Template-aware Attention Model for Earnings Call Report Generation

Yangchen Huang, Seyed Danial Mohseni Taheri and **Prashant K Dhingra**
J.P.Morgan Chase & Co., New York, NY
{yangchen.huang, seyeddanial.mohsenitaheri,
prashant.k.dhingra}@jpmchase.com

Abstract

Earning calls are among important resources for investors and analysts for updating their price targets. Firms usually publish corresponding transcripts soon after earnings events. However, raw transcripts are often too long and miss the coherent structure. To enhance the clarity, analysts write well-structured reports for some important earnings call events by analyzing them, requiring time and effort. In this paper, we propose TATSum (**T**emplate-**A**ware a**T**tention model for **Sum**marization), a generalized neural summarization approach for structured report generation, and evaluate its performance in the earnings call domain. We build a large corpus with thousands of transcripts and reports using historical earnings events. We first generate a candidate set of reports from the corpus as potential soft templates which do not impose actual rules on the output. Then, we employ an encoder model with margin-ranking loss to rank the candidate set and select the best quality template. Finally, the transcript and the selected soft template are used as input in a seq2seq framework for report generation. Empirical results on the earnings call dataset show that our model significantly outperforms state-of-the-art models in terms of informativeness and structure.

1 Introduction

Earnings Calls, conference calls held by public companies to disclose their performance of a specific period, are key resources in providing signals for financial analysts' decision-making process. Analysts, investors, and the mass media can learn about a company's financial results, operation details, and future guidance by listening to these conference calls. Previous works have highlighted the importance of earnings calls in modeling analysts' behavior (Frankel et al., 1999; Keith and Stent, 2019).

As nowadays firms disclose more and more information (Dyer et al., 2017), earnings call transcripts

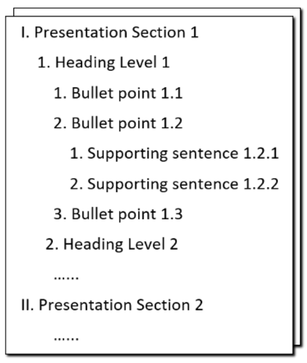
I. Presentation Section 1
1. Heading Level 1
1. Bullet point 1.1
2. Bullet point 1.2
1. Supporting sentence 1.2.1
2. Supporting sentence 1.2.2
3. Bullet point 1.3
2. Heading Level 2
......
II. Presentation Section 2
......

Figure 1: Example of an analyst report. Generated reports of our system follow the same structure.

are usually longer and contain more information than before, resulting in challenges in efficiently analyzing these unstructured documents and detecting informative facts. Some financial analysts write well-structured reports (Figure 1) (Refinitiv) for earnings calls after attending the event or reading the transcript. However, writing such reports usually takes time and effort. In addition, reports are not available for every company and event. Therefore, generating earnings reports quickly and automatically can fill the gap for no-report-available conferences and strongly accelerate the research process in the financial industry.

To fulfill this goal, we aim to develop an effective text summarization system to automatically generate reports with a hierarchical structure. Text summarization (Maybury, 1999), as an important field of Natural Language Processing, attracts considerable interest from researchers. Various datasets for summarization tasks have been built, most of which contains small to middle-sized document and short summary, e.g., CNN/Daily Mail (Hermann et al., 2015; Nallapati et al., 2016), WikiHow (Koupaee and Wang, 2018), Reddit (Kim et al., 2019), etc. Researchers design innovative architectures and benchmark the model performance on these mainstream datasets, yet extending summarization framework to the domain of earnings call transcripts, has never been explored.

In contrast to popular summarization tasks, structured earnings report generation has several challenges due to the special properties of data. First, earnings call conferences usually take a few hours, and transcripts contain thousands of words. This feature makes it impossible for popular pre-trained models such as BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) to provide high-quality summaries since these approaches partition the long documents into smaller sequences within 512 tokens to meet the input limit, resulting in loss of cross-partition information. Second, generated reports are required to be clearly organized and well-formatted. In addition to summarization, it is important for the model to recognize and output the explicit logical structure of the earnings call presentation. To the best of our knowledge, the structure quality of generated summaries from lengthy documents, in terms of logic and format, has rarely been examined in literature.

In this paper, we formally conceptualize report generation as an extension of summarization tasks and propose a novel approach, **T**emplate-**A**ware a**T**tention model for **Sum**marization (TATSum), to produce hierarchically-structured reports. Inspired by traditional template-based summarization (Zhou and Hovy, 2004), and soft template-based sentence summarization (Cao et al., 2018), we use historical reports as soft templates to provide supplemental structure information for a summarization system. We use soft templates as they do not enforce an actual rule in the generated summaries. To deal with the long sequence problem, we leverage the advantage of Long-Documents-Transformer (Longformer) (Beltagy et al., 2020), which reduces the complexity of the self-attention mechanism in Transformers (Vaswani et al., 2017) and allow for longer input sequences.

We collect historical earnings call transcripts and reports, and divide them into speaker sections with fewer words. The combination of a transcript section and a report section serves as an individual data point in the corpus. Our proposed framework consists of three modules as illustrated in Figure 2: (i) *Candidate Generation*, which generates a set of potential soft template candidates for a transcript section, (ii) *Candidate Ranking*, which ranks candidates through a Siamese-architected (Bromley et al., 1993) Longformer Encoder and selects the candidate with the highest rank as the final soft template for the transcript, and (iii) *Report Generation*, which generates the report using the soft template together with the raw transcript through a Longformer-Encoder-Decoder (LED) model (Beltagy et al., 2020). Figure 1 illustrates the structure of reports generated by our algorithm.

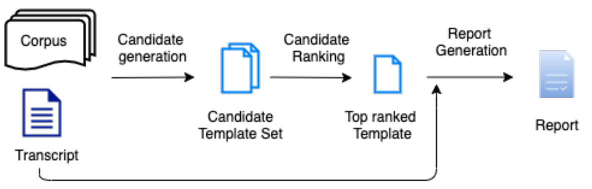

Figure 2: Flowchart of the report generation system

We evaluate the proposed framework on hundreds of earnings call events. Experiments show that TATSum significantly outperforms the state-of-the-art summarization models in terms of informativeness, format, and logical structure. Besides, extensive experiments are conducted to analyze the effect of different components of our framework on the performance of the model.

The contributions of this work are summarized as follows:

- We introduce a section-based soft template as supplemental information to the encoder-decoder framework to generate structured and readable earnings call reports.
- We design a Siamese-architected Longformer encoder for better template selection and further improve the quality of generated reports.
- Our algorithm adopts and extends the LED to provide template-aware summarization and overcome the challenge of long sequence encoding and long document generation.
- We conduct experiments on earnings call transcripts for the first time and evaluate the impact of different components of the proposed system. Results show that TATSum achieves superior performance compared with state-of-the-art baselines.

The paper is organized as follows: in Section 2, we review the relevant prior literature. Section 3 presents the novel architecture of TATSum. We conduct extensive experiments on the earnings call dataset and analyze the results in Section 4. Section 5 concludes the paper and provides future directions.

2 Related Work

Earlier studies of neural abstractive summarization employ encoder-decoder architecture to generate a shorter version of a sentence (Rush et al., 2015). Nallapati et al. (2016) extend previous work to summarize documents with more than one sentence using hierarchical attention. A variety of studies focus on building advanced attention mechanism for better summarization, e.g., convolutional attention (Chopra et al., 2016), graph-based attention (Tan et al., 2017), Bottom-up attention (Gehrmann et al., 2018), etc. See et al. (2017) propose a hybrid pointer-generator network and a coverage mechanism to keep track of already-summarized words. Paulus et al. (2018) introduce a deep reinforced model with a novel intra-attention mechanism and show improved performance for long document summarization.

Recently, pre-trained language models, which are trained to learn contextual representations from large-scale corpora, have been proved to be successful in summarization tasks. Popular pre-trained models like BERT (Devlin et al., 2019) and BART (Lewis et al., 2020) are adopted to build summarization-specific architecture. BERTSum (Liu and Lapata, 2019) proposes a novel document-level BERT-based encoder and an auto-regressive decoder with Trigram Blocking techniques and shows strong performance in both extractive and abstractive summarization. Aghajanyan et al. (2020) integrate BART with the Robust Representations through Regularized Finetuning (R3F) method to perform better fine-tuning for pre-trained models and achieve the state-of-the-art performance on CNN/Daily Mail.

Longformer (Beltagy et al., 2020) significantly reduces the time and space complexity of the attention mechanism and allows for much longer input sequences. It achieves this goal by replacing the self-attention in traditional Transformers (Vaswani et al., 2017) with windowed attention and introducing new task-oriented global attention. Longformer-Encoder-Decoder (LED) (Beltagy et al., 2020), a variant of Longformer, is also introduced for supporting long document seq2seq tasks. LED-large 16K, a BART-pretrained LED model with no additional pretraining, outperformed Bigbird summarization (Zaheer et al., 2020), a modified Transformer for long sequences with Pegasus pretraining (Zhang et al., 2020), and achieved the state-of-the-art performance on arXiv dataset (Cohan et al., 2018). In this paper, LED is adopted as the base model and a strong baseline benchmark.

In the domain of earnings calls, there is limited work exploring the potential of applying text summarization techniques. Cardinaels et al. (2019) generate an automatic summary for earnings releases using off-the-shelf unsupervised summarization methods such as KLSum (Haghighi and Vanderwende, 2009), LexRank (Erkan and Radev, 2004), etc., and conduct experiments to analyze the impact of automatic and management summaries on the investors' judgment. However, comprehensive experiments on the performance of summarization techniques are missing. In this work, we develop a novel summarization algorithm for report generation, provide extensive experiments on the earnings call dataset, and compare it with the state-of-the-art models in the literature.

An important feature of our system is to generate well-structured and formatted reports. Template-based summarization (Zhou and Hovy, 2004) is a traditional technique to summarize sentences. With a manually designed incomplete sentence template, the method fills the template using some input text, following pre-defined rules. This method can guarantee that the output sentence follows a specific format. However, constructing templates for long documents and large-scale datasets still remains challenging and requires domain knowledge. Cao et al. (2018) extended the template-based summarization and introduced a soft template, which is a summary sentence selected from the training set, to resolve this issue. Re3Sum (Cao et al., 2018) selects the soft template through an Information Retrieval (IR) platform and jointly learns template quality as well as generates the summary through a seq2seq framework. In this paper, we select historical reports from the corpus and form candidate sets.

To further improve template quality, our Candidate Ranking Module is inspired by MatchSum (Zhong et al., 2020). Zhong et al. (2020) formulates extractive summarization as a semantic text matching problem, and architect a Siamese-BERT network with margin-ranking loss to select the best candidate summary.

3 Method

Our goal is to generate hierarchically structured reports from long documents. The automatic generation system, TATSum, consists of three modules: *Candidate Generation*, *Candidate Ranking*, and *Report Generation*. Given an earnings call transcript, we divide it into different sections and consider each section as an input sequence T. This helps in the tractability of the summarization process by considering fewer tokens and results in well-structured reports since each section usually follows a coherent structure that is different from others. Similarly, human-written reports are divided into sections, R, and mapped to the document sections in the training phase.

For each document section T, the *Candidate Generation* module filters a candidate set of top i soft templates $\mathcal{C}^T := \{R_1, \cdots, R_i\}$ from a built corpus. We rank the candidate set $\mathcal{C}^T$ using the *Candidate Ranking* module to select the best soft template $\hat{R}^T$ to use. Finally, in the *Report Generation* module, the best soft template $\hat{R}^T$ and the raw document section T together are encoded into hidden states. A decoder takes the combination of encoded hidden states as well as decoder inputs to generate the abstractive report. Figure 3 illustrates *Candidate Ranking* and *Candidate Generation* modules. The three modules will be described in detail in the following subsections.

3.1 Candidate Generation

This module finds soft templates from the training corpus and forms a set of candidates. Our corpus, P, includes all document sections $\mathcal{T}$ and report sections $\mathcal{R}$ in the training set. To find the set of template candidates, we consider two assumptions: (i) similar transcripts should have similar reports, and (ii) a good template should give instructions about the format while not adding misleading information.

Since our dataset includes thousands of documents, we use an information retrieval technique, TF-IDF, to efficiently find the set of candidates. TF-IDF is a traditional unsupervised learning technique that can convert document text into a bag of words and quickly vectorize it. Since transcripts and reports have different styles, we consider the similarity between transcripts following assumption (i). Therefore, we first compute the similarity between section T and the other transcript sections in the corpus P using TF-IDF cosine similarity. Then, we select the top 5 scored document sections and use their corresponding reports in the corpus to form the candidate set $\mathcal{C} = \{R_1, \cdots, R_5\}$.

Although TF-IDF is a quick and easy method to calculate similarities and select candidates, it may not always provide candidates that resemble gold reports. We test this hypothesis by calculating the ROUGE average score (average of ROUGE-1, ROUGE-2, and ROUGE-l F1 score) between a sample of candidate sets and the human-written reports. We find that only 17.4% of the best TF-IDF candidates have the highest ROUGE average score among all the candidates, indicating that TF-IDF is not sufficient for retrieving the best soft-template. Thus, we add the second module, *Candidate Ranking*, to rank the candidate set and predict the best candidate that has the highest ROUGE score with the human-written report.

3.2 Candidate Ranking

The purpose of this module is to precisely select the best template, i.e., the template which has the highest ROUGE score with the human-written report. To train this module, we first calculate the ROUGE average score between each candidate template in the candidate set, $\mathcal{C}^T$, and the human-written report, R^{*T}, and store them in the descending order in $\tilde{C}^T$ as a label.

Inspired by Siamese network (Bromley et al., 1993) and Siamese-BERT structure (Zhong et al., 2020), we construct a Siamese-Longformer architecture to rank the candidate set. Longformer, a.k.a Long-Document Transformer,(Beltagy et al., 2020) is a model that successfully addresses the input length limitation of Transformer-based models by reducing the time complexity of the attention mechanism. The Siamese-Longformer model consists of two Longformer encoders with tied weights and a cosine similarity layer to compute comparable output vectors. One Longformer network encodes transcripts, T, and the other one encodes reports, R. We use the encoded hidden state of the bos_token '$< s >$' from the final Longformer layer to extract the transcript and report embedding vectors, e_T and e_R, respectively. The cosine-similarity layer connects these representation vectors and obtains the semantic similarity between the two documents.

$$S(T, R) = cosine(e_T, e_R)$$

If two documents have a higher ROUGE score, we expect them to have higher predicted semantic similarity.

We use margin-ranking loss to update the weights, and the model is expected to predict the correct rank of the candidate set based on the ROUGE score. Specifically, the loss function is

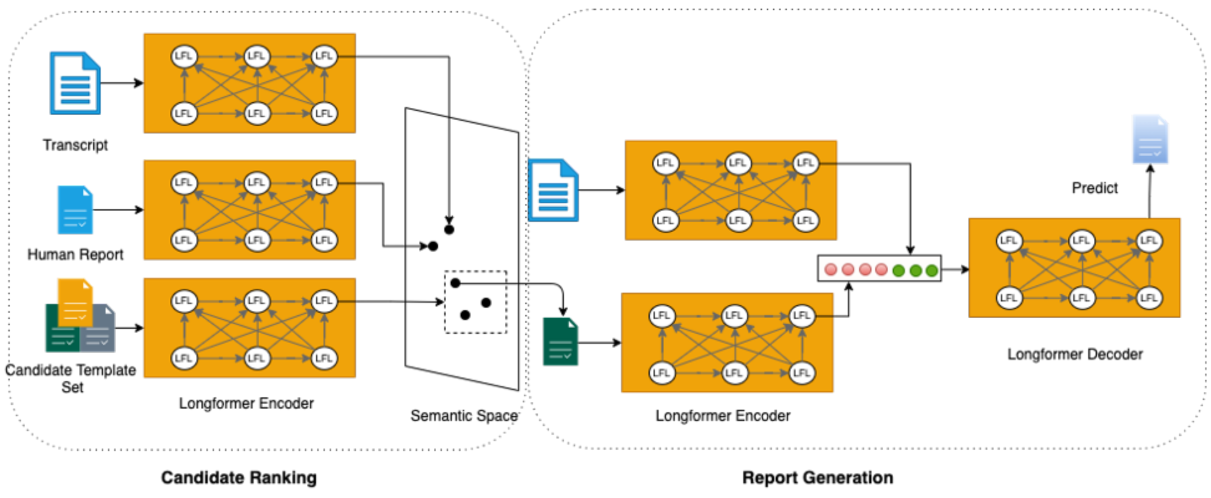

Figure 3: Illustration of different components of the *Candidate Ranking* module (left) and the *Report Generation* module (right). First, we learn parameters of the Longformer layers (LFL) in the *Candidate Ranking* to rank existing reports in the candidate set. Then, we encode the representations of the transcript and the top candidate using Longformer Encoder and decode their embeddings to generate reports.

designed from the following criteria.

- Human-written report should be the most semantically similar with the transcript
- A candidate template that has a higher rouge score with the human-written report should have a higher semantical similarity with the transcript.

Based on the first criteria, we derive the construction of the first loss function:

$$L_1 = \sum_{R \in \mathcal{C}^T} \max(0, S(T,R) - S(T, R^{*T}))$$

Where R^{*T} is the human-written report, and $R \in \mathcal{C}^T$ denotes all the candidate templates for transcript T.

Based on the second criteria, we use sorted candidate set ranking in $\tilde{\mathcal{C}}^T$ and design a margin-ranking loss as follows:

$$L_2 = \sum_{\{i,j\} \in \tilde{\mathcal{C}}^T} \max(0, S(T, R_j^T) - S(T, R_i^T) + (j-i)\epsilon) \quad (i < j)$$

where R_i denotes the candidate template ranked i, and ϵ is a hyperparameter that distinguishes between candidates with good, i, and bad, j, rankings. As described in criteria 2, the construction aims to measure the loss of any mis-ranking within the candidate set. Finally, the margin-ranking loss we use to train the Siamese-Longformer network is a combination of the two loss functions.

$$L_R = L_1 + L_2 \quad (1)$$

During the inference phase, the model will predict the similarity scores of candidates in the candidate set, and the candidate with the highest score will be set as the best soft template $\hat{R}^T$ for the transcript for further report generation.

$$\hat{R}^T := \arg\max_{R \in \mathcal{C}^T} S(T, R)$$

3.3 Report Generation

This module aims to generate the final report based on the soft template and the transcript. To generate an abstractive report, we design a soft-template-based encoder-decoder architecture to conduct seq2seq generation. We employ a pretrained LED as the base encoder-decoder model. The model takes a transcript section T and a soft template $\hat{R}^T$ as the input. They are tokenized and encoded by a Longformer encoder respectively. Similar to module 2, we use the encoded hidden state of '$< s >$' from the top Longformer layer as the representation of the corresponding transcript/template in the semantic space. The hidden states of the encoded transcript H_t and template H_s are concatenated as the final encoding outputs.

$$H_T = LongformerEncoder(T)$$
$$H_{\hat{R}^T} = LongformerEncoder(\hat{R}^T)$$
$$H_e = [H_T; H_{\hat{R}^T}]$$

The combined encoding outputs are then fed into a Longformer Decoder, and the decoding hidden state, H_d, is generated auto-regressively at position k based on the previous report tokens y_{k-1}:

$$H_{d,k} = Decoder(H_{d,k-1}, y_{k-1}, H_e)$$

Finally, a softmax layer predicts the probability vector of words at position k in the report:

$$\mathbf{P}_k = softmax(H_{d,k}W^P),$$

where W^P is learnable matrix. In cases in which the template includes too many tokens, we truncate it to ensure that we get more information from the transcript than the template since the main content in the report should source from the transcript, and templates should only provide information for formatting. Generally, the tokens from a template are about 25% of the transcript.

The whole encoder-decoder architecture is fine-tuned during training. A beam search is conducted during the test to generate an abstractive report that has the highest overall probability.

$$R = Beam(\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_k)$$

3.4 Optimization

We use two sets of loss functions in generating the reports of the earning calls. In *Candidate Ranking* module, we aim to train the parameters of Siamese-Longformer to find the best template in the candidate set. We use ranking loss, L_R, defined in equation (1) to achieve this goal.

In *Report Generation* module, our learning goal is to maximize the negative log-likelihood of the probability of the actual report.

$$L_G = -\sum_k log(p_{y_k})$$

We optimize the two losses separately over their respective parameters using gradient-based approaches (see section 4.4 for more details).

4 Experiment

We evaluate TATSum on Earnings call reports. We aim to answer the two following questions.

- Q1: How does TATSum perform compared to the state-of-the-art summarization systems?
- Q2: How do different components of TATSum such as soft-template and template ranking affect the performance of the model?

4.1 Dataset

We collected transcripts and human-written reports for 3655 earnings call events from 2017 to 2020, hosted by 1948 listed companies. Most selected companies are listed in NYSE or NASDAQ. For better generalization of our model, we also select a few companies from world-wide exchanges, such as TSX, FWB, Euronext, etc. These transcripts and reports are divided into 11141 sections, and each section is treated as an individual sequence. The statistics of our dataset is shown in Table 1. In our dataset, transcript lengths are significantly larger than the majority of public datasets such as Daily Mail and NYTimes and similar to long documents such as arXiv (Cohan et al., 2018). However, the reports in our dataset are substantially longer than the summaries of existing datasets, indicating that instead of doing lots of condensation, analysts tend to retain most of the information by paraphrasing and restructuring the oral transcript. Therefore, generating long sequences in natural language with a well-organized structure, is the most critical part for automatic earnings call reports.

Dataset	#docs	avg.doc. len(words)	avg. report len(words)
Docs	3655	3621	2524

Table 1: Statistics of the built earnings call dataset

We retain about 20% of the dataset as validation and use the rest for training. To prevent data leakage or utilizing future information, we test the performance of TATSum on 2000 transcript sections extracted from 666 earnings events in late 2020 and early 2021. In this setting, when generating the report for an earnings call transcript, TATSum only leverage a historical report that is available prior to the event. The entire dataset is shown in Table 2.

Dataset	Transcripts/Reports	Sections
Training Set	2929	8786
Validation Set	726	2265
Test Set	666	2000

Table 2: Document and section number information of the dataset

4.2 Evaluation Metrics

We employ ROUGE (Lin and Hovy, 2003) as the automatic evaluation metric. ROUGE has been used as the standard evaluation metric for machine translation and automatic summarization since 2004. The commonly adopted ROUGE metrics are ROUGE-1 (overlap of unigram), ROUGE-2 (overlap of bigrams), and ROUGE-l (Longest Common Subsequence, or LCS). In the Candidate Ranking module, we use an average of ROUGE-1, ROUGE-2, and ROUGE-3 F1 score as labels for training. Through testing of the whole system, we calculate and report all these metrics.

ROUGE can measure how much information

the generated report maintains compared with the human-written report. However, correctness, which can not be captured by ROUGE, is another important metric for the generated report in the domain of earnings calls. We manually review the rendered result, especially focusing on important financial statistics, trends, and sentiment, to evaluate whether it reports correct information from the earnings call.

4.3 Baselines

To compare the performance of our proposed model with others, we consider following state-of-the-art neural summarization baselines:

- **BERTSum**(Liu and Lapata, 2019): It uses document-level BERT-based encoder and an autoregressive decoder with Trigram Blocking for summarization.
- **LED** (Beltagy et al., 2020): It modifies the self-attention in Transformers with windowed attention for long document summarization.

4.4 Implementation Details

We implement our model with Pytorch and adopt the pretrained LED-base model in *Candidate Ranking* and *Report Generation* modules. In particular, we use LEDs with 768 hidden state and 3072 feed-forward layer dimensionality. We use dropout with probability 0.1 and a customized Adam optimizer (Kingma and Ba, 2014) ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1e - 9$) during training. The learning rates through optimization follow Noam decay scheme (Vaswani et al., 2017) with a warmup step of 500 and are set to be:

1. Module Candidate Ranking:

$Lr = 3\text{e}{-}3{*}\min(step^{-0.5}, step{*}wrm.steps^{-1.5})$

2. Module Report Generation:

$Lr = 3\text{e}{-}5{*}\min(step^{-0.5}, step{*}wrm.steps^{-1.5})$

We save a model checkpoint every 5000 steps and choose the best-performed checkpoint on the validation set. In *Report Generation* module, we use the Block Trigram technique (Liu and Lapata, 2019) to reduce potential redundancy. However, we find this approach ineffective for some reports and observe repetitions of words with punctuations in between. Therefore, we add a new Block Tri-word method that forces the decoder never to output the exact same three words in a predicted sequence with all punctuations deleted. When the decoder creates the same three words that exist in the previous pure word sequence, the probability of the beam is set to be 0.

Although we employ the Longformer architecture to deal with long sequences, we still face memory challenges in *Report Generation* module when the earnings call section is too long. To increase the performance for earnings call sections of arbitrary length, we divide a long section into several short sub-sections and generate reports for each sub-section. We then combine each sub-section and report them in the same hierarchical structure. This method is proved to perform well when a transcript section exceeds the sequence length limit.

We search for best hyperparameters for baselines, and use optimization schemes suggested by authors. Models are trained on Tesla-V100 GPUs.

4.5 Experiment Results

Table 3 shows the ROUGE F1 score for different methods.

Metrics	ROUGE1	ROUGE2	ROUGE-l
BERTSum	36.89	22.16	35.40
LED	65.17	53.07	64.93
TATSum	**76.20**	**61.89**	**75.94**

Table 3: Results of TATSum and Baseline models

Due to the small input length limit in the architecture of BertSum, it is not able to generate fluent and readable reports by partitioning the long document and combining the output. Taking advantage of the modified attention mechanism and huge sequence length limit, LED, on the contrary, achieves quite good performance. All three Rouge-F scores are above 50, indicating that reports generated by LED can extract critical information from earnings presentations similar to human beings. By adding a precisely-selected soft template to LED, our proposed system, TATSum, boosts the report quality even more, with a significant improvement over the performance of LED by 17%.

As discussed in section 4.1, earnings call reports contain much more words than summaries in popular summarization datasets. Therefore, the ROUGE scores are higher than those observed from CNN/DM and arXiv correspondingly. In order to guarantee the quality of generated reports for real use cases, in terms of structure and accuracy, we further conduct manual checking on a small sample of earnings events selected from the test set. We read their transcripts, human-written reports, and automatic report generated by TATSum, and compare the content in these documents. Generated reports mimic the analyst report well in format and structure. For information accuracy, we primarily focus on the numbers, trends, and sentiment in generated reports. Our observation shows that except in a few cases where some parts are missing, information within the generated report is accurate and coherent.

4.6 Ablation Study

In this section, we analyze variants of our model to find the effect of different components on the model performance. We consider variations as follows: (1) *No template*: we remove the first and second modules and consider a pure LED architecture for report generation and (2) *NoRanking*: We forgo *Candidate Ranking* module and use the template with the highest TF-IDF cosine similarity in *Candidate Generation*.

In table 4, we report the average ROUGE score of generated reports on the test set under different experiment setting.

Metrics	ROUGE1	ROUGE2	ROUGE-l
NoTemplate	65.17	53.07	64.93
NoRanking	75.90	61.48	75.63
TATSum	**76.20**	**61.89**	**75.94**

Table 4: Ablation study of design choices in TATSum

Effect of soft template: To capture the impact of soft templates on the performance of our model, we compare the results of *NoTemplate* and *NoRanking*. As illustrated in Table 4, A soft template based seq2seq model achieves significantly higher ROUGE scores. In addition to boosting the performance, incorporating the template stabalize the training process and results in faster convergence, indicating that the model can better learn to write reports in a quicker manner with supplemental information. We also compare several reports generated by the two models, and find that the report of *NoRanking* model has better format and logical structure. The report is clearly organized, with good heading levels and correct serial numbers. In contrast, the report of *NoTemplate* contains more incorrect indentations, levels, and serial numbers. It proves that adding a soft template do provide the model with more information on how to write a report logically like a human.

Effect of template ranking: Similarly, results of *NoRanking* and TATSum are compared to study whether ranking the candidate set of templates improves the performance. As shown in Table 4, ranking the candidate set and selecting a template of better quality can slightly increase all three ROUGE scores of generated reports. It is worth mentioning that unlike adding a soft template, ranking the candidate set can take a longer time for labeling and training. For labeling, ROUGE scores need to be calculated for each template in the candidate set with the human-written report, and all data points in the training set should be labeled. For training, a Siamese-Longformer encoder is constructed to predict the rank of the candidate set, which also requires long training and validation time. Therefore, further thoughts on balancing the tradeoff between performance and training time are necessary for each dataset.

5 Conclusions

This paper proposes an innovative neural summarization system, TATSum, with three modules, Candidate Generation, Candidate Ranking, and Report Generation, to generate structured reports automatically. In Candidate Generation module, we build a corpus of historical documents and reports, and for each document, we generate a candidate set using quick and easy similarity-based criteria. The candidate set is then ranked in the Candidate Ranking module, following the predicted result of an encoder model with margin-ranking loss. We choose the candidate with the highest rank as the soft template. In the final Report Generation module, we encode both template and document into hidden states and feed the combined hidden states into a decoder to generate the report. Extensive experiments are conducted on the earning call dataset and show that our model can generate reports with high informativeness (ROUGE) and high accuracy (numbers, trends, etc.). We also prove that adding a template can significantly improve the quality of the generated report, and finely selecting a template with good quality can increase performance even more.

We mainly test TATSum on automatic report generation for earnings call events. However, the advantage of Longformer architecture for long sequence tasks, as well as the significant power of adding soft templates for structured document generation, can extend our proposed framework to various domains, e.g., medical report, employee annual review, call center record, etc. We would like to take advantage of this proved architecture to explore more potential in structured report generation.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2020. Better fine-tuning by reducing representational collapse. *arXiv preprint arXiv:2008.03156*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6:737–744.

Ziqiang Cao, Wenjie Li, Furu Wei, Sujian Li, et al. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. Association for Computational Linguistics (ACL).

Eddy Cardinaels, Stephan Hollander, and Brian J White. 2019. Automatic summarization of earnings releases: attributes and effects on investors' judgments. *Review of Accounting Studies*, 24(3):860–890.

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

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 615–621.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Travis Dyer, Mark Lang, and Lorien Stice-Lawrence. 2017. The evolution of 10-k textual disclosure: Evidence from latent dirichlet allocation. *Journal of Accounting and Economics*, 64(2-3):221–245.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Richard Frankel, Marilyn Johnson, and Douglas J Skinner. 1999. An empirical examination of conference calls as a voluntary disclosure medium. *Journal of Accounting Research*, 37(1):133–150.

Sebastian Gehrmann, Yuntian Deng, and Alexander M Rush. 2018. Bottom-up abstractive summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4098–4109.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of human language technologies: The 2009 annual conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pages 1693–1701.

Katherine A Keith and Amanda Stent. 2019. Modeling financial analysts' decision making via the pragmatics and semantics of earnings calls. *arXiv preprint arXiv:1906.02868*.

Byeongchang Kim, Hyunwoo Kim, and Gunhee Kim. 2019. Abstractive summarization of reddit posts with multi-level memory networks. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2519–2531.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *arXiv preprint arXiv:1810.09305*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pretraining for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram cooccurrence statistics. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 150–157.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Mani Maybury. 1999. *Advances in automatic text summarization*. MIT press.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Caglar Gucehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *International Conference on Learning Representations*.

Refinitiv. https://www.refinitiv.com/en/financial-data.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2017. Abstractive document summarization with a graph-based attentional neural model. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1181.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30:5998–6008.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuan-Jing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208.

Liang Zhou and Eduard Hovy. 2004. Template-filtered headline summarization. In *Text summarization branches out*, pages 56–60.