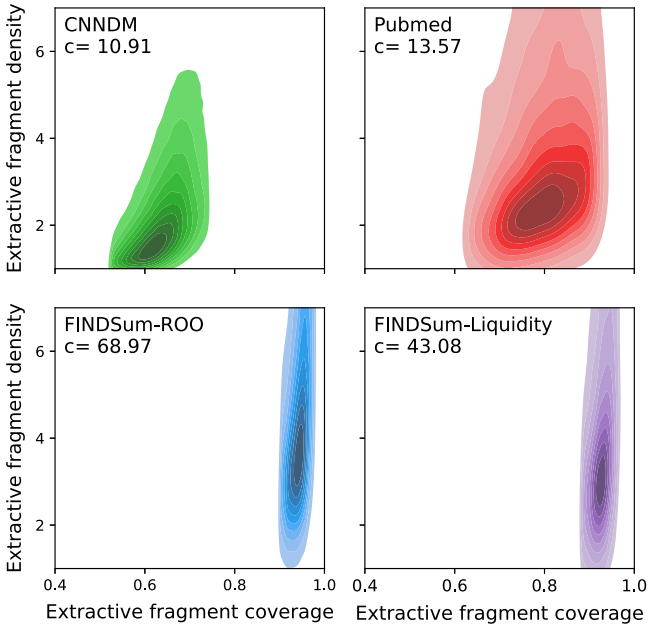


Fig. 2. Distributions of extractive fragments' density and coverage.

of previous datasets. The variability along the y-axis (density) suggests the varying writing styles in its target summaries.

$$\text{COVERAGE}(D, S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f| \tag{1a}$$

$$\text{DENSITY}(D, S) = \frac{1}{|S|} \sum_{f \in F(D,S)} |f|^2 \tag{1b}$$

$$\text{COMPRESSION}(D, S) = \frac{|D|}{|S|} \tag{1c}$$

IV. METHOD

Summarizing long text and multiple tables has several challenging issues: identifying the salient information from a large amount of input content, incorporating the text and tabular content into the summary generation, and efficiently processing long input and output sequences. This section presents our solution to the above issues.

A. Textual and Tabular Content Selection

Fig. 1 shows the three main steps of our solution: data pre-processing, content selection, and summarization. After the pre-processing, we get dozens of text segments and thousands of tuples from dozens of tables in each report. It is challenging to accurately identify the scattered salient content. We add the content selection step as a rough selection to compress long inputs while maximizing the recall of salient content that should be preserved in summaries. Then compressed inputs are fed into the summarizer for further selection. Content selection methods' output lengths should not exceed pre-specified lengths, as neural summarization models' complexity can scale with input sequence length.

Algorithm 1: Maximum Marginal Recall Gain (MMRG).

Input: Input m examples $I \leftarrow [e_1, \ldots, e_m]$, each example e_i contains n parts for selection $e_i \leftarrow [p_i^1, \ldots, p_i^n]$, the list of target item $T \leftarrow [t_1, \ldots, t_m]$, and the maximum number of selected parts n' ($n' \ll n$)
Output: The list of selected parts' id $S \leftarrow [j, \ldots, k]$ and the selected inputs $I' \leftarrow [e'_1, \ldots, e'_m]$, in which each example e'_i has selected parts $e'_i \leftarrow [p_i^j, \ldots, p_i^k]$ ($|e'_i| = |S| \leq n'$)

```
S ← [ ];
e'_1, ..., e'_m ← "", ..., "";
I' ← [e'_1, ..., e'_m];
while |S| < n' do
  //SelectPart finds the part p^{j_select} bringing the largest
  average recall gain across all examples
  j_select = SelectPart(I, I', T, S);
  if j_select > 0 then
    S ← S ∪ [j_select];
    while i ≤ m do
      I'[i] ← Concat(I'[i], p_i^{j_select});
    end while
  end if
end while
```

We employ separate methods to select salient content from textual and tabular data considering their different natures. To select salient text segments, we adopt a method named Maximum Marginal Recall Gain (MMRG) on our training set. Specifically, MMRG keeps adding the text segment bringing the maximum gain of n-gram's recall into the combination of selected segments till reaching the length limit. Finally, we can get selected salient segments' indexes and choose text segments with the same indexes for samples in our test set. Algorithm 1 is MMRG's pseudocode. We also follow Liu et al. [28] to try some extractive summarizers, like Textrank [29] and Lexrank [30], for salient text selection. Experimental results in Section VI-B show that MMRG performs the best, so we use it in our experiments.

As for those thousands of tuples extracted from tables, we model the salient tuple selection as a binary classification problem. Based on the FINDSum dataset, we annotate a tuple selection dataset for training and evaluating different classifiers (e.g., logistic regression, support vector machine, AdaBoost [31], XGBoost [32], and Multi-layer Perceptron).[3] We also utilize various features, including positional features (e.g., indexes of the row, column, table, and section, together with their normalized values) and text features (e.g., word embedding of row and column names). Considering the content selection step focuses more on the recall of salient content, we sort these tuples by their positive probability predicted by the trained classifier and use the top-n tuples' recall to evaluate these classifiers. Table VII shows that the XGBoost and MLP models equipped with positional features and Glove embedding [33] outperform others. We adopt them for tuple selection and compare their impact on the produced summaries in Section VI-B. We follow the setting in [34] to flatten the selected tuples into a linearized sequence.

[3]We use XGBoost's implementation from xgboost.readthedocs.io/ en/stable/ and other classifiers from scikit-learn.

B. Generating Summary for Textual and Tabular Data

To incorporate text and tabular data into summary generation, we present four types of methods: Generate-then-Combine (GC), Combine-then-Generate (CG), Generate-Template-then-Fill (GTF), and Generate-Combine-then-Generate (GCG). We show their structures in Fig. 3.

GC method makes two assumptions: 1) The summary of long text and multiple tables can be divided into text summary and table summary. 2) Summary generation can be divided into two parallel processes generating these two parts of summary. It assigns the maximum output lengths for the text summary and table summary, generates these two summaries separately, and concatenates them to form the final summary. GC has obvious limitations: 1) It cannot merge the information from text and tables when generating each summary sentence. 2) The pre-defined length assignment is not flexible enough to adapt to diverse examples.

CG is an end-to-end method generating a summary for both text and table content. It first concatenates the selected text segments and tuples with a special symbol and then feeds them into a sequence-to-sequence summarizer. The summarizer needs to learn both text-to-text and tuple-to-text generation. When generating summaries, it should jointly consider these two types of input content.

GTF method is inspired by how humans write quantitative descriptions of report documents. The CG and GC methods use similar processes to generate qualitative and quantitative descriptions, while the way people write quantitative descriptions differs from the way they write qualitative descriptions. Specifically, people usually decide which metrics to describe and then read tables to find numerical values to fill in the quantitative descriptions. When writing qualitative descriptions, they mainly refer to text content. GTF method has two stages: template generation and template filling, which mimic how humans write quantitative descriptions. The template generation stage generates all the words and the special token [num] as the placeholder for numerical values from tables. We regard the template filling as a question answering (QA) task. We use each template sentence containing the placeholder as a question and the linearized sequence of selected tuples as the context. We train the QA model to find the numerical value from table content as an answer to replace the placeholder. Table X shows that Bigbird's large model performs the best on the ROO subset. Longformer's large model performs the best on the Liquidity subset. We use them for the template filling in GTF.

GCG is another two-stage method. It employs a tuple-to-text generator to produce input tuples' text descriptions, concatenates the input text with the tuples' descriptions, and feeds them into the summarizer. Compared with the CG, GCG simplifies the requirement on the summarizer to focus on summarizing text, but the extra tuple-to-text generation process can lose

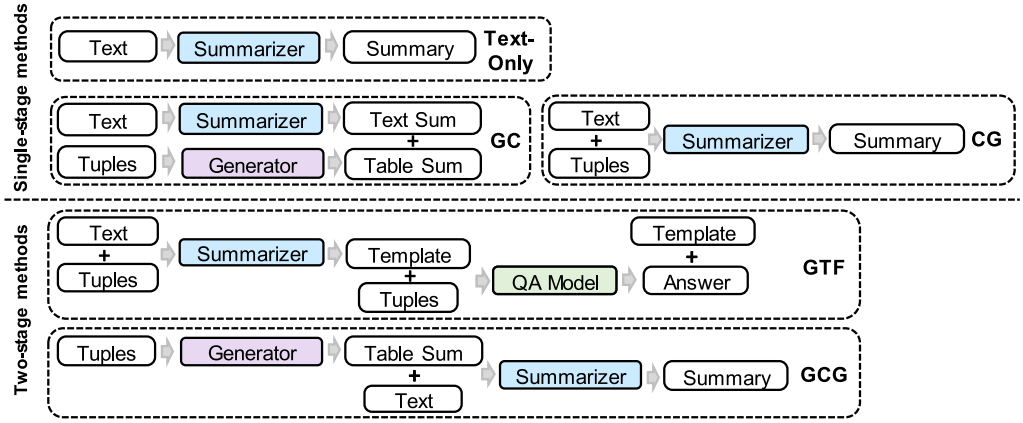


Fig. 3. Overview of our summarization methods.

some tuples' information. We annotate a tuple-to-text generation dataset based on our FINDSum dataset for training and evaluating various generators. Table XII indicates that the BART-large outperforms other baselines, so we use it as the tuple-to-text generator in GCG.

C. Processing Long Inputs and Outputs

Input documents in our FINDSum-ROO and FINDSum-Liquidity subsets contain tens of thousands of words. The average length of target summaries in FINDSum-Liquidity exceeds 1,000 words. Long inputs and outputs bring some problems: 1) The transformer model's self-attention mechanism [2], [35] scales quadratically with the input sequence length. It is prohibitively expensive for long input [36] and precludes the usage of large pre-trained models with limited computational resources. 2) Existing autoregressive abstractive summarization methods still have difficulty in generating long text in terms of efficiency and quality [3], [4]. To deal with the first problem, we employ sparse attention mechanisms [37], [38] in our summarization models' encoders. The content selection step in our solution also reduces the length of input sequences. To handle the second problem, we follow a divide-and-conquer method [39] and decompose the long summary generation problem into multiple sub-problems of summary segment generation. These summary segments can be generated in parallel and merged as a final summary. To minimize output summaries' redundancy, we add a constraint that the MMRG in the content selection step should not select the same combination of input text segments for generating different summary segments.

V. Experiments

A. Baselines

In our experiments, we adopt advanced extractive and abstractive summarization models as baselines.

LexRank and TextRank [29], [30] are two graph-based ranking methods that can be used for unsupervised extractive summarization.

BART [40] is a denoising autoencoder built with a sequence-to-sequence model pre-trained to reconstruct the original input text from the corrupted text.

PEGASUS [16] is a transformer-based model pre-trained with the Gap Sentences Generation (GSG) and Masked Language Model (MLM) objectives.

LongT5 [41] extends the original T5 encoder [42] with a global-local attention mechanism to handle long inputs.

BigBird-PEGASUS [37] adopts the BigBird encoder with sparse attention mechanisms and the PEGASUS decoder.

Longformer-Encoder-Decoder (LED) [38] follows BART's architecture and adopts sparse attention in its encoder.

B. Experimental Setting

The vocabulary's maximum size is set as 50,265 for summarization models, while the tuple-to-text generators use 32,128 as default. When fine-tuning these pre-trained models, we use the learning rate of $5e^{-5}$ and adopt the learning rate warmup and decay. The optimizer is Adam with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We use dropout with a probability of 0.1. In the generation process, we use beam search with a beam size of 5. Trigram blocking is used to reduce repetitions. We adopt the implementations of BART, PEGASUS, T5, BigBird, and LED from HuggingFace's Transformers [43]. All the models are trained on one NVIDIA RTX 8000 GPU.

C. Evaluation Metrics

We propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. This is necessary for long text and multi-table summarization. We use D, S, and H to denote the input document, human-written target summary, and the summarizer's output summary. D_n, S_n, and H_n are sets of numbers contained in them. $|D_n|$, $|S_n|$, and $|H_n|$ denote the sizes of these number sets. For a produced summary H, we first extract the number set H_n from it.[4] Then $M(H_n, S_n)$ counts numbers appearing in both the produced summary H and the

[4]We do not count numbers in a word, like COVID-19.

TABLE III
AUTOMATIC EVALUATION RESULTS ON TEST SETS OF FINDSUM-LIQUIDITY AND FINDSUM-ROO

Type	Method	FINDSum-Liquidity						FINDSum-ROO					
		R-1	R-2	R-L	NP	NC	NS	R-1	R-2	R-L	NP	NC	NS
Only Text	LexRank	40.67	10.61	16.28	12.58	14.50	13.47	34.43	7.73	14.92	14.77	9.73	11.73
	TextRank	41.71	10.90	16.54	13.37	13.02	13.19	35.93	7.74	15.08	14.68	10.96	12.55
	BART	52.37	17.91	19.59	21.18	22.78	21.95	49.00	16.88	19.14	14.38	23.72	17.91
	PEGASUS	52.57	18.46	19.75	16.98	22.74	19.44	51.92	19.31	21.47	10.90	21.89	14.55
	LongT5	44.89	14.61	17.39	13.74	17.00	15.20	43.26	11.84	17.83	8.75	10.37	9.49
	LED	53.52	18.91	19.75	18.68	22.56	20.44	53.06	20.33	22.28	14.25	22.99	17.59
	BigBird-PEGASUS	53.42	19.39	20.07	17.16	22.44	19.45	53.08	20.85	20.94	13.15	23.82	16.95
Single Stage	GC-LED	52.30	20.09	19.61	15.13	**44.47**	22.58	53.19	21.97	22.84	12.83	**41.54**	19.60
	GC-BigBird	51.61	20.00	19.86	14.76	44.21	22.13	53.13	22.03	23.11	12.49	41.30	19.18
	CG-LED	54.12	20.26	20.46	21.86	35.14	**26.95**	54.24	22.08	23.10	16.41	33.89	22.11
	CG-BigBird	53.82	20.15	20.39	20.98	34.29	26.03	**54.40**	**22.48**	**23.21**	**16.46**	35.84	**22.56**
Two Stage	GTF-LED	53.88	19.82	20.13	21.37	31.76	25.55	53.60	21.61	22.89	15.49	29.06	20.21
	GTF-BigBird	53.66	19.56	19.97	**21.96**	30.52	25.54	54.07	21.93	22.85	15.27	29.99	20.24
	GCG-LED	**54.55**	20.36	20.41	21.15	34.52	26.23	54.32	21.92	23.03	16.03	32.54	21.48
	GCG-BigBird	53.90	**20.47**	**20.59**	20.67	36.43	26.38	54.12	22.11	23.02	15.33	32.82	20.90

target summary S. $M(D_n, S_n)$ counts numbers appearing in both the input document D and the target summary S.

We mainly consider three metrics: Number Precision (NP), Number Coverage (NC), and Number Selection (NS). Calculated by Equation (2), NP is the ratio of numbers in the produced summary that also appears in the target summary. It measures how well the produced summary matches the target summary in terms of contained numbers. NC measures how well the produced summary covers the numbers appearing in both the target summary and the input document. Some of the numbers in the target summary cannot be directly found in the inputs (including textual and tabular data) and need numerical reasoning. Some of them may be lost when preparing the summarization model's inputs, which can limit the produced summary's number recall computed by Equation (3a). To evaluate the summarization model's coverage capability, we divide the produced summary's number recall by the input document's number recall in Equation (3b). NS calculates the harmonic mean of NP and NC in Equation (4) and reflects the quality of number selection in the produced summary.

$$\text{NP}(H_n, S_n) = \frac{M(H_n, S_n)}{|H_n|} \tag{2}$$

$$\text{NR}(H_n, S_n) = \frac{M(H_n, S_n)}{|S_n|} \tag{3a}$$

$$\text{NC}(D_n, H_n, S_n) = \frac{\text{NR}(H_n, S_n) * |S_n|}{M(D_n, S_n)} \tag{3b}$$

$$\text{NS}(D_n, H_n, S_n) = \frac{2 * \text{NP} * \text{NC}}{\text{NP} + \text{NC}} \tag{4}$$

VI. RESULTS AND DISCUSSION

This section presents our experimental results and analysis. We conduct automatic and human evaluations to compare the quality of summaries produced by different models. We also conduct extensive comparative experiments to compare and analyze the influence of different components and configurations of summarization models. Finally, a case study compares and analyses different models' output summaries.

A. Summarization Results

In the automatic evaluation, we calculate the ROUGE F_1 scores [5], including the overlaps of unigrams (R-1), bigrams (R-2), and longest common subsequence (R-L),[5] and NP, NC, and NS scores. We employ a divide-and-conquer approach to generate summary segments in parallel and then merge them as the final summary. Tables III, IV, V, and IX report the final merged summaries' scores. In Table III, all the abstractive methods are built on large pre-trained models. Limited by the GPU memory size, the input text length of models using full attention mechanism is 1024, while that of models with sparse attention mechanisms is 2048. The GC, CG, GTF, and GCG methods in Table III receive selected tuples' linearized sequences of length 1024. Table III shows that these abstractive summarizers based on pre-trained models outperform unsupervised extractive summarizers. Compared with other baselines based on full attention, LED [37] and BigBird-PEGASUS [38] equipped with sparse attention can model longer context and achieve higher ROUGE scores. Longer inputs can cover more scattered salient content, which benefits output summaries' informativeness.

Our CG, GTF, and GCG methods outperform these text-only baselines on FINDSum's two subsets. Incorporating tabular information is conducive to improving the NP, NC, NS, and ROUGE scores. GCG methods perform better on FINDSum-Liquidity, while CG methods perform better on FINDSum-ROO. Table I shows that target summaries in the FINDSum-ROO subset have a larger ratio of numerical information not found in the input text and rely more on tables. The table content passes one generation process in CG methods but needs to pass through two stages in GTF and GCG methods. The

[5] github.com/falcondai/pyrouge/

TABLE IV
GC METHODS' EVALUATION RESULTS

Text/ Tuple	Method	FINDSum-Liquidity						FINDSum-ROO					
		R-1	R-2	R-L	NP	NC	NS	R-1	R-2	R-L	NP	NC	NS
1:1	GC-LED	52.30	20.09	19.61	15.13	**44.47**	**22.58**	53.19	21.97	22.84	12.83	**41.54**	19.60
	GC-BigBird	51.61	20.00	**19.86**	14.76	44.21	22.13	53.13	22.03	**23.11**	12.49	41.30	19.18
2:1	GC-LED	52.28	18.37	19.12	**16.63**	22.45	19.11	53.56	21.95	22.78	13.45	36.54	19.66
	GC-BigBird	52.99	**20.18**	19.81	14.43	35.62	20.54	53.51	22.02	22.69	12.82	38.74	19.26
3:1	GC-LED	52.57	18.47	19.21	16.13	22.24	18.70	**53.66**	21.88	22.48	**13.62**	36.21	**19.79**
	GC-BigBird	**53.33**	20.15	19.81	14.58	32.30	20.09	53.59	**22.07**	22.73	13.18	35.84	19.27

"Text/Tuple" denotes the assigned length ratio of text summary and table summary in each combined summary.

TABLE V
HUMAN EVALUATION RESULTS

	CG-BigBird				GTF-BigBird				GCG-BigBird			
	Win	Lose	Tie	Kappa	Win	Lose	Tie	Kappa	Win	Lose	Tie	Kappa
FINDSum-ROO												
Informativeness	44.2%	20.8%	35.0%	0.626	42.5%	23.3%	34.2%	0.631	43.3%	20.8%	35.8%	0.653
Fluency	26.7%	25.8%	47.5%	0.616	25.0%	26.7%	48.3%	0.648	27.5%	24.2%	48.3%	0.613
Non-Redundancy	35.0%	23.3%	41.7%	0.632	34.2%	21.7%	44.2%	0.636	33.3%	21.7%	45.0%	0.644
FINDSum-Liquidity												
Informativeness	40.8%	20.8%	38.3%	0.620	40.0%	22.5%	37.5%	0.649	41.7%	21.6%	36.7%	0.655
Fluency	25.0%	24.2%	50.8%	0.615	24.2%	23.3%	52.5%	0.637	25.8%	25.0%	49.2%	0.611
Non-Redundancy	31.7%	22.5%	45.8%	0.626	30.8%	24.2%	45.0%	0.629	32.5%	23.3%	44.2%	0.638

"Win" represents the generated summary of our method is better than that of BigBird-PEGASUS.

extra stage can lose some required tabular information and accumulate more errors. In FINDSum-Liquidity, a larger ratio of the numerical values can be found in the input text, and the loss of tabular information in the extra stage has less effect. Table XII depicts that these tuple-to-text generators perform better on the Liquidity subset, which also contributes to the GCG methods' performance on the FINDSum-Liquidity. GTF methods' performance is mainly limited by the template generation process, which needs to decide whether to copy the numerical values appearing in the input text or the values in input tables and put placeholders in the exact positions. Meanwhile, better template filling methods can also benefit produced summaries' quality.

The GC methods do not perform well, which is due to GC's limitations mentioned in Section IV-B. In Table III, the GC methods' evaluation results represent summaries combining text and table summaries of the same length. Although they can achieve high NC scores, their NP and ROUGE scores are unsatisfactory. The result reflects that it is not appropriate to treat long text and multi-table summarization as two independent processes. Table IV shows that the pre-defined length assignment can affect the combined summaries' quality. It is not flexible enough to adapt to diverse examples.

Our human evaluation compares different models' output summaries in terms of informativeness (the coverage of information from input documents), fluency (content organization and grammatical correctness), and non-redundancy (less repetitive information). We randomly selected 30 samples from the test sets of the FINDSum-ROO and FINDSum-Liquidity, respectively. Four annotators are required to compare two models' output summaries presented anonymously. We also assess their agreements by Fleiss' kappa [44]. Table V exhibits that CG-BigBird, GTF-BigBird, and GCG-BigBird significantly outperform the BigBird-PEGASUS only using input text in terms of informativeness and are comparable in terms of fluency and non-redundancy. It verifies that incorporating text and tabular data into summary generation can benefit output summaries' informativeness.

The following subsections discuss our extensive comparative experiments comparing the performance of different model components (e.g., content selection methods, divide-and-conquer method, sparse attention mechanisms, template filling methods in GTF, and tuple-to-text generators in GCG). We also analyze their influence on summarization results.

B. Discussion on Content Selection Methods

As introduced in Section IV-A, the content selection step filters out the non-prominent content and retains the salient content as summarizers' inputs. Different methods are employed to select salient content from text and tabular data, considering their different natures. We conduct a series of experiments to compare the performance of different content selection methods and their impact on summarization results.

To select salient text content, we compare the statistics-based MMRG method with some extractive summarization methods, like TextRank and Lexrank. We use the recall of n-grams to evaluate these methods' performance in selecting the salient text of the same length. Table VI indicates that MMRG outperforms these extractive summarizers. We also compare their impact on

TABLE VI
EVALUATION RESULTS OF INPUT TEXT SELECTION METHODS

Method	FINDSum-ROO				FINDSum-Liquidity					
	Segment 1		Segment 2		Segment 1		Segment 2		Segment 3	
	R-1	R-AVG	R-1	R-AVG	R-1	R-AVG	R-1	R-AVG	R-1	R-AVG
LexRank	56.01	22.14	53.96	20.72	49.71	18.59	48.92	17.97	46.45	17.00
TextRank	58.38	22.94	56.25	21.53	55.18	20.94	54.02	20.40	51.72	19.49
MMRG	**63.38**	**28.01**	**61.68**	**27.85**	**58.61**	**24.28**	**56.69**	**23.09**	**53.94**	**21.62**

R-1 denotes the recall of unigram, and R-AVG is the average recall of unigram, bigram, trigram, and 5-gram.

TABLE VII
EVALUATION RESULTS OF SALIENT TUPLE SELECTION

Method	Features	Liquidity						ROO					
		Top-100		Top-200		Top-400		Top-100		Top-200		Top-400	
		ACC	Rec	ACC	Rec	ACC	Rec	ACC	Rec	ACC	Rec	ACC	Rec
LR	Pos	94.53	40.36	89.32	61.95	78.64	73.01	94.54	41.53	89.27	56.08	78.60	68.78
	Pos+Glove	94.64	52.96	89.36	66.84	78.70	79.69	94.56	43.39	89.31	60.58	78.64	73.28
SVM	Pos	94.55	43.19	89.34	64.27	78.63	72.24	94.55	42.86	89.28	57.14	78.62	70.63
	Pos+Glove	94.64	53.73	89.36	66.58	78.69	79.43	94.56	43.65	89.31	60.58	78.65	74.34
AdaBoost	Pos	94.61	50.13	89.35	65.04	78.68	78.15	94.57	45.24	89.31	60.05	78.62	70.90
	Pos+Glove	94.69	58.87	89.42	73.78	78.74	85.35	94.56	43.12	89.30	58.99	78.65	75.40
XGBoost	Pos	94.61	49.61	89.38	69.15	78.70	79.95	94.59	47.62	89.32	62.17	78.65	75.13
	Pos+Glove	**94.74**	**65.30**	**89.46**	78.15	**78.77**	**88.43**	94.63	52.65	**89.36**	**67.20**	**78.71**	**82.28**
MLP	Pos	94.61	50.13	89.37	67.87	78.69	78.41	94.60	48.41	89.32	61.90	78.65	74.60
	Pos+Glove	**94.74**	**65.30**	**89.46**	**78.66**	78.76	87.40	**94.64**	**53.97**	89.34	64.55	78.67	77.25

TABLE VIII
IMPACT OF TEXT CONTENT SELECTION METHODS ON SUMMARIZATION RESULTS

Summarizer	Text Select	R1/R2/RL
FINDSum-Liquidity		
LED-large	MMRG	**53.52/18.91/19.75**
	Textrank	51.76/17.35/19.02
	Lexrank	51.68/17.29/19.00
BigBird-PEGASUS	MMRG	**53.42/19.39/20.07**
	Textrank	51.09/17.29/19.20
	Lexrank	51.00/17.16/19.15
FINDSum-ROO		
LED-large	MMRG	**53.06/20.33/22.28**
	Textrank	48.75/16.07/19.78
	Lexrank	48.73/16.10/19.78
BigBird-PEGASUS	MMRG	**53.08/20.85/20.94**
	Textrank	49.59/17.30/20.67
	Lexrank	49.78/17.48/20.66

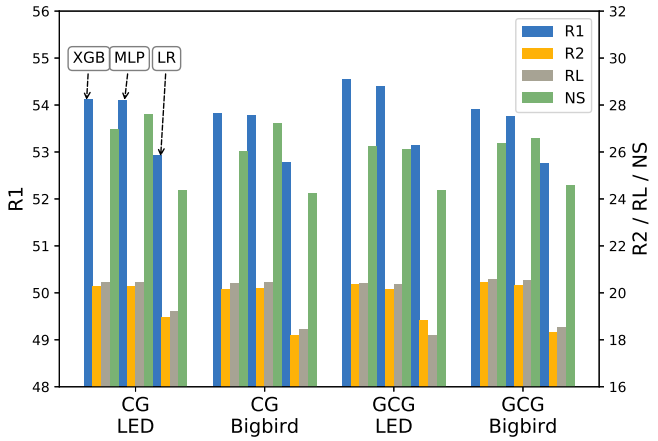


Fig. 4. Impact of tuple selection methods on FINDSum-Liquidity. Each summarization method using outputs of XGBoost, MLP, and LR has three parts of scores.

the ROUGE scores of produced summaries. Table VIII depicts that the better text content selection method benefits the quality of produced summaries.

As for those thousands of tuples extracted from tables, we model the salient tuple selection as a binary classification task. We annotate a tuple selection dataset based on the FINDSum dataset. Salient tuples (positive samples) are usually sparse in these report documents. To deal with the class imbalance problem, we perform undersampling over negative samples to ensure the ratio of positive and negative samples is 1:10 in the training set. We train and evaluate different classification methods, including the logistic regression (LR), support vector machine (SVM), AdaBoost, XGBoost, and Multi-layer Perceptron (MLP), on our annotated tuple selection dataset. We use the accuracy and recall of salient tuples to evaluate these classifiers. Table VII shows that adding word embeddings of row and column names as features significantly improves the recall of all classifiers and facilitates finding salient table content. Besides, XGBoost and MLP models equipped with positional features and Glove embedding [33] outperform other classifiers. We compare three tuple selection methods' impact on the quality of produced summaries. Figs. 4 and 5 depict that summarization models receiving the outputs of XGBoost and MLP outperform

TABLE IX
EFFECT OF INPUT SEQUENCE LENGTH ON GENERATED RESULTS

Input Len	Method	FINDSum-Liquidity						FINDSum-ROO					
		R-1	R-2	R-L	NP	NC	NS	R-1	R-2	R-L	NP	NC	NS
2K+1K	LED-base	52.91	18.41	19.58	19.97	24.07	21.83	52.96	20.52	22.14	14.60	24.67	18.34
	CG-LED-base	53.73	19.68	20.26	21.76	34.39	26.65	54.38	22.13	23.10	17.21	34.94	23.06
	GTF-LED-base	53.89	19.39	19.88	20.77	30.73	24.79	53.76	21.54	22.81	14.69	29.07	19.52
	GCG-LED-base	54.01	19.94	20.26	20.29	35.32	25.77	53.91	21.69	22.83	15.90	30.23	20.84
4K+2K	LED-base	53.58	19.29	20.01	19.68	24.63	21.88	53.68	21.64	22.95	15.66	26.10	19.58
	CG-LED-base	54.42	20.40	20.44	22.32	38.10	28.15	54.51	22.63	23.42	17.55	35.06	23.39
	GTF-LED-base	54.02	19.83	20.10	21.75	30.70	25.46	54.16	22.31	23.27	16.11	30.37	21.05
	GCG-LED-base	54.14	20.11	20.32	21.23	36.78	26.92	**54.53**	**22.63**	23.45	16.07	32.71	21.55
8K+4K	LED-base	54.04	19.88	20.31	20.40	26.35	23.00	53.39	21.53	22.97	15.91	26.94	20.01
	CG-LED-base	**54.82**	**20.95**	**20.78**	**24.49**	**41.36**	**30.76**	54.43	22.59	**23.46**	**18.24**	**35.69**	**24.14**
	GTF-LED-base	54.61	20.67	20.55	23.05	32.27	26.89	53.92	21.96	23.11	16.61	30.09	21.40
	GCG-LED-base	54.60	20.71	20.57	22.53	38.26	28.36	54.09	22.23	23.21	15.98	32.58	21.44

"Input Len" denotes the length of input text and flattened tuples.

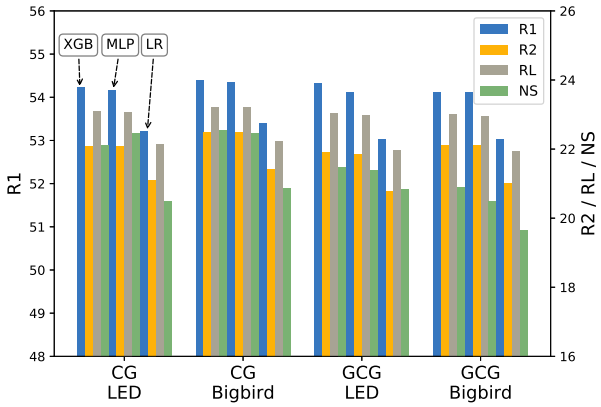


Fig. 5. Impact of tuple selection methods on FINDSum-ROO. Each summarization method using outputs of XGBoost, MLP, and LR has three parts of scores.

summarization models using the LR model's outputs. Better tuple selection methods benefit the quality of produced summaries. This verifies the effectiveness of our tuple selection methods.

C. Discussion on Input Length of Summarization Model

How to set the input sequence length of the summarization model is also an important issue. We conduct a series of experiments to analyze the input sequence length's impact on the performance of summarization models. Considering the constraint of GPU memory size, we use the base model of LED as the backbone. We compare the performance of text-only, CG, GTF, and GCG methods over the inputs of different lengths. Table IX shows that these methods built on the base model with longer inputs can surpass these methods based on the large model shown in Table III. When the GPU memory size is limited, using the base model with longer inputs may be better. However, increasing input length does not always bring performance gain. As shown in Figs. 6 and 7, all of these summarization models improve when the input length is doubled. When the input length increases from double to quadruple, models' performance on the Liquidity subset improves, while some models' ROUGE scores on the ROO subset decrease. Longer inputs can cover more salient information, which benefits generating informative summaries. Meanwhile, longer inputs can also introduce more non-prominent content, making it more difficult for summarization models to identify salient content. When adjusting the input length, we should find a balance between covering salient information and reducing non-prominent content to meet the requirements of various outputs.

D. Discussion on the Divide-and-Conquer Method

Existing autoregressive abstractive summarization methods still have difficulty in generating long text in terms of efficiency and quality [3], [4]. We adopt a divide-and-conquer (DC) method [39], which decomposes the long summary generation problem into multiple sub-problems of summary segment generation. These summary segments can be generated in parallel and merged as a final summary. We conduct experiments comparing the performance of these summarization models with and without the divide-and-conquer method. Figs. 6 and 7 show that DC can bring additional performance gains even when the context length grows. When the GPU memory size limits the model's context length, DC can help the model produce better summaries with relatively short inputs. It reveals the effectiveness of the divide-and-conquer method. To reduce the redundancy in the merged summary, we add constraints to MMRG to avoid providing the same inputs for summarizers and use trigram blocking in the summary generation process, as discussed in Sections IV-C and V-B. Besides, we train a separate model to generate each summary segment. Different segments of target summaries can supervise separate summarization models to focus on different content.

E. Discussion on Template Filling Methods

Template filling is the second stage in GTF methods introduced in Section IV-B. We model the template filling process as a question answering (QA) task. We use each template sentence

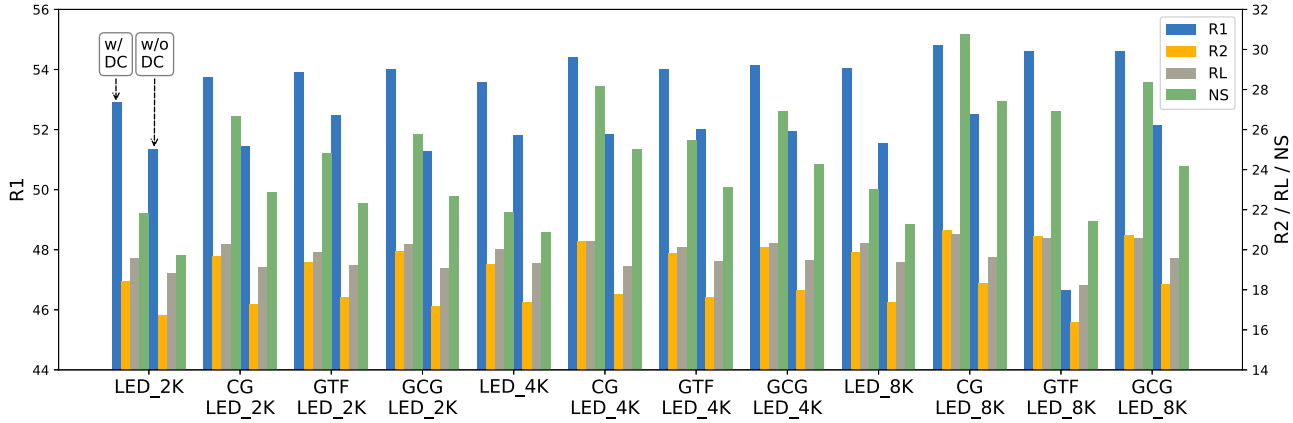


Fig. 6. Impact of input length and Divide-and-Conquer (DC) on FINDSum-Liquidity. Each summarizer has two parts of scores denoting w/ and w/o DC.

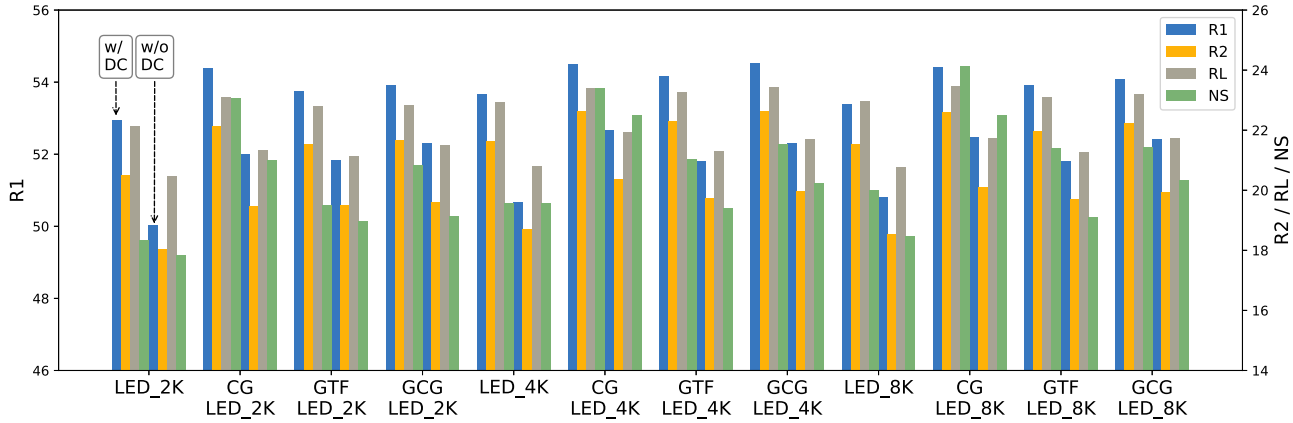


Fig. 7. Impact of input length and Divide-and-Conquer (DC) on FINDSum-ROO. Each summarizer has two parts of scores denoting w/ and w/o DC.

containing the placeholder as a question and the linearized sequence of selected tuples as the context. We annotate a question answering dataset based on FINDSum and employ extractive or generative QA models to find numerical values from table content as answers to replace placeholders in questions. Considering the requirements of template filling, we use exact match (EM) to evaluate template filling methods. Table X shows that the Bigbird-large outperforms other baselines on the ROO subset, while the Longformer-large performs the best on the Liquidity subset. Besides, these extractive methods perform better than generative methods. We find that generative methods still suffer from hallucinations and can generate inaccurate numerical values. Compared with the backbone BART model [40], pre-training on table-related tasks brings performance gain to the TAPEX model [34]. Table XI compares the impact of different template filling methods on the quality of generated summaries when using the same template generation method. As shown in Table X, the large models of Bigbird and Longformer perform better than their base models in template filling. These better template filling methods benefit the NP, NC, and NS scores of produced summaries. Presenting accurate numerical values is essential for financial reports' summaries. Template filling methods have less impact on ROUGE scores because there are many fewer numerical values than words in target summaries.

TABLE X
EVALUATION RESULTS OF TEMPLATE FILLING

Dataset	Type	Method	EM
ROO	Gen	BART-base	71.71
		BART-large	75.40
		TAPEX-base	74.75
		TAPEX-large	76.82
	Ext	Bigbird-base	76.75
		Bigbird-large	**80.72**
		Longformer-base	77.88
		Longformer-large	80.30
Liquidity	Gen	BART-base	73.13
		BART-large	70.96
		TAPEX-base	75.49
		TAPEX-large	74.59
	Ext	Bigbird-base	77.59
		Bigbird-large	78.12
		Longformer-base	78.31
		Longformer-large	**79.99**

EM denotes exact match.

F. Discussion on Tuple-to-Text Generation Methods

As introduced in Section IV-B, tuple-to-text generation is the first step in GCG methods. We annotate a tuple-to-text

Input

Table 1: Statements of Cash Flows (In thousands)

	2019
Net loss	(94,433)
Adjustments to reconcile net loss to net cash used in operating activities:	
Depreciation and amortization	4,745
Net amortization of premiums and discounts on marketable securities	(1,349)
Stock-based compensation	5,299
Non-cash operating lease expense	6,382
Changes in operating assets and liabilities:	
…	…
Deferred revenue	(4,297)
…	…
Purchase of marketable securities	(80,979)
Maturities of marketable securities	112,993
Purchase of property and equipment	(16,173)
…	…

Table 2: (In thousands)

	2019
Cash used in operating activities	(85,011)
Cash provided by (used in) investing activities	15,841
…	…

…

Input Text:

…

we have funded our operations to date primarily from private placements of our convertible preferred stock , the net proceeds from our initial public offering … we have incurred net losses each year since inception. our net losses were $ 94.4 million, $ 64.8 million and $ 41.4 million for the years ended december 31, 2019, 2018 and 2017, respectively. as of december 31, 2019, we had an accumulated deficit of $ 221.0 million … since our inception, we have incurred significant losses and negative cash flows from operations . we have an accumulated deficit of $ 221.0 million through december 31, 2019. we expect to incur substantial additional losses in the future as we conduct and expand our research and development activities. …

Output Summary

Target Summary: cash used in operating activities was $ 85.0 million , which consisted of a net loss of $ 94.4 million , adjusted by non-cash charges of $ 15.1 million and cash used due to changes in our operating assets and liabilities of $ 5.7 million. the non-cash charges consisted primarily of depreciation and amortization expense of $ 3.4 million , stock-based compensation of $ 5.3 million, and non-cash operating lease expense of $ 6.4 million. … during the year ended december 31 , 2019 , cash provided by investing activities was $ 15.8 million , which consisted of $ 113.0 million in proceeds from the maturity of marketable securities , offset by $ 81.0 million of purchases of marketable securities and $ 16.2 million of capital expenditures to purchase property and equipment. …

GCG-Bigbird: operating activities net cash used in operating activities was $ 85.0 million for the year ended december 31, 2019, primarily resulting from our net loss of $ 94.4 million, which was partially offset by non-cash charges of $ 50.0 million for depreciation and amortization, $ 0.7 million for stock-based compensation, and $ 6.4 million for operating lease expense. … net cash provided by investing activities was $ 15.8 million during the periods presented, primarily due to maturities of marketable securities of $ 64.3 million and purchases of property and equipment of $ 5.7 million offset by purchases of available-for-sale marketable securities totaling $ 80.0 million …

CG-Bigbird: since our inception, we have incurred significant losses and negative cash flows from operations. we have an accumulated deficit of $ 221.0 million through december 31, 2019. we expect to incur substantial additional losses in the future as we conduct and expand our research and development activities and invest in our manufacturing facility. … operating activities net cash used in operating activities was $ 81.0 million for the year ended december 31, 2019, and consisted primarily of a net loss of $ 94.4 million, partially offset by non-cash charges of $ 55.0 million and net cash provided by changes in our operating assets and liabilities of $ 15.8 million. the net loss was primarily due to an increase in the net deferred revenue of $ 4.3 million due to the timing of payments …
$ 5.3 million in stock-based compensation expense and $ 6.4 million in operating lease expense … net cash from investing activities for 2019 was $ 16.4 million and consisted primarily of maturities and purchases of marketable securities of $ 48.7 million …

GTF-Bigbird: since our inception, we have incurred significant losses and negative cash flows from operations. we have funded our operations to date primarily from private placements of our convertible preferred stock, the net proceeds from our initial public offering … net cash used in operating activities for the year ended december 31, 2019 was $ 85 million and consisted primarily of our net loss of $ 94.4 million, partially offset by non-cash charges for stock-based compensation, depreciation and amortization, operating lease expense, net amortization of premiums and discounts on marketable securities and net changes in our operating assets and liabilities. … our investing activities consist primarily of purchases and maturities of marketable securities, purchases and sales of property and equipment, and purchases of intangible assets. …

BigBird-PEGASUS: since our inception, we have incurred significant losses and negative cash flows from operations. we have funded our operations to date primarily from private placements of our convertible preferred stock, the net proceeds from our initial public offering, … net cash used in operating activities was $ 64.6 million, primarily resulting from our net loss of $ 92.2 million and changes in our operating assets and liabilities, partially offset by non-cash charges totaling $ 19.9 million. … net cash from investing activities was $ (20.0) million for the years ended december 30, 2019 and 2018, respectively. during the year, we purchased marketable securities totaling $ 20.0 million and purchased property and equipment, net of cash acquired, of $ 0.2 million and $ 0.1 million respectively, which were offset by maturities of marketable securities of $ 25.0 million …

Fig. 8. Input content and output summaries of an example from the FINDSum-Liquidity. In these output summaries, the underlined content comes from row names or cell values of input tables or input text fragments. The summary sentences marked with dotted lines below are mainly derived from the input text, while those marked with solid lines below mainly come from the input tables.

TABLE XI
IMPACT OF TEMPLATE FILLING METHODS

TG	TF	R1/R2/RL	NP/NC/NS
FINDSum-ROO			
BigBird-PEGASUS	Bigbird-base	53.90/21.81/22.76	14.16/27.7/18.74
	Bigbird-large	**54.07**/**21.93**/22.85	15.27/**29.99**/**20.24**
LED	LF-base	53.51/21.50/22.78	14.08/28.48/18.84
	LF-large	53.60/21.61/**22.89**	**15.49**/29.06/20.21
FINDSum-Liquidity			
BigBird-PEGASUS	Bigbird-base	53.52/19.45/19.86	21.14/30.16/24.86
	Bigbird-large	53.66/19.56/19.97	**21.96**/30.52/25.54
LED	LF-base	53.75/19.71/20.03	20.64/31.04/24.79
	LF-large	**53.88**/**19.82**/**20.13**	21.37/**31.76**/**25.55**

TG and TF denote the template generation and template filling methods. LF is the longformer model.

TABLE XII
EVALUATION RESULTS OF TUPLE-TO-TEXT GENERATION

Dataset	Method	R-1	R-2	R-L	BLEU
ROO	T5-base	45.45	24.77	28.84	12.20
	T5-large	45.81	24.64	29.04	12.87
	BART-base	42.08	20.45	25.86	10.57
	BART-large	**47.21**	**25.63**	**31.08**	**13.14**
Liquidity	T5-base	48.90	**28.34**	31.98	15.44
	T5-large	49.03	28.05	32.02	15.86
	BART-base	45.71	24.75	29.28	13.66
	BART-large	**49.78**	28.24	**32.59**	**16.05**

generation dataset based on our FINDSum dataset for training and evaluating various generators. These tuple-to-text generators are evaluated by the ROUGE [5] and BLEU scores[6] [45]. Table XII depicts the performance of different tuple-to-text generators on ROO and Liquidity subsets. The

[6]www.nltk.org/api/nltk.translate.bleu_score.html. We report the cumulative 4-gram BLEU score.

TABLE XIII
N-GRAM RECALL OF TUPLE-TO-TEXT GENERATION RESULTS ON TEST SETS OF FINDSUM-ROO AND FINDSUM-LIQUIDITY

	Top-N	R-1	R-2	R-3	R-5	R-AVG
ROO						
XGB	100	36.32	16.55	7.45	1.47	15.45
	200	42.73	20.67	9.78	2.01	18.80
	400	49.25	**24.88**	**12.19**	**2.59**	22.23
MLP	100	36.75	16.69	7.51	1.49	15.61
	200	42.98	20.69	9.74	1.97	18.85
	400	**49.38**	24.86	12.16	2.57	**22.24**
Liquidity						
XGB	100	33.47	15.27	7.12	1.69	14.39
	200	40.11	18.99	9.26	2.13	17.62
	400	46.84	22.78	11.51	2.67	20.95
MLP	100	33.65	15.39	7.17	1.68	14.47
	200	40.30	19.10	9.31	2.15	17.72
	400	**47.06**	**22.87**	**11.55**	**2.70**	**21.05**

R-AVG is the average recall of Unigram, Bigram, Trigram, and 5-Gram.

large model of BART [40] performs the best on these two subsets, so we use it as the tuple-to-text generator in GCG. Table XIII depicts that both the tuple selection methods in the content selection step and the number of input tuples can affect tuple-to-text generators' performance.

G. Case Study

We conduct a case study to compare and analyze summaries generated by different models. Fig. 8 has two parts. Its left part shows some fragments of input text and tables from one example in the FINDSum-Liquidity. The right part presents fragments in the target summary and different models' output summaries. When comparing these summaries, we find that our GCG, CG, and GTF methods can generate quantitative descriptions of some critical items in tables. The text-only baseline BigBird-PEGASUS focuses more on narratives in the input text. Without tabular data as evidence, most of the numerical values generated by the BigBird-PEGASUS are inaccurate. It reflects the importance of incorporating tables when summarizing report documents.

GCG method's input is the concatenation of input text and generated text descriptions of selected tabular data, which differs from the CG method. The summary generated by GCG focuses more on descriptions of tables. Unlike the GCG method, the CG method needs to handle text-to-text and tuple-to-text generation simultaneously, which is quite challenging. The generated summary reflects that the CG method can find a balance for its focus on text and table content. The accuracy of its tuple-to-text generation needs further improvements. As for the GTF method, it enumerates many critical items in its generated summary, but it does not mention these items' values. As discussed in Section VI-A, the GTF method's performance is mainly limited by the template generation process. If the generated template does not add or add placeholders in wrong positions, the template filling step cannot produce quantitative descriptions correctly.

Some items mentioned in the target summary need numerical reasoning over tabular data. For example, the item "changes in our operating assets and liabilities" has many components. Although its value is not shown in the table, we can calculate it by adding up all its components. Some items like "non-cash charges" do not appear in inputs. To handle these more complex cases, the summarization model needs more knowledge about the relationships among all these items and better numerical reasoning ability.

VII. CONCLUSION AND FUTURE WORK

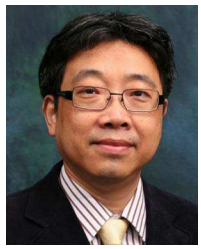
In this paper, we introduce FINDSum, the first large-scale dataset for long text and multi-table summarization. Besides, we propose a solution for the long text and multi-table summarization. It has three main steps: data pre-processing, content selection, and summarization. The content selection step aims to compress long inputs while maximizing the recall of salient content in long text and dozens of tables. As for the summarization step, we present and compare four types of summarization methods incorporating text and tabular data when summarizing report documents. Additionally, we propose a set of evaluation metrics to assess the usage of numerical information in produced summaries. Our summarization methods significantly outperform advanced baselines. Dataset analyses and experimental results indicate the necessity of incorporating textual and tabular data when summarizing report documents. In our extensive comparison experiments, we find some vital model components and configurations that can improve summarization results, including the content selection method, divide-and-conquer method, input sequence length, sparse attention mechanism, and pre-trained model. In the future, we intend to explore more methods and evaluation metrics for long text and multi-table summarization. There is still room to improve the produced summaries' quality and summarization methods' efficiency. Long text and multi-table summarization is still an open problem, and there is still a lot of work to do.

REFERENCES

[1] U. S. SEC, "How to read a 10-k/10-q," Jan. 2021. [Online]. Available: www.sec.gov/fast-answers/answersreada10khtm.html
[2] A. Vaswani et al., "Attention is all you need," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[3] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–16.
[4] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–16.
[5] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. Workshop Text Summarization Branches Out*, 2004, pp. 74–81.
[6] M. Grusky, M. Naaman, and Y. Artzi, "Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics-Hum. Lang. Technol.*, 2018, pp. 708–719.
[7] A. Fabbri, I. Li, T. She, S. Li, and D. Radev, "Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model," in *Proc. Assoc. Comput. Linguistics*, 2019, pp. 1074–1084.
[8] A. Cohan et al., "A discourse-aware attention model for abstractive summarization of long documents," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics-Hum. Lang. Technol.*, 2018, pp. 615–621.

[9] S. Liu, J. Cao, R. Yang, and Z. Wen, "Generating a structured summary of numerous academic papers: Dataset and method," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 4259–4265.
[10] L. Huang, S. Cao, N. Parulian, H. Ji, and L. Wang, "Efficient attentions for long document summarization," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics-Hum. Lang. Technol.*, 2021, pp. 1419–1436.
[11] W. Kryściński, N. Rajani, D. Agarwal, C. Xiong, and D. Radev, "BookSum: A collection of datasets for long-form narrative summarization," 2021, *arXiv:2105.08209*.
[12] M. El-Haj, A. AbuRa'ed, M. Litvak, N. Pittaras, and G. Giannakopoulos, "The financial narrative summarisation shared task (FNS 2020)," in *Proc. 1st Joint Workshop Financial Narrative Process. MultiLing Financial Summarisation*, 2020, pp. 1–12.
[13] X. Li, L. Du, and Y.-D. Shen, "Update summarization via graph-based sentence ranking," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 5, pp. 1162–1174, May 2013.
[14] X. Zhou, X. Wan, and J. Xiao, "CMiner: Opinion extraction and summarization for chinese microblogs," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1650–1663, Jul. 2016.
[15] W. Li and H. Zhuge, "Abstractive multi-document summarization based on semantic link network," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 1, pp. 43–54, Jan. 2021.
[16] J. Zhang, Y. Zhao, M. Saleh, and P. Liu, "PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 11 328–11 339.
[17] W. Chen, M.-W. Chang, E. Schlinger, W. Y. Wang, and W. W. Cohen, "Open question answering over tables and text," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–18.
[18] W. Chen, H. Zha, Z. Chen, W. Xiong, H. Wang, and W. Y. Wang, "HybridQA: A dataset of multi-hop question answering over tabular and textual data," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 1026–1036.
[19] F. Zhu et al., "TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance," in *Proc. Assoc. Comput. Linguistics*, 2021, pp. 3277–3287.
[20] P. Liang, M. Jordan, and D. Klein, "Learning semantic correspondences with less supervision," in *Proc. Assoc. Comput. Linguistics*, 2009, pp. 91–99.
[21] R. Lebret, D. Grangier, and M. Auli, "Neural text generation from structured data with application to the biography domain," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2016, pp. 1203–1213.
[22] S. Wiseman, S. M. Shieber, and A. M. Rush, "Challenges in data-to-document generation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2017, pp. 2253–2263.
[23] C. Rebuffel, L. Soulier, G. Scoutheeten, and P. Gallinari, "A hierarchical model for data-to-text generation," in *Proc. Eur. Conf. Inf. Retrieval*, Springer, 2020, pp. 65–80.
[24] R. Puduppully and M. Lapata, "Data-to-text generation with macro planning," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 510–527, 2021.
[25] F. Wang, Z. Xu, P. Szekely, and M. Chen, "Robust (controlled) table-to-text generation with structure-aware equivariance learning," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics-Hum. Lang. Technol.*, 2022, pp. 5037–5048.
[26] U. S. SEC, Form 10-k general instructions. 2023. [Online]. Available: www.sec.gov/about/forms/form10-k.pdf
[27] NARA, Regulation S-K item 303 management's discussion and analysis of financial condition and results of operations. 1982. [Online]. Available: www.ecfr.gov/current/title-17/chapter-II/part-229
[28] P. J. Liu, "Generating wikipedia by summarizing long sequences," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–18.
[29] R. Mihalcea and P. Tarau, "TextRank: Bringing order into text," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2004, pp. 404–411.
[30] G. Erkan and D. R. Radev, "LexRank: Graph-based lexical centrality as salience in text summarization," *J. Artif. Intell. Res.*, vol. 22, pp. 457–479, 2004.
[31] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statist. Interface*, vol. 2, no. 3, pp. 349–360, 2009.
[32] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.
[33] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2014, pp. 1532–1543.
[34] Q. Liu et al., "TAPEX: Table pre-training via learning a neural SQL executor," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–19.
[35] W. Guan, X. Song, H. Zhang, M. Liu, C.-H. Yeh, and X. Chang, "Bi-directional heterogeneous graph hashing towards efficient outfit recommendation," in *Proc. ACM Multimedia*, 2022, pp. 268–276.
[36] K. Choromanski et al., "Rethinking attention with performers," 2020, *arXiv: 2009.14794*.
[37] M. Zaheer et al., "Big bird: Transformers for longer sequences," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 17283–17297.
[38] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv: 2004.05150*.
[39] A. Gidiotis and G. Tsoumakas, "A divide-and-conquer approach to the summarization of long documents," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 3029–3040, 2020.
[40] M. Lewis et al., "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.
[41] M. Guo et al., "LongT5: Efficient text-to-text transformer for long sequences," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics*, 2022, pp. 724–736.
[42] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, no. 140, pp. 1–67, 2020.
[43] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. Conf. Empir. Methods Natural Lang. Process.*, 2020, pp. 38–45.
[44] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychol. Bull.*, vol. 76, no. 5, 1971, Art. no. 378.
[45] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proc. Assoc. Comput. Linguistics*, 2002, pp. 311–318.

Shuaiqi Liu received the BEng degree from Zhejiang University, Hangzhou, China, in 2019. He is currently working toward the PhD degree with the Department of Computing, the Hong Kong Polytechnic University. His research interests include document summarization, large language models, natural language generation, and natural language processing.



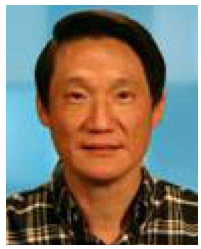
Jiannong Cao (Fellow, IEEE) received the MSc and PhD degrees in computer science from Washington State University. He is currently a chair professor with the Department of Computing at the Hong Kong Polytechnic University. He is the dean of Graduate School and the director of the Research Institute for Artificial Intelligence of Things with PolyU. His research interests include distributed systems, machine learning, and edge computing. He is a member of Academia Europaea and an ACM distinguished member.

Zhongfen Deng (Member, IEEE) received the bachelor's degree in automation from Chongqing University and the master's degree in control science and engineering from Chongqing University. She is currently working toward the PhD degree with the Department of Computer Science, the University of Illinois Chicago (UIC), USA. Her research interests include natural language processing and generation, and representation learning.

Wenting Zhao received the master's degree in software engineering from Shandong University. She is working toward the PhD degree with BDSC Lab, the Department of Computer Science, the University of Illinois Chicago (UIC), USA. Prior to joining UIC, Her research interests are natural language processing and few-shot learning, structured data-to-text generation.

Ruosong Yang received the BEng degree from the Beihang University in 2013, the MSc degree from the National University of Defense and Technology in 2016, and the PhD degree in computer science from The Hong Kong Polytechnic University in 2022. He is currently a postdoctoral fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include natural language processing and machine learning.

Zhiyuan Wen received the BEng degree from the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. He is currently working toward the PhD degree with the Department of Computing, the Hong Kong Polytechnic University. His current research interests include Open-domain dialog systems and text generation.



Philip S. Yu (Fellow, IEEE) received the BS degree in EE from National Taiwan University, the MS and PhD degrees in EE from Stanford University, and the MBA degree from New York University. He is a distinguished professor in computer science with the University of Illinois Chicago and also holds the Wexler Chair in Information Technology. Before joining UIC, he was with IBM, where he was manager of the Software Tools and Techniques department at the Watson Research Center. His research interest is Big Data, including data mining, data stream, databases, and privacy. He is a fellow of the ACM. He was the editor-in-chiefs of *ACM Transactions on Knowledge Discovery from Data* (2011-2017) and *IEEE Transactions on Knowledge and Data Engineering* (2001-2004).