BigBirdFLY: Financial Long text You can read

Stanford CS224N Custom Project — Mentor: Rui Yan

Dhaval Dangaria, Riccardo Giacomelli, Wilfrido Martinez
{dhavald, ricgiac, wilfrido.martinez}@stanford.edu

Abstract

The development of new architectures allows to process long input windows of text at once, overcoming both memory and computational constraints. New developments pushed maximum input windows to 65k+ words compared to the 512 BERT limit. We aim to explore, compare and improve state-of-the-art long window architectures to summarize long texts. We consider BERT, BigBird and GPT-3 models. We focus on the financial narrative domain, summarizing 100- to 200-page documents. We aim to test models with different maximum input size exploring benefits and limitations. Long input windows allow to include wider context in the summarization process, avoiding out-of-context sentence extraction that can lead to changes in sentence-level semantic. We compare extractive and abstractive methods on key aspects in the financial context as numerical accuracy and summary semantic. We show extractive methods (BERT-based) can retain sentence by sentence accuracy from text, nevertheless the extraction process can produce fragmented summaries which can lead to misleading interpretation. We also reveal abstractive methods (by introducing BigBirdFLY, a wide context summarization method based on BigBird) can produce fluent summaries. By using human evaluation, we show BigBirdFLY can produce summaries more similar to human-generated summaries and excel in the human evaluation criteria, whereas extractive methods are able to score high in automatic metrics (ROUGE). Finally, we explore how enhanced greedy sentence-selection methods exploiting long input window in a single step compare to recursive solutions based on Reinforcement Learning.

1 Introduction

Long text summarization compresses the source text (100 pages) into a watered-down (yet fluent and self-contained) version (1,000 words) that keeps its information content and overall meaning [1]. Existing research can be classified into one of two approaches to text summarization: 1) Extractive, where sentences are extracted from the text and used to represent it; and, 2) Abstractive, where you produce a new body of text to represent it. Historically, both tasks have been relatively difficult, even for neural approaches [2], with long text adding additional complexity to the task.

Specifically, in the financial domain sentence extraction-based summarization without considering broad context can be misleading; moreover, highly relevant numerical information could be meaningful only if related with context. E.g., in the following extract from a report:

> *Assuming the progressive increase of oil price to $60 per barrel, as per our scenario assumption*
> ...
> *cash flow from operation before working capital is expected to amount to around €44 billion along the plan period.*

the second sentence has a different meaning depending on if it is related or not to the first sentence. Extracting the second sentence would lead to the wrong conclusions. The semantic meaning of the original text can change depending on the summary length (i.e. a generated summary of 2,000 words could have a very different meaning compared to a summary of 1,000 words generated from the same text.) Moreover, two sentences located far apart in the original text could be placed close by in the summary, creating fictional meaning. The introduction of longer-context windows (4k+ tokens) allows to consider the full context of the sentence in the summarization process, avoiding deceptive and detrimental summaries. Additionally, long context windows allow the model to have flexibility to rank relevance of text considering a context breakdown at the section/sentence level.

In this work, we aimed to compare the performance of extractive/abstractive models at the sentence level based on BERT (maximum 512 tokens), GPT-3 generated summary (maximum 2,048 tokens), and propose a new solution with a maximum input window 8 times longer based on BigBird [4] (maximum 4k tokens.) We compared them using both standard metrics and human evaluation. The aim is to, eventually, expand the study to longer context windows [5].

In the domain of sentence selection, solutions spanning from greedy (local sentence selection) to recursive multi-stage sentence selection (Reinforcement Learning) [3] have been proposed. This study is an empirical test on how greedy algorithms can be enhanced exploiting long window input sizes and how they compare to recursive ones [3] and others methods [6] for the summarization of long text financial documents.

2 Related Work

There have been several algorithmic approaches for both extractive and abstractive summarization, from the simplest extractive approach (LEAD-3 [13] benchmark, which proposes the first three sentences of the document as the summary, and achieves an R-1 F1 score of 30 percent,) to more complex ones, like MUSE [9] benchmark (based on rules, classification for sentences — R-1 of 48 percent), sumTO [6] benchmark (a sentence by sentence approach based on BERT — R-1 of 45 percent), and Pointer Networks [3] benchmark (R-1 of 46 percent.) Other approaches to text summarization include, topic words, bayesian topic models, and graph based approaches [16].

Transformer-based models (together with language model pre-training) have proved a game-changer for Machine Learning and NLP [13]. Both BERT and BigBird have been used extensively in different fields for many NLP tasks. BERT has proven to be a key advancement in NLP by achieving state-of-the-art results in many NLP taks, such as question answering and natural language inference [14]. In order to fine-tune BERT for a specific task, only one additional layer is necessary for the pre-trained BERT model, as opposed to substantial architecture modifications. Use of transfer learning has become a norm for state-of-the-art research using BERT [14]. Thus, for summarization, the architecture of the original BERT model is only subjected to small modifications. In BERT, the proposed sequence is [15]: input document, followed by the summation of three kinds of embeddings for each token. The summed vectors are used as input embeddings to several bidirectional Transformer layers, generating contextual vectors for each token. For summarization (Figure 3), BERT is extended by inserting multiple [CLS] symbols to learn sentence representations and using interval segmentation embeddings to distinguish multiple sentences. finBERT has used BERT for the financial domain. It proved good for extracting explicit sentiments, but modeling explicit information was not apparent [17].

BERT has been one of the most successful deep learning models for NLP, but a core limitation is the quadratic dependency (mainly in terms of memory) on the sequence length due to their full attention mechanism [4]. To remedy this, researchers developed BigBird, a sparse attention mechanism that reduces this quadratic dependency to linear. Because BigBird is relatively new, there is no extensive literature on its use for summarization on financial corpus, but it can handle sequences of length up to 8x (and more) of what was previously possible using similar hardware (i.e. BERT.) As a consequence of the capability to handle longer context, it is hypothetized that BigBird drastically improves performance summarization [4]. In the context of long windows, many solutions have been proposed. We have chosen BigBird in particular because it achieved SOTA level on different summarization tasks. For Human evaluation we included the GPT-3 [21] to check output as well. It is not tuned to achieve long text summaries, but can generate per-section summaries.

While many methods have been proposed for human evalution in NLP (e.g. Pyramid Evaluation, DUC), it remains a difficult task due to the subjectivity involved [25]: 1) because of the subjectivity of assessing the summarization criteria — the agreement between human evaluators is low (and so we do not have a reliable baseline to report); and, 2) because of the amount of effort required to evaluate the summaries — very time-consuming (considering, also, that manual assessment is not reusable.) `ROUGE` has shown correlation with human summary evaluation (albeit low) [26], even when using simple human evaluation (i.e. comparing a manual summary to the gold standard.)

3 Approach

While both `BERT` and `BigBird` have been used for summarization, we extended their capabilities to address the task of summarizing finance-specific long text. Our code, and, outputs can be found on Github. We also used human evaluation to obtain a qualitative understanding of the condition of our summaries. We selected two main architectures: `BERT` and `BigBird`, because they are general architectures and have achieved `SOTA` on multiple summarization benchmarks [19, 20]. Furthermore `BERT` has been pre-trained for financial text analysis, and `BigBird` can leverage weight from `Pegasus`, a model pre-trained specifically for summarization.

3.1 BERT

`BERT` [2] uses a standard `BERT` architecture pre-trained on financial data. It utilizes only the encoder for classification (Figure 1) using 12 layers — 768 hidden state, 12 layers, 12 heads, 3,072 feed-forward parameters — with 512 input tokens. `BERT` is bidirectional (i.e. its self-attention layer performs self-attention on both directions.) The sentence-by-sentence classification is conducted by adding a dense layer (feed forward network with two layers, 64/32 or 128/64, and final single output) after the last hidden state of the `[CLS]` token [15]. Only the weights of the classification layer are fine-tuned during training. To do this, we require a training set that pairs individual sections of a document with a positive (include) or negative (don't include) label depending on whether or not the gold-standard summary is derived from it. If selected, the sentence is reported unchanged in the summary. We employ a sentence-overlap method to identify a single section as the "summary section" section, and label all other sections as negative. Specifically, we regard one sentence as overlapping a report section if most of its words appear in that section. When there are multiple gold standard summaries for a financial report, we choose the summary with the highest sentence overlap rate as the gold standard [8]. In our experiments, we used a checkpoint from `BERT`, reserving for future work loading weights from the `finBERT` checkpoint.

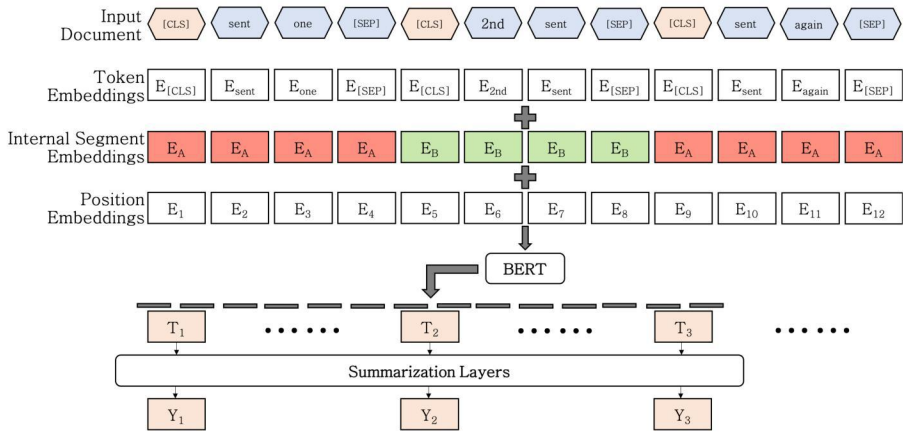

Figure 1: BERT Encoder. The encoder is the pretrained `BERT`-based encoder from the masked language modeling task [23]. The task of extractive summarization is a binary classification problem at the sentence level. We assign each sentence a label indicating whether the sentence should be included in the final summary. Thus, add a token [CLS] before each sentence. After we run a forward pass through the encoder, the last hidden layer of these [CLS] tokens will be used as the represention for our sentences. *Adapted from [24].*

3.2 BigBird

Unlike BERT, which runs on a full self-attention mechanism,BigBird works on a sparse attention mechanism (for the encoder or decoder) that allows it to overcome the quadratic dependency of BERT, while preserving the properties of full-attention models. This property is justified by random graph theory [4]. Intuition is that even if the attention matrix is not full, the graph of the tokens links are strongly interconnected (high cluster coefficient) [18]. The full self-attention matrix (NxN) — every token attends to itself and all other tokens — is decomposed into an arrow block that attends to every token and every token attends to itself (global tokens), plus a band block around diagonal of current token attending itself, plus random connection between tokens (configured in this setting to 3 each row) as in Figure 2. The resulting total computational time is reduced from N^2 to $20N$. The sparse computation can be packed into a full matrix computation that can run quickly on GPU/TPU. Because of this, BigBird can process sequences 8x times longer than BERT (We used a maximum of 4,096 input tokens, 256/512 output tokens in the decoder, 16 heads, 16 layers, 1,024 hidden size, fed forward size of 4,096.) BigBird for summarization is a seq-to-seq model. For the base model, weights are shared between encoder and decoder, whereas for large models weights are leveraged from Pegasus [9] (SOTA model pre-trained specifically for summarization tasks.) Pegasus is pre-trained as a masked language model (MLM), novelty is to mask most relevant sentences ranked according to ROUGE-1 F1 between a sentence and every other sentence. It is very effective in summarization of benchmarks, and in particular of small datasets [4]. Encoder and decoder weights are not fine tuned. Cross encoder-decoder attention weights are calibrated during fine tuning.

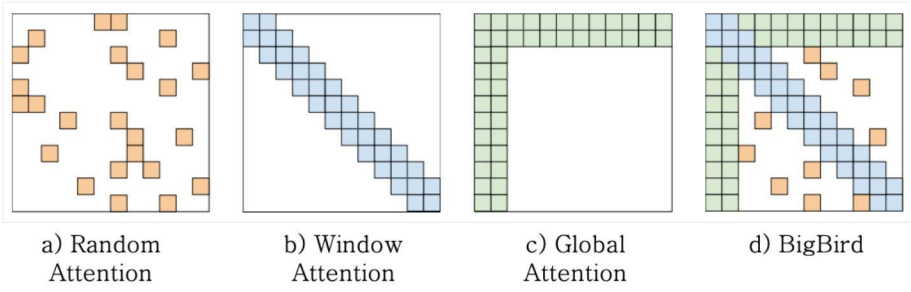

Figure 2: Building Blocks of the Attention Mechanism used in BigBird. White color indicates absence of attention; a) random attention with $r = 2$; b) sliding window attention with $w = 3$; c) global attention with $g = 2$. (d) the combined $BigBird$ model. *Adapted from [4].*

The code [29] is specialized to run on Google Cloud TPU utilizing TPU Estimator framework [27], for Tensorflow 2. The GitHub repository is missing evaluation and prediction scripts necessary to generate the predicted summaries, which we added. The data converted to the tfrecords format is loaded on Google Buckets and the code run on Google VM. The executing code is passed to the host connected to TPU [28]. After the initial time needed to set up the code and infrastructure, build an understanding and write the code, the training time is reduced linearly with the number of TPU cores. All training was run on TPU v2.8-512 and v3.8-8 under the TFRC program [30]. Minimum training+valuation time was reduced to 30 minutes compared to ∼64+ hours needed on a standard GPU.

3.3 GPT-3

Summaries from GPT-3 are produced as best of 5 results, appending command tl;dr to the input text. GPT-3 is a pre-trained multi-task with 175B parameters, and the model is capable of in-context-learning. It has a maximum window size of 2,048 tokens, a midpoint between the 512 tokens of BERT and the 4k tokens of BigBird. It has a standard Transformer architecture with a few modifications. Similar to Spars transformers [22], it utilizes alternating dense and locally banded sparse attention patterns.

3.4 BigBirdFLY

We propose a new solution based on BigBird (Figure 3). BigBird is used at different stages and pre-trained for two different tasks. The first task (A in Figure 3) classifies the relevance of a single

section in the full document. For this task, the encoder is coupled with a single layer connected to the first encoded `[CLS]` token of the text. In output, both binary classification and probability can be extracted. This way, the relevance of sections can be ranked continuously from 0 to 1. In the second stage (B in Figure 3) `BigBird` summarizes every section. The summaries are then merged into the final full summary. Details about parameters and checkpoints used are provided in section **4 Experiments.** A further step — left as future research — is to extract the relevance of single blocks of text from attention weights, so then block relevance is coupled with section relevance to output context-aware hierarchical relevance ranking of blocks of text.

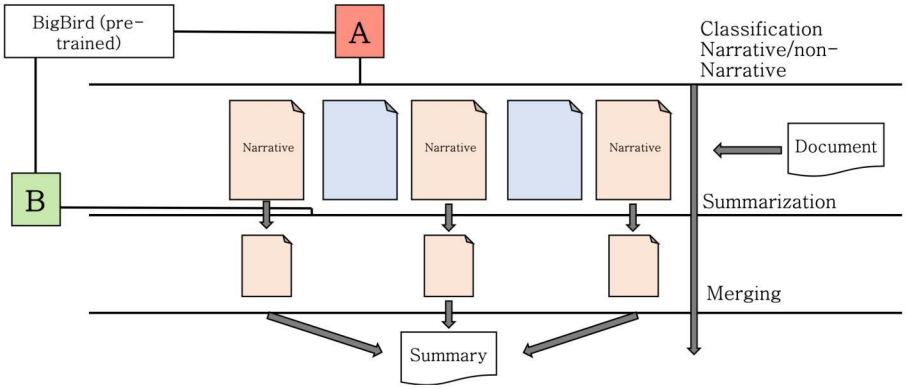

Figure 3: BigBird Data Flow. In **A** BigBird classified the relevance of a single section in the full document. In **B**, BigBird summarized every section.

3.5 Human Evaluation

To obtain a better understanding of the summaries our models produced, we performed simple human evaluation. Given the time constraints we faced and our limited human resources, we randomly selected 12 samples, ranging from 500 to 7,500 words (i.e. we did not select whole 100-page documents, but rather random subsections of the random samples). We obtained a gold standard from a third party (i.e. an expert in finance summarized the samples), and then, team members summarized each of them. Because most approaches use multiple individuals for manual summarization in an attempt to reduce variability, each sample was summarized by two team members. Thus, each team member independently summarized 8 of the samples (i.e. each sample was summarized twice — a time-consuming endeavor.) At the same time, we summarized these samples using both `BERT` and `BigBird`. To obtain a quantitative understanding of our results, we computed `ROUGE-1 F1` for these samples, and measured the time it took to obtain the summaries. To obtain a qualitative understanding of our results, we (blindly) compared our summaries against those produced by the models. In particular, we looked at four things for our qualitative assessment of the summaries: 1) fluency — whether the summary was readable and self-contained; 2) length and relevance — whether the summary is concise and finds the key points instead of less important/random facts; 3) interpretability — whether it was easy to understand the summary; and, 4) overall meaning — whether, as seen in the introduction, the summary preserves the facts and does not change meaning of original document.

4 Experiments

4.1 Data

We used annual reports produced by UK firms listed on The London Stock Exchange (LSE)[1]. We used 3,863 annual reports divided into training, testing, and validation. Full text of each annual report along with the extracted sections and gold standard summaries are provided. [2]. On average there are at least 2 gold standard summaries per annual report. In this task we produced one summary for each annual report, whose lengths should not exceed 1,000 words. For all sections, we computed an average of 2.5k tokens, a median of 986, a 90 percent quantile of 5,500, and a maximum of 260k.

[1] http://multiling.iit.demokritos.gr/pages/view/1648/task-financial-narrative-summarization

[2] Gold summaries are missing for the test set

We fine-tuned BigBird on a different version of the dataset. We used almost raw data (data_v1), and also data_v2, which includes processed data with further cleansing including only narrative sections and further text cleaning. The output did improve, but there were repeated words, and summaries were rather short. For data_v3, only the best summary for each document was kept. Best summary was selected based on R-2 recall and R-L recall. The matching of sub-parts of the summary to every section was improved as well, by limiting the minimum and maximum length of each sentence. Furthermore, 5 percent of the worst summaries were discarded from training, as well as the worst 5 percent of all sub-summaries matched to sections. We did this to exclude possible errors in both summary matching and summary selection. The total, final number of examples in data_v3 are 11,265. To train models with a larger decoder output (512) we used a minimum-length summary in the training set. data_v4 and data_v5 are filtered with a minimum of 100 and 200 words, respectively. The total examples in data_v5 are 2,100. To test excess cleaning we also utilized a data_v6 that includes a 200-word minimum summary rule but less cleaning rule than data_v5, as in Table 1.

Table 1: Fine-tuning of BigBird on different versions of the dataset.

Data	Number of Examples	Cleaning Rules (Data Preprocessing)	N Summaries Considered per Document	Sub-Summary Matching Method	Discarded Samples
v1	67k	-	2 to 3	R-1	0%
v2	25k	1 to 3	2 to 3	R-1+R-2	0%
v3	11k	1 to 8	1	R-1+R-2+R-L+min/max sentence length	5% of summaries, 5% of sections
v4	7.5k	1 to 8	1	R-1+R-2+R-L+min/max sentence length	5% of summaries, 5% of sections; min. summary length of 100 words
v5	2.1k	1 to 8	1	R-2+R-L+min/max sentence length	5% of summaries, 5% of sections; min. summary length of 200 words
v6	2.5k	1 to 3	1	R-2+R-L+min/max sentence length	5% of summaries, 5% of sections; min. summary length of 200 words

4.2 Data Preprocessing

We cleaned the data cleaning using Python's regular expressions and packages ftfy [11], cleantext [12] and includes the following patterns:

1. delete sequences of 3+ symbols
2. insert space between numbers+letters (40USD), letters+numbers (eps0.2)
3. delete multiple spaces and new lines
4. reduce to 1: 3+ consecutive non-alphanumeric symbols (£££ # # #)
5. delete 4+ consecutive all upper-case words
6. keep first of repeated sentences
7. delete sentences with more than 50% upper-case letters
8. delete sentence with alphanumeric words<50%

We converted each version of the dataset to tfrecords files for training.

4.3 Evaluation Method

We are using the automatic evaluation method ROUGE (Recall-Oriented Understudy for Gisting Evaluation), including ROUGE-1, ROUGE-2, ROUGE-3, and ROUGE-4 [7]. The measures count the number of overlapping units such as n-gram, word sequences, and word pairs between the computer-generated summary to be evaluated and the ideal summaries created by humans. ROUGE-N refers to the overlap of N-grams between the system and the reference summaries. For example, ROUGE-1, ROGUE-2 refer respectively to the overlap of unigram (each word) and bigrams between the system and reference summaries. We also included F1. We measured human performance to obtain a more qualitative understanding of our results using ROUGE-1, (as described in the Human Evaluation section above.)

4.4 Experimental Details

4.4.1 BERT

We used PyTorch, OpenNMT and the 'bert-base-uncased' version of BERT to implement summarization. Both source and target texts were tokenized with BERT's subwords tokenizer. When predicting summaries for a new document, we first use the model to obtain the score for each sentence, and then rank these sentences by their scores from highest to lowest, and select the top-3 sentences as the summary. Our code can be found here.

During sentence-selection, we use Trigram Blocking to reduce redundancy. Given summary S and candidate sentence c, we skip c if there exists a trigram overlapping between c and S. We want to minimize the similarity between the sentence being considered and sentences which have been already selected as part of the summary.

Let d denote a document containing sentences $[sent_1, sent_2, \cdots, sent_m]$, where $sent_i$ is the i^{th} sentence in the document [15]. Extractive summarization can be defined as the task of assigning a label $y_i \in 0, 1$ to each $sent_i$, indicating whether the sentence should be included in the summary. It is assumed that summary sentences represent the most important content of the document. In the summary layer, vector t_i which is the vector of the i^{th} [CLS] symbol from the top layer can be used as the representation for $sent_i$. Several inter-sentence Transformer layers are then stacked on top of BERT outputs, to capture document-level features for extracting summaries:

$$\hat{h}^l = LN(h^{l-1} + MHAtt(h^{l-1})) \tag{1}$$

$$h^l = LN(\hat{h}^l + FFN(\hat{l}) \tag{2}$$

where $h^0 = PosEmb(T)$; T denotes the sentence vectors output by the summary layer, and function PosEmb adds sinusoid positional embeddings to T, indicating the position of each sentence. The sigmoid classifier is:

$$\hat{y}_i = \sigma(W_o {h_i}^L + b_o) \tag{3}$$

where ${h_i}^L$ is the vector for $sent_i$ from the top layer (the L^{th} layer) of the Transformer. The loss of the model is the binary classification entropy of prediction $\hat{y}_i$ against gold label y_i . Inter-sentence Transformer layers are jointly fine-tuned. Adam optimizer with $\beta 1 = 0.9$, and $\beta 2 = 0.999$ is used. The learning rate follows [31] with warming-up ($warmup = 10,000$):

$$lr = 2e^{-3} \cdot min(step^{-0.5}, step \cdot warmup^{-1.5}) \tag{4}$$

It took 2 hours to generate a summary for 363 documents. All details (e.g. layers) are presented in section **3.1 BERT**.

4.4.2 BigBirdFLY

We refer to Figure 3 for the experiments. We fine-tuned BigBird starting from the three checkpoints in Table 2.

Table 2: BigBird Fine-tuning Checkpoints with Different Versions of the Dataset.

Checkpoint	enc/dec	Encoder	Decoder	ff	heads/layers/h. size	Task
Roberta-L	e	4,096	256/512	3,072	12/12/768	classif.
Roberta-Base	e+ d	3,072	-	3,072	12/12/768	summary
Pegasus-L	e+ d	3,072	256/512	4,096	16/16/1024	summary

For classification (Figure 3), using only the encoder, we were able to use a maximum encoder input size of 4,096. For summarization, we used a maximum encoder size of 3,072 to fit into 16GBs of memory. For classification, every text section is classified into a binary variable (narrative/non-narrative.) Labels (0-1)/logits are the model's output. The initial checkpoint is Roberta-L. Table 3 reports the precision of the best fine-tuned results on the evaluation set.

For summarization we report 5 experiments with different cleansing routines and model parameters. For every section, a corresponding matched summary was extracted from the full document summary

Table 3: Precision of Best Fine-Tuned Results.

Classification	Accuracy	Step	Batch Size	Loss	Training Time	Evaluation Time
BigBird-L	92.30%	1,117	512	0.038	15m	15m

— non-narrative sections do not appear in final summary. A matching algorithm is utilized to match every section with its corresponding summary, extrapolated from the full-length summary. Every sentence in the summary is matched with the best matching sentence across all sections, using methods in the sub-summary matching column (Table 1). The scoring method is an improved version used in [3]:

$$\max_i(w_l R^l l(k_i) + w_{r2} R^{recall-2} l(k_i) + w_{r1} R^{recall-1}_{num} l(k_i)) \tag{5}$$

where R^l stands for `ROUGE-L`, $w_{r2}R^{recall-2}$ is `ROUGE-2` recall metric and $R^{recall-1}_{num}$ is `ROUGE-1`, computed only on numerical part of the sentence, and k_i is the sentence i across all sections in the document. Weights are $w_l = 0.1$, $w_{r2} = 0.4$, and $w_{r1} = 0.5$. The base model loss was evaluated for every epoch (maximum of 10) on the evaluation set. The best model with lower loss was chosen for every run. We fine-tuned large models on `data_v4`, `data_v5`, and `data_v6`. Results for all experiments, including the number of training steps and loss of best model, are shown in Table 4. Additionally, we computed metrics for best model for very run. Scores for `Base-2` and `Pegasus-L-v6` are also reported in Table 4, which reports full fine-tuning parameters as well. Our optimizer was `Adafactor` with an initial warmup linear rate, which allows to save memory [10]. We used the `SentencePiece` tokenizer with a dictionary from the `Pegasus` model [9].

Table 4: Results from BigBird Summarization Experiments.

BigBird	Decoder Output	Dataset	Batch Size	Learning Rate	Loss	Pred. Loss	Log-likelihood	Steps	Training Time	Evaluation Time
Base-1	256	v2	512	5E-04	2.590	2.630	-3.000	10,500	15m	15m
Base-2	256	v3	512	5E-04	2.587	2.649	-2.975	3,600	15m	15m
Pegasus-L-v4	512	v3	8	5E-04	2.586	2.676	-3.010	3,600	2h	8h
Pegasus-L-v6	256	v4	8	5E-04	2.586	2.676	-3.010	324	1h	1h
Pegqasus-L-v6	512	v5	8	5E-04	2.680	2.680	-3.030	972	2h	5h

Batch size was decided based on `TPU` configuration. The base model was trained on `TPU v2.8-512` cores; the large model was trained on `TPU v3.8-8` cores. Loss is not directly comparable on different dataset versions. All runs improved considerably in the first epoch and showed further improvement in the final metric (up to 3-4 epochs.)

The first run on `data_v1` produced unsatisfactory results, as summaries were one-word long, or empty. We cleansed the dataset further and improved the matching algorithm until obtaining `data_v3` — as described in Data sub-section. We then fine-tuned the `Pegasus` large model with a decoder output maximum length of 512 on `data_v3`. It was unable to produce longer summaries. The output was filled with repeated words. We decided to use a minimum length m_l of 100 words in summaries included in the training set (`data_v4`) and increasing it to 200 words for `data_v5`. This improved the quality of the summaries generated, and also the final metrics. Using a higher percentage of long summaries in the training set improved the generation of longer summaries. We tested if the cleaning rules were impacting the score, so we kept the minimum length of 200 words and 3 cleaning rules (`data_v6`). This produced the best results (`BigBird` v3 in Table 5.)

The first run on `data_v1` produced unsatisfactory results, as summaries were one-word long, or empty. We cleansed the dataset further and improved the matching algorithm until obtaining `data_v3` — as described in Data sub-section. We then fine-tuned the `Pegasus` large model with a decoder output maximum length of 512 on `data_v3`. It was unable to produce longer summaries. The output was filled with repeated words. We decided to use a minimum length m_l of 100 words in summaries included in the training set (`data_v4`) and increasing it to 200 words for `data_v5`. This improved the quality of the summaries generated, and also the final metrics. Using a higher percentage of long summaries in the training set improved the generation of longer summaries.

4.5 GPT-3

We produced summaries for random samples of the test set from `beta version GPT-3` (`Davinci` model). We generated summaries from a maximum input length of 2k tokens; maximum output length 512, temperature 0.25, top likelihood (0.7-1), best of 5 results. Even if `GPT-3` has not been pre-trained in financial summarization, it produced summaries that we included in human evaluation. In some cases it added information about companies not present in the text which turned out to be correct, nonetheless.

4.6 Results

Table 5 shows the outputs of our models, based on `ROUGE` metrics. We compare `BERT`, `baseline(sumTO)`, `BigBird`, `GPT-3` on the summarization task. `BERT` and baseline are similar methods. Main advantange of `sumTO` is to be fine-tuned on the dataset and add a final abstraction summarization layer. `BERT` is not fine-tuned on financial data. The gap between the two indicates fine-tuning makes a difference. Nevertheless, `BERT` can produce fluent summaries and have the best `R-1` precision score. This can be explained by analyzing the lengths of the summaries produced by `BERT` which are the shortest. Since the precision is expressed in matched words as a percentage of length of summary, genrating shorter summaries can boost the metric. We conclude `BERT` is producing short summaries retaining information but it is missing some information in the text. `sumTO` is producing good overall results on `ROUGE` metrics. `BigBird` is producing good recall scores, but not so good precision scores. We think the reason is that `BigBird`, keeping more context, producing longer summaries than `BERT`, it is penalized in the precision metric. Additionally, we have to keep in mind that depending on the threshold, the precision-recall curve could change. Given our time constraints, we could not change the threshold and run more experiments. The average summary length is: 2,010 words for `BERT`, 2,791 words for `BigBird`, and 6,381 words for the `Baseline`.

Table 5: ROUGE Metrics for Our Models

Model/ Metric	Rouge-1			Rouge-2			Rouge-3			Rouge-4		
	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
Baseline	53.031	36.782	41.636	17.167	11.939	13.490	7.043	4.855	5.518	4.078	2.792	3.190
BERT	21.237	48.731	28.052	5.466	13.106	7.290	2.207	5.424	2.953	1.371	3.435	1.840
BigBird	24.993	42.779	28.802	6.667	13.011	7.888	2.735	6.139	3.336	1.649	4.032	2.050
BigBird(v2)	49.787	29.537	35.141	14.419	8.550	10.149	5.542	3.302	3.909	3.149	1.892	2.227
BigBird(v3)	54.991	22.575	30.672	15.788	6.426	8.740	5.875	2.398	3.256	3.268	1.342	1.817

4.7 Human Evaluation

Table 6 shows the quantitative results of human evaluation. These numbers are for `R-1`, using the gold standard. We see that once a model is trained, the time for summarization is greatly reduced (the time to summarize for humans includes the average time to summarize the 12 samples only.) We also see that the recall and the precision for `BERT` and `BigBird` are similar (in line with the results shown in Table 6), but much lower than that of our manual summaries. As a consequence, the `F1` is also much higher for human summarization. We hypothesize that given the nature of the source text, it is possible the models had difficulties interpreting the text (and relating information.) Given the high scores for human summarization, we realize we might have tried a bit too hard to get good summaries in our manual approach. It is also possible that the gold standard tried hard to come up with a good summary as well, and so that, our metrics for human evaluation are high because our human-produced summaries are similar to the gold-standard. This is the opposite case of someone doing a gold standard by just copy-pasting, which will see high metrics in the models as the model is closer to the source. We also have to consider that this task is highly abstractive, and that variance is rampant in human evaluation (but our sample is too small to show it in here.)

In the qualitative side, we see that `BigBird` tries to mantain the structure of the text provided (e.g. using vignettes if they are present in the original text, getting rid of titles/random entries that look out of place), while `BERT`'s approach tried to cram everything together (while still maintaining important information). In general, manual summaries are more alike to `BigBird`-produced ones, albeit not completely so. We think this is a consequence of us producing our summaries "from scratch." This is to say, we did not copy-pasted information, or trimmed the original text. Another important observation is that `BigBird` summaries are cleaner as they keep the information neat and better

Table 6: Quantitative Results for Human Evaluation.

Summarization Approach	Time to Summarize	Precision	Recall	F1
Human	8h	81	97	87
BERT	1m	34.47	15.31	19.37
BigBird	20s	37.15	14.26	21.43

organized (i.e. better readability, as human summaries) while BERT summaries seem to have trouble separating information, seeming to be more in line with an extractive-only approach.

4.7.1 Fluency, Length and Relevance, Interpretability and Overall Meaning

BERT was at times difficult to follow, given it "crammed" text together, and often changed topics abruptly. BigBird was more similar to the manual summary, and had only minor problems in sentence transition. Human-produced summaries were the most fluent. For shorter texts, the length of BERT's summaries was longer than that of BigBird's. However, much of the text included was left-over information (e.g. text that was not needed or relevant.)For longer texts, the length of BERT's summaries was shorter than that of BigBird's, and the information remained relevant. In general, manual summaries were on the longer side.) BigBird was easier to interpret, as it keep the information neatly organized, and the necessary context was provided. BERT also proved useful, but sometimes out-of-context sentences/words made it difficult to understand what the message was. Unlike BERT, BigBird did not include seemingly random information at times, and kept the information relevant. In general, BigBird summaries were better than we expected (and better than BERT's). We think this might be because the architecture allows to analyze longer sentences at the same time, and so key connections are seldomly lost. As the capacity to use longer windows for analysis increases, the summarization task is likely to deliver better results.We were able to generate a few full samples from GPT-3. They were fluent, but included information not in the text. We could not investigate further given the time constraint. sumTo generated summaries similar to BERT, but more fragmented. The extractive nature was noticeable. In a few cases, the interpretation of few sentences was ambiguous. For example:

> *There remains a strong demand for exposure to property and we have been working to identify additional property products to meet future demand for the UK and continental Europe. It remains one of our fastest growing separate accounts and as at 31 December 2007 had a value of £752m.*

where the first and second sentences are related to two different sections.

5 Analysis

5.1 Comparison between Our Model and the Original Model

We compare BERT, baseline(sumTO), BigBird, GPT-3 on the summarization task. BERT and baseline are similar methods. Main advantange of sumTO is to be fine-tuned on the dataset. BERT is not fine-tuned on financial data. The gap between the two indicates fine-tuning makes a difference. Nevertheless, BERT can produce fluent summaries and have the best R-1 precision score. This can be explained by shortest lengths of the summaries produced by BERT. We conclude BERT is producing short summaries retaining part of the text but it is missing some content. sumTO is producing good results on ROUGE metrics. From human evaluation we see BERT produced good enough summaries, but also keeps some irrelevant information. GPT-3 produced few summaries whereas others are mixed with information not contained in the text. According to human evaluation, this happens because when there is a lack of text or obvious connections, the model just pulls information it has learned and tries to blend it in. Despite, these summaries contain much of the information included in the human-produced reports, using other words — that are more similar to the original text than words contained in human summaries. BigBird scored first in R-1 recall results. This shows it can match maximum amount of single words compared to the target summary among the three. Lower numbers on precision are justified by the abstractive nature of the approach which needs more words to generate a fluent summary and retains more context. This feature helps in semantic retention.

Furthermore BigBird is producing summaries longer than BERT. We have to take into account we truncate very long sections losing some information contained towards end.

5.2 Results from Different Datasets

The dataset is a real-world, challenging scenario in which text is extracted from pdfs, and is not fully clean from headers. Furthermore, tables are translated to plain text. Implementing cleansing rules helped in training. Improving the rules and the section-summary matching algorithm, We noticed a performance booster.

5.3 Summary Length

Length of gold standard summaries and section can vary a lot. One of the challenges is to train BigBird to produce long and short summaries as required. This is in line with similar challenges in literature where they provide a specific checkpoint for long summaries[9]. Proving more long summaries in the training set improved the length of the summary.

6 Conclusion

We compared BERT, baseline (sumTO), BigBird, and GPT-3 on the summarization of financial narrative task. We considered a real-word scenario in which semantic retention and ROUGE metrics are equally important. We scored the models based on ROUGE metrics and on human evaluation. The financial narrative summarisation task is highly extractive in nature, and abstractive features are challenging. Moreover semantic retention is highly relevant. Our proposed context-aware approach BigBirdFLY scored first in R-1 recall numbers. This indicates that it can match the maximum number of 1-gram from target summary. Importantly, it produced more fluent summaries. To keep summaries coherent and fluid it utilized more words compared to BERT, sumTO, and it is penalized in overall F1 metric. Nevertheless in a few samples numerical facts were misplaced, possibly due to a lack of pre-training on financial data. Human evaluation proved valuable to understand differences between models. In particular, BigBird proved more aligned with our manual summaries than BERT, and kept the information better organized (i.e. better readability), including mostly relevant information. Results from sumTO are similar to BERT, but more fragmented, and at times the interpretation was ambiguous. GPT-3 generated fluent summaries which includes information not in the text which nonetheless happened to be correct.

We showed how a sentence-by-sentence extraction method can benefit from a wide context window. Comparison with recursive methods[3] is left as future work. The financial summarization task revealed to be a challenging task without a unique method being better overall. We believe this could lead to introduction of new tasks and new models. For future work, and given the very specific feature of the task, we think adding further domain/task specific pre-training on financial text can help BigBird score better. A better section-to-summary matching algorithm (topic-based) is needed as well. Finally, the abstractive feature of BigBird can be modified to be more extractive, utilizing weights from attention, thus taking the best from both worlds.

References

[1] Tas, O.,& Kiyani, F. (2007). A Survey Automatic Text Summarization. *PressAcademia Procedia*, 5(1), 205-213.

[2] Zhao, N., Jiang, Y., & Liu, Y. (2019.) Sentence-Level Extractive Text Summarization *Final Report of Stanford CS224N.*

[3] A. Singh (2020, December). *PoinT-5: Pointer Network and T-5 based Financial Narrative Summarisation* (pp. 105-111).

[4] Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., ... & Ahmed, A. (2020). Big Bird: Transformers for Longer Sequences. *arXiv preprint arXiv:2007.14062.*

[5] Nenkova, A., & Passonneau, R. J. (2004). Evaluating Content Selection in Summarization: The Pyramid Method. *In Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: Hlt-naacl 2004* (pp. 145-152).

[6] M. L Quatra, L, Cagliero (2020, December). *End-to-end Training For Financial Report Summarization*.

[7] Lin, C. Y. (2004, July). Rouge: A Package for Automatic Evaluation of Summaries. *In Text Summarization Branches Out* (pp. 74-81).

[8] Zheng, S., Lu, A., & Cardie, C. (2020, December). SUMSUM@ FNS-2020 Shared Task. *In Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation* (pp. 148-152).

[9] J. Zhang, Y. Zhao, M. Saleh, P. J. Liu *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization*.

[10] N. Shazeer, M. Stern, *Adafactor: Adaptive Learning Rates with Sublinear Memory Cost*.

[11] python-ftfy: *https://github.com/LuminosoInsight/python-ftfy*

[12] clean-text: *https://github.com/jfilter/clean-text*

[13] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*

[14] He, Y., Chen, J., & Kim, P. (2020.) Generative Pre-Training Language Models with Auxiliary Conditional Summaries *Final Report of Stanford CS224N.*

[15] Liu, Y., & Lapata, M. (2019). Text Summarization with Pretrained Encoders. *arXiv preprint arXiv:1908.08345.*

[16] Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E. D., Gutierrez, J. B., & Kochut, K. (2017). Text Summarization Techniques: A Brief Survey. *arXiv preprint arXiv:1707.02268.*

[17]Araci, D. (2019.) Financial Sentiment Analysis with pre-Trained Language Models. *arXiv preprint arXiv:1908.10063.*

[18] Watts, D. J., & Strogatz, S. H. (1998). Collective Dynamics of 'Small-World' Networks. *nature*, 393(6684), 440-442.

[19] CNN-Daily Mail Dataset *https://github.com/abisee/cnn-dailymail*

[20] BBC Xsum Dataset *https://github.com/EdinburghNLP/XSum/tree/master/XSum-Dataset*

[21] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). Language Models Are Few-Shot Learners. *arXiv preprint arXiv:2005.14165.*

[22] Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509.*

[23] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.*

[24] Tran, C. (2020). Extractive Summarization with BERT. (*Accessed 29/02/2021.*) https://chriskhanhtran.github.io/posts/extractive-summarization-with-bert/

[25] Lloret, E., Plaza, L., & Aker, A. (2018). The Challenging Task of Summary Evaluation: An Overview. *Language Resources and Evaluation*, 52(1), 101-148.

[26] Liu, F., & Liu, Y. (2008, June). Correlation between Rouge and Human eEaluation of Extractive Meeting Summaries. In *Proceedings of ACL-08: HLT, short papers* (pp. 201-204).

[27] TPU Estimator *https://www.tensorflow.org/api_docs/python/tf/compat/v1/estimator/tpu/TPUEstimator*

[28] Google Cloud TPU. *https://cloud.google.com/tpu*

[29] BigBird repository *https://github.com/google-research/bigbird*

[30] FTRC program *https://www.tensorflow.org/tfrc*

[31] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. *arXiv preprint arXiv:1706.03762.*

A ROUGE Metrics

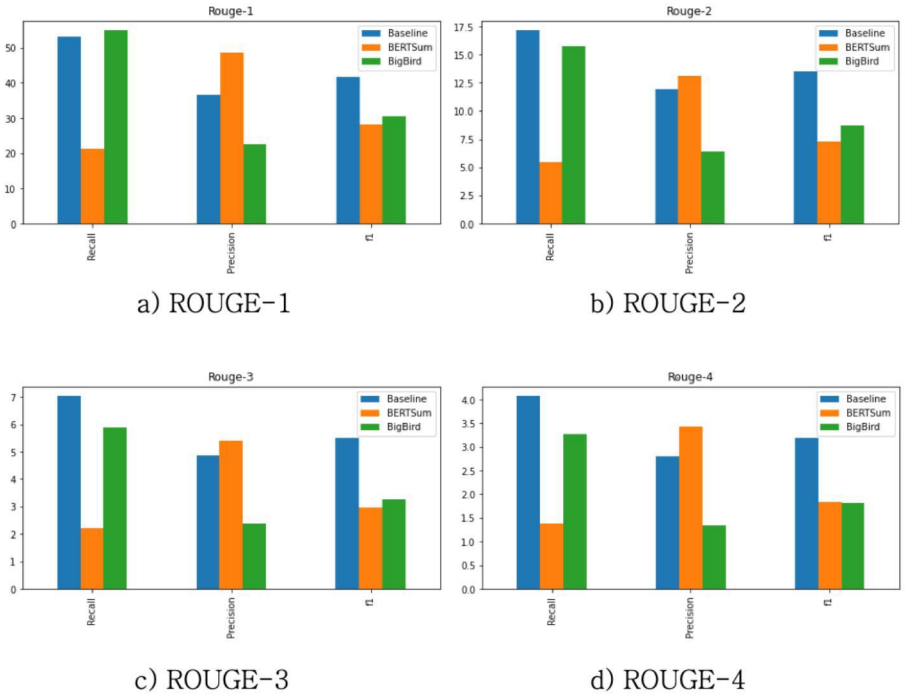

Figure 4: ROUGE Metrics for our Models. Summary Evalution of the Baseline, BERT+Summary Layer, and BigBird. Rouge-1 matrix (upper left), Rouge-2 matrix (upper right), Rouge-3 matrix (lower left), and Rouge-4 matrix (lower right). Scores were generated by comparing generated summary from each of these models against the available gold summaries for each document. We have used the sacrerouge module to generate the matrices.