FETILDA: Evaluation Framework for Effective Representations of Long Financial Documents

BOLUN (NAMIR) XIA, Rensselaer Polytechnic Institute School of Science, Troy, United States
VIPULA RAWTE, Rensselaer Polytechnic Institute School of Science, Troy, United States
APARNA GUPTA, Rensselaer Polytechnic Institute, Troy, United States
MOHAMMED J. ZAKI, Rensselaer Polytechnic Institute, Troy, United States

In the financial sphere, there is a wealth of accumulated unstructured financial data, such as the textual disclosure documents that companies submit on a regular basis to regulatory agencies, such as the Securities and Exchange Commission. These documents are typically very long and tend to contain valuable soft information about a company's performance that is not present in quantitative predictors. It is therefore of great interest to learn predictive models from these long textual documents, especially for forecasting numerical key performance indicators. In recent years, there has been great progress in natural language processing via pre-trained language models (LMs) learned from large corpora of textual data. This prompts the important question of whether they can be used effectively to produce representations for long documents, as well as how we can evaluate the effectiveness of representations produced by various LMs. Our work focuses on answering this critical question, namely, the evaluation of the efficacy of various LMs in extracting useful soft information from long textual documents for prediction tasks. In this article, we propose and implement a deep learning evaluation framework that utilizes a sequential chunking approach combined with an attention mechanism. We perform an extensive set of experiments on a collection of 10-K reports submitted annually by U.S. banks, and another dataset of reports submitted by U.S. companies, to investigate thoroughly the performance of different types of language models. Overall, our framework using LMs outperforms strong baseline methods for textual modeling as well as for numerical regression. Our work provides better insights into how utilizing pre-trained domain-specific and fine-tuned long-input LMs for representing long documents can improve the quality of representation of textual data and, therefore, help in improving predictive analyses.

CCS Concepts: • **Computing methodologies** → **Machine learning**; **Information extraction**; **Natural language processing;**

Additional Key Words and Phrases: Text regression, language models, long text documents, financial documents, 10-K reports

B. (Namir) Xia and V. D. Rawte contributed equally to the article.

This work was supported in part by an industry funded award from the RPI-Stevens NSF IUCRC Center for Research toward Advancing Financial Technologies (CRAFT; NSF Award No. 2113850).

Authors' Contact Information: Bolun (Namir) Xia, Computer Science, Rensselaer Polytechnic Institute School of Science, Troy, New York, United States, e-mail: xiabolun@gmail.com; Vipula Rawte, Computer Science, Rensselaer Polytechnic Institute School of Science, Troy, New York, United States, e-mail: rawtevipula25@gmail.com; Aparna Gupta, Lally School of Management, Rensselaer Polytechnic Institute, Troy, New York, United States; e-mail: guptaa@rpi.edu; Mohammed J. Zaki, Department of Computer Science, Rensselaer Polytechnic Institute, Troy, New York, United States; e-mail: zaki@cs.rpi.edu.

ACM 1556-4681/2024/06-ART182
https://doi.org/10.1145/3657299

ACM Reference Format:
Bolun (Namir) Xia, Vipula Rawte, Aparna Gupta, and Mohammed J. Zaki. 2024. FETILDA: Evaluation Framework for Effective Representations of Long Financial Documents. *ACM Trans. Knowl. Discov. Data.* 18, 7, Article 182 (June 2024), 27 pages. https://doi.org/10.1145/3657299

1 INTRODUCTION

Unstructured data such as text is growing very fast in different domains. Especially, textual data from financial documents has been found to be beneficial in making predictions [8]. Utilizing such large volumes of textual data requires **natural language processing (NLP)** and **machine learning (ML)** techniques. These techniques summarize the text as (a set of) numeric feature vectors, which are called representations or embeddings, and which can in turn serve as inputs to machine learning models to predict some target variables.

The traditional approach for text-based learning is via the **Term Frequency-Inverse Document Frequency (TF-IDF)** method [17], which can represent the document as a long numeric vector of TF-IDF scores for each word. However, TF-IDF does not attempt to directly extract the latent semantic information within the text. The current progress in text representations was initiated by word embedding methods, such as word2vec [25] and GloVe [28], which capture both the lexical and semantic information of a document to some extent. The main idea is to learn word representations based on the context of each word. However, these methods learn only a single, *static* representation for each word, and do not take into consideration the phenomenon of polysemy, where a word can change its meaning depending on the context (for example, the the word "bank" in the financial context has a very different meaning compared to the "bank" of a river). The **state-of-the-art (SOTA)** pre-trained language models, such as GPT [29] and BERT [10] are built on top of the very effective Transformer-based attention model [41], which can learn *contextual* word embeddings. These embeddings are *dynamic* in terms of the surrounding block or context of the word, so that the same word can get different representations that are most effective in capturing the lexical and semantic information. These models have shown SOTA performance on a variety of downstream tasks such as question answering, text classification, and regression. However, extracting "good" representations for such long documents remains a challenging task: the length of the 10-K documents poses both a methodological and ontological burden. Methodologically, financial reports are significantly longer, compared to the maximum length of a textual sequence that BERT [10]-based models can handle. For instance, the **Management Discussion and Analysis (MD&A)** section of the 10-K reports that companies publish annually is usually around 12,000 word-tokens. BERT-based models have a restriction on the maximum number of tokens, around 512, with some newer models, such as Longformer [4] and BigBird [48], reaching up to 4,096 tokens. Hence, to use these **language models (LMs)** on long documents directly would require significant truncations of the texts, leading to information loss. Ontologically, the challenge is the classic machine learning task of extracting or learning informative features that can represent the input well. This question becomes quite complex in the context of representing a long document. Contextual word embeddings are well suited for this given their ability to "understand" different meanings for a word in different contexts. However, it remains an open question as to how to combine the various contextual word embeddings into an effective document level embedding.

An additional challenge is that the SOTA language models are pre-trained on massive and generic corpuses, e.g., from web crawls, Wikimedia, and so on. However, to be effective for the financial context, it is important for LMs to learn domain-adapted and task-specific representations of long documents to meaningfully support predictive analyses. This can usually

be achieved either by pre-training (from scratch) a domain-specific language model on a huge financial corpus to adapt to its particular domain, or by fine-tuning a pre-trained LM on a specific financial dataset for the downstream tasks, or by combining the two approaches of adapting the LM to a particular financial domain followed by fine-tuning on downstream tasks. Recently there have been several attempts at pre-training BERT on large financial corpuses to adapt it for tasks in the financial domain. Liu et al. [23], Huang et al. [16], Araci [3], and DeSola et al. [9], each pre-trained the BERT model from scratch on financial corpora, such as financial news, corporate reports, financial websites, and so on. Incidentally, all four approaches are called FinBERT!

To address the long document representation challenge within the context of financial disclosure documents, we propose our evaluation framework called **Fine-tuned Embeddings of Texts in Long Document Analysis (FETILDA)**. Our approach is particularly designed for downstream predictive or regression tasks, where the input document representations are combined with other numeric attributes (if available) to predict a target response variable of interest, such as key performance indicators (e.g., return on assets, earnings per share), stock volatility, and so on. FETILDA comprises a chunk-based deep learning framework, where a long document is split into several smaller chunks and then each chunk is processed using an specific language model (e.g., BERT [10], FinBERT [16], Longformer [4], Nystromformer [44], etc.). The layers of the LM can remain frozen, or they may be unfrozen for fine-tuning, or only the last layer can be frozen. The chunk level representations are then pooled together using a **Bidirectional Long Short-term Memory (Bi-LSTM)** model equipped with self-attention mechanism. The pooled chunks are then aggregated into a document level representation, which serves as input to fully connected layers for target variable prediction. This way, long documents can be represented by various LMs without significant information loss due to truncation.

We experiment our evaluation framework using different corpora: (i) **FIN10K (All Public Companies):** 10-K reports for all US companies from 1996 to 2013 [22], and (ii) **U.S. Banks:** 10-K reports submitted annually to the **Securities and Exchange Commission (SEC)** by U.S. banks for the period from 2006 to 2016. We have conducted extensive experiments using these datasets and applied our evaluation framework to different predictive analysis regression tasks: (i) analysis of a company's stock market volatility on the FIN10K dataset, and (ii) predicting **key performance indicators (KPIs)** of a bank's financial performance on the U.S. Banks dataset: the indicators include **Return on Assets (ROA)**, **Earnings Per Share (EPS)**, **Return on Equity (ROE)**, **Tobin's Q Ratio (TQR)**, **Leverage Ratio (LR)**, **Tier 1 Capital Ratio (T1CR)**, **Z-Score (Z)**, and **Market to Book Ratio (MBR)**. Our results compared against the different baseline methods show that our approach is significantly better and yields SOTA results for long financial text regression tasks. In summary, our main contributions are:

- We propose an evaluation framework of language models for long document regression tasks in the financial domain. Our FETILDA approach is designed to learn effective document level representations via a sequential chunking approach combined with an attention mechanism. As such, our approach combines the best of both the attention-based Transformer model and Bi-LSTM recurrent networks.
- We conduct an extensive set of experiments to quantitatively examine the efficacy of language models in long financial document representation. We applied the framework on two different 10-K datasets, and on 9 different regression tasks (in terms of the target variable). We show that through our evaluation framework, pre-trained domain specific LMs outperforms several different baseline methods, and achieves SOTA results on long financial documents.

2 RELATED WORK

Machine learning plays an important role in financial analytics. One of the important areas of finance is investment stock return forecasting, as well as fundamentals forecasting and risk modeling, that mainly employ quantitative or numeric data [12]. Different ML models such as **Support Vector Machines (SVMs)**, single hidden layer Feed-forward Neural Networks and **Multi-layer Perceptrons (MLPs)** were used for the prediction of future price movements in Reference [27]. They mainly used two sets of features for their ML classifiers: (1) handcrafted features formed on the raw order book data and (2) features extracted using ML algorithms. Some other models such as Random Forest [19], XGBoost [42], Bi-LSTM, and stacked LSTMs [33] were also implemented to predict business risk and stock volatility. The main limitation of these works is that they ignore valuable textual data that can provide more insight into the intangible features such as sentiment, knowledge capital, risk culture, and so on.

2.1 Textual Data: Sentiment Analysis

An approach to predict financial quantitative variables is using financial textual sources such as news reports, analyst assessments, earnings call transcripts, and company filing reports. And much work has been done in analyzing the value of the soft information that financial reports contain over the years, such as in Reference [11]. In Reference [35], textual features were created by using the negative words in the Harvard-IV-4 TagNeg dictionary and constructing a document-term matrix from the news stories. These features were used to predict firms' earnings and stock returns. A novel tree-structured LSTM was proposed to automatically measure the usefulness of financial news using both text and cumulative abnormal returns [6]. A dual-layer attention-based neural network model was developed to predict stock price movement using the text in financial news [47]. Estimating the value of text in financial news is important, because it drives the investment decision making process.

Financial sentiment analysis is challenging because of lack of labeled data specific to this domain. Moreover, the general-purpose pre-trained language models fail to capture the financial context. Reference [3] proposed the FinBERT model, which can be fine-tuned on the **financial sentiment analysis dataset (FiQA)** to outperform the general BERT model. Besides financial news, in Reference [21], the authors constructed textual features from 10-K reports. They used these features to predict the future stock volatility indicating the effectiveness of text. A deep learning model trained on the SEC filings was used to improve the prediction of company's stock price over the traditional ML models [32].

The authors of References [38, 39] extracted additional textual features by expanding the L&M sentiment word list [24] semantically and syntactically, using word2vec [25]. Similarly, the uncertainty word list in L&M dictionary was expanded using word2vec to predict stock volatility [36]. The authors in Reference [37] expanded the L&M dictionary by training industry-specific word embedding models using word2vec to predict volatility, analyst forecast error and analyst dispersion. Reference [34] showed how automatic domain adaption of the L&M sentiment list using word2vec [25] improved the prediction of excess return and volatility. The aforementioned dictionary expansion approaches used word2vec model to select the top k closest words to the words existing in the L&M dictionary. Since word2vec is a model based on static word embeddings, it fails to capture the dynamic context of the words.

2.2 Language Models in Finance

In terms of domain-adapted pre-trained LMs, in the English-speaking Finance sphere, four models have been proposed and implemented, all named FinBERT: Liu et al. [23], Huang et al. [16], Araci [3], and DeSola et al. [9], all of which are pre-trained to adapt to different financial domains.

Originally, in the general domain, BERT [10] was pre-trained on two corpora: BooksCorpus (0.8 billion words), and English Wikipedia (2.5 billion words), forming a total of 3.3 billion words, so the idea of these financial language models is to take the original model, and pre-train it on their respective financial corpora.

Araci [3] was the first to propose FinBERT as a pre-trained domain-adapted BERT [10] on a corpus called TRC2-financial, which includes 46,143 documents with more than 29M words and nearly 400K sentences, from a set of Reuters news stories. In experimentation, they saw a 15% increase in accuracy for classification tasks, a significant margin. Liu et al. [23] focused on financial news and dialogues present on websites, and collected three financial corpora: 13 million financial news (15 GB) and financial articles (9 GB) from Financial Web, totaling 24 GB and 6.38 billion words; financial articles from Yahoo! Finance, totaling 19 GB and 4.71 billion words; and question-answer pairs about financial issues from Reddit, totaling 5 GB and 1.62 billion words. They pre-trained their model on these corpora to adapt it to the financial news and dialogues domain. In experimentation, they saw their model outperform BERT [10] on all the financial tasks in their experiments, in terms of accuracy, precision, and recall [23]. Huang et al. [16] focused on financial and business communications that companies produce, and collected a corpus of three types of data: 10-K and 10-Q reports, totaling 2.5 billion word tokens; earnings call transcripts, totaling 1.3 billion word tokens; and analyst reports, totaling 1.1 billion word tokens [16]. They report that their model outperforms BERT [10] in three sentiment analysis tasks, all by significant margins [16]. DeSola et al. [9] introduced another domain-specific pre-trained language model, also named FinBERT, for financial NLP applications. This model was trained on the 10-K filings from 1998 to 1999 and from 2017 to 2019, totaling 497 million words, and it showed better performance than BERT on the masked LM and next sequence prediction tasks.

2.3 Long Document Language Models

Apart from LMs adapted to specialized domains, there has been a slew of papers on state-of-the-art pre-trained LMs in the general domain, such as GPT-1 [29], GPT-2 [30], GPT-3 [5], T-5 [31], ELECTRA [7], and so on. These are massive models trained on enormous corpora, but the challenge of representing long documents persists, in that these models still cannot handle long textual sequences, due to the quadratic computational complexity that they usually entail.

To tackle this challenge head-on, several recent works, such as Longformer [4], **Extended Transformer Construction (ETC)** [1], and BigBird [48], have been proposed, all of which innovate on the self-attention mechanism to reduce the computational complexity from quadratic to linear, which then enables it to process longer sequences of text. In addition, more recent works on transformer models with linear attention, such as Reformer [20] and Nystromformer [44], innovate on how to mathematically approximate the self-attention matrix calculations with less time and space complexity, instead of changing the self-attention mechanism.

Longformer [4] replaces the full self-attention matrix, which scales quadratically with the length of the input sequence, with three types of sparse attention schemes: sliding window attention, which selects only the entries on the descending diagonal line of the self-attention matrix, with the "thickness" of the line being a certain size; dilated window attention, which adds gaps of a certain size in between the sliding window, making the descending diagonal line dilated; global attention, which has certain specific tokens attend to all the tokens across the sequence, both horizontally and vertically, thereby enabling global contextual representation of the sequence. Longformer was shown to outperform baseline methods consistently, and particularly, its results were more apparent where the experiment required long contextual information.

ETC [1] is very similar to Longformer, with nuanced variations. ETC replaces the full self-attention matrix with global-local attention, which splits the self-attention matrix into four parts:

global-to-global, which is a small square on the top left of the matrix, where certain special global tokens attend to each other; global-to-long, which is a horizontal rectangle on the top right of the matrix, where global tokens attend to regular tokens; long-to-global, which is a vertical rectangle on the bottom left of the matrix, where regular tokens attend to global tokens; long-to-long, which is a compressed version of the descending diagonal line in the large square on the bottom right of the matrix, essentially a sliding window attention compressed into a rectangular matrix, where regular tokens attend to other regular tokens in its window. In experimentation, ETC yielded state-of-the-art results, especially in question answering scenarios.

BigBird [48] extends further on ETC [1], adding random sparse attention into the mix, building on top of the global-local attention mechanism of ETC. Random entries in the self-attention matrix are selected to generalize over the full matrix. From a graph theory perspective, this means a shorter average path between any two nodes, making it a better approximation of the full graph. And from a NLP perspective, in most texts, there tends to be locality of reference, where a word relates closely with words around it, so BigBird tries to account for this with their particular random sparse attention scheme.

2.4 Chunking-based Representation Schemes

Several papers proposed chunking-based approaches to enable representations of documents longer than the maximum length of word-tokens that LMs can take, for various different downstream tasks. Yang et al. [46] proposed a Siamese hierarchical matching model, using sentence blocks to construct representations for twin documents at the same time, for the purpose of document matching. In their approach, the sentence block representations are passed through a transformer model and the first token of the resulting output is used as the document representation. With this framework, they are able to handle maximum document lengths up to 2,048 word tokens, being able to take a maximum of 64 sentences, with the maximum length of each sentence being 32. Since their downstream task was document matching, they chose the Siamese design for their model, which takes in two document inputs simultaneously to compare their similarity, while our evaluation framework is oriented towards document regression, which utilizes a document to predict quantitative metrics. In terms of the underlying architecture, they employ a sentence-level block merging mechanism that can take in "long form" documents not exceeding 2,048 word tokens, while our framework utilizes a paragraph-level chunking and merging mechanism that can actually analyze real-world long documents in the financial domain. For example, even just one section of the 10-K reports that we learn on, namely, Item 7, averages around 12,000 words, and can be more than 24,000 tokens after tokenization. Therefore, instead of sentence-level blocks, we use (paragraph-level) chunks, and we weigh these chunks using their respective attention scores obtained by a self-attention mechanism. Finally, we pool the weighted chunks together into a document representation, using a bi-LSTM network. In our experiments, we utilize different models that can enable us to have the maximum chunk length range from 512 up to 8,192 word tokens, which then enables us to generate representations for documents that average 24,000 word tokens.

Gong et al. [14] proposed a recurrent chunking mechanism for the purpose of machine reading comprehension, where the machine is given a long document and a question, and is required to extract a piece of text from the document as the answer to the questions. Towards that end, they needed the chunking mechanism to be such that the separation point of various chunks would not cut the correct answer in half, nor prevent surrounding contexts from being retained. Therefore, their main innovation is in enabling a more flexible chunking policy, and in a recurrent chunking mechanism that can provide context surrounding a chunk segment. In experimentation, they use BERT, which enables maximum sequence lengths ranging from 192 to 512 word tokens.

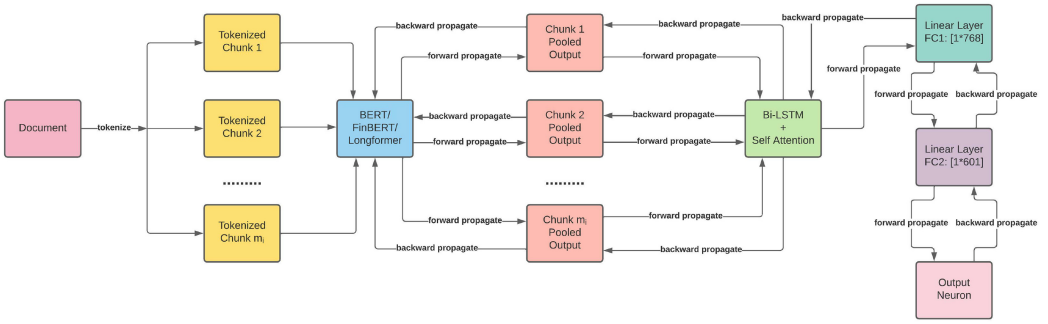

Fig. 1. FETILDA: Overall framework.

Our framework takes a different approach to the chunking policy, since our goal is the extraction of predictive financial text features from the document. Moreover, we also try to increase the maximum sequence length of a chunk up to 8,192 word tokens, which then allows us to take in more information in one chunk in an organic way.

In more recent work, Grail et al. [15] use a bi-GRU network to pool the chunks together, instead of a bi-LSTM network. But the purpose of their framework is long document summarization, instead of extracting predictive text features. In their approach, they use BERT as the LM, and therefore, can only process up to 512 word tokens for a chunk. Further, they consider their approach to be an alternative to long sequence LMs such as Longformer. Instead of considering long sequence LMs as alternatives, we incorporated them into our FETILDA framework. Therefore, in our approach, we experiment with various underlying LMs for FETILDA, be it BERT-based models, or long sequence LMs, or linear attention transformer LMs, enabling us to have different maximum sequence lengths ranging from 512 to 8,192 tokens.

3 FETILDA: LONG DOCUMENT REPRESENTATION

Figure 1 shows the FETILDA framework. FETILDA first splits a long document into smaller fragments or chunks, then processes each chunk using a language model, all of whose layers are fully unfrozen for fine-tuning, then pools the chunks together using a Bi-LSTM layer endowed with a self-attention mechanism into an aggregate vector representation of the entire document. The chunk representations are extracted from the underlying language model (BERT [10], FinBERT [16], Longformer [4], or Nystromformer [44]) using several different pooling strategies including using the default pooler output and combining the features from the last few layers. These chunk sequences are passed onto a Bi-LSTM model whose hidden context states and outputs are used to learn chunk-level attention scores to extract the final document embedding. Finally, the document embedding is passed through the linear layers to obtain the final target prediction. In addition, we perform task-specific fine-tuning on our entire model, including BERT, FinBERT, or Longformer, whose layers are fully unfrozen (or can be kept frozen if only pre-trained inputs are to be used), using **mean-squared error (MSE)** as the loss function. Overall, as shown in Figure 1, our methodology consists of four stages: (1) Chunk Generation, (2) Chunk-Level LM Pooling, (3) Document-Level Attention Pooling, and (4) Model Training and Fine-Tuning. We shall describe each of these next.

3.1 Chunk Generation

Let $L = \{d_1, d_2, \ldots, d_N\}$ denote a text corpus containing N long documents, where d_i denotes the ith document in the corpus. We tokenize each document d_i into a sequence of tokens

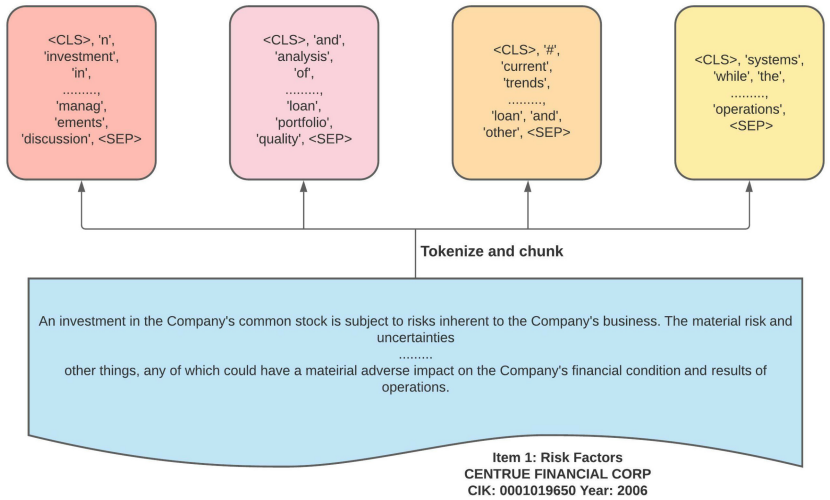

Fig. 2. Chunk generation.

$\{t_1, t_2, \ldots, t_{n_i}\}$, where n_i is the number of tokens for document d_i. The document token sequence is divided into chunks of length b, where b is the block or chunk size. Thus, each document d_i can be represented as a sequence of chunks $\{c_1, c_2, \ldots, c_{m_i}\}$, with m_i chunks of length b. We also prepend and append <CLS> and <SEP> tokens to each chunk, respectively, resulting in chunks of size $b + 2$. The chunk size dictates a maximum of $b + 2$ tokens for each chunk $c_i = \{t_0, t_1, \ldots, t_b, t_{b+1}\}$, with t_0 = <CLS> and t_{b+1} = <SEP>. We experiment with $b + 2 = 512$, $b + 2 = 4096$ and $b + 2 = 8192$, depending on the underlying language model used. For document where the last chunk has $k < b$ tokens, we pad the last chunk by appending the padding token (<PAD>) $(b - k)$ times to keep the chunk length intact. For each chunk, we also create an attention mask with [0] for padding tokens and [1] for non-padding tokens, which helps in attending only to the valid tokens and not the <PAD> tokens. Figure 2 shows an excerpt from Item 1 from a company's 10-K report, and the tokenization and chunking process with four resulting chunks.

3.2 Chunk-level Language Model Pooling

Given the sequence of chunks for a document, $\{c_1, c_2, \ldots, c_{m_i}\}$, we need to convert these into features vectors $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{m_i}\}$, that represent the token sequence in each respective chunk as a whole. We use SOTA language models like BERT [10], Longformer [4], and FinBERT [16] to generate contextual token and chunk embeddings. We thus input each chunk into the underlying language model, which typically outputs 12 hidden state layers $\{l_1, l_2, \ldots, l_{12}\}$, where l_i denotes layer i. The output of each of these layers contains $b + 2$ hidden state vectors $\{\mathbf{z}_0^l, \mathbf{z}_1^l, \ldots, \mathbf{z}_{b+1}^l\}$, for $b+2$ tokens in the chunk, each of which has a size of 768, which is the dimensionality of the hidden states. The language model also yields a default pooler output, which is the embedding vector for the <CLS> token, the first token, of the last hidden state layer after processing and activation, denoted by $\mathbf{z}_0^{12}$. Figure 3 shows the schematic of how we use the underlying language model to generate the hidden state layers, as well as the default pooler output, which are then combined using various strategies outlined below to yield the chunk embedding vector $\mathbf{c}_i$ for each chunk c_i within each document.

Creating contextual embeddings is challenging, since a word can have different meanings in different contexts. So it is important to first create contextual token embeddings and then experiment with different strategies to generate different chunk representations from these contextual

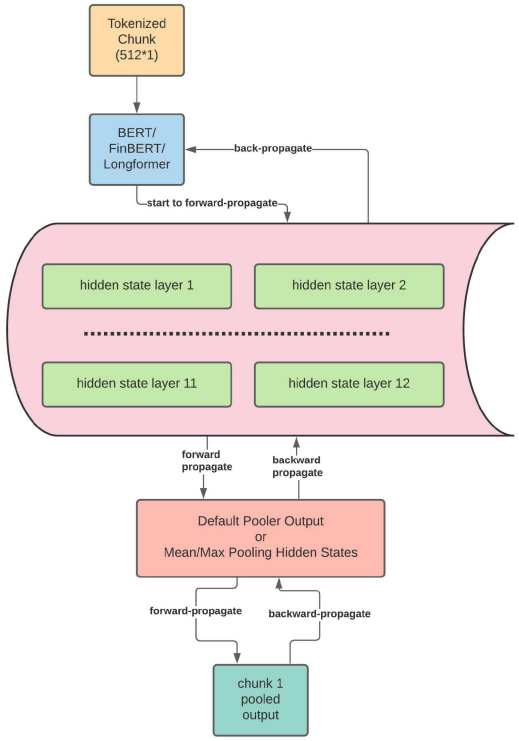

Fig. 3. Chunk level language model pooling for chunk embeddings.

embeddings. We therefore studied several approaches for creating the final chunk embedding vectors $\mathbf{c}_i$:

— *Default pooler output:* Since the <CLS> token embedding is an attention-weighted aggregation of all the tokens in a given chunk, each chunk c_i can therefore be represented by the default pooling output vector $\mathbf{z}_0^{12}$ as the chunk embedding vector $\mathbf{c}_i$. The size of $\mathbf{c}_i$ is equal to the default hidden layer size of 768.

— *Pooled hidden layers*: The empirical evaluation conducted in BERT-as-a-Service [43] shows that using the last hidden layer gives the highest accuracy, but they also observed that it could also be more biased, since it is the closest layer to the output layer. Hence, it is advisable to select the second-to-last hidden layer or a combination of different layers. In implementing this idea in practice, we take the set of all $b+2$ hidden state vectors from the penultimate layer, namely, $\{\mathbf{z}_0^{11}, \mathbf{z}_1^{11}, \ldots, \mathbf{z}_{b+1}^{11}\}$ and mean/max pool them into one vector of size 768, which, after some non-linear activation, can be used as the chunk embedding vector $\mathbf{c}_i$. In addition, we can also follow a similar approach by selecting the last four hidden layers, namely, $\{\mathbf{z}_0^{9}, \mathbf{z}_1^{9}, \ldots, \mathbf{z}_{b+1}^{9}\}$, $\{\mathbf{z}_0^{10}, \mathbf{z}_1^{10}, \ldots, \mathbf{z}_{b+1}^{10}\}$, $\{\mathbf{z}_0^{11}, \mathbf{z}_1^{11}, \ldots, \mathbf{z}_{b+1}^{11}\}$, and $\{\mathbf{z}_0^{12}, \mathbf{z}_1^{12}, \ldots, \mathbf{z}_{b+1}^{12}\}$, and produce four mean/max pooled vectors in the same way. These four vectors and mean/max are pooled into one vector, which on activation can be used as the chunk embedding vector $\mathbf{c}_i$.

3.3 Document-Level Attention Pooling

Given the chunk embedding vectors $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{m_i}\}$, we need to aggregate them into an effective document vector $\mathbf{d}_i$ for document d_i. Since the chunks are sequential in nature, we can accomplish

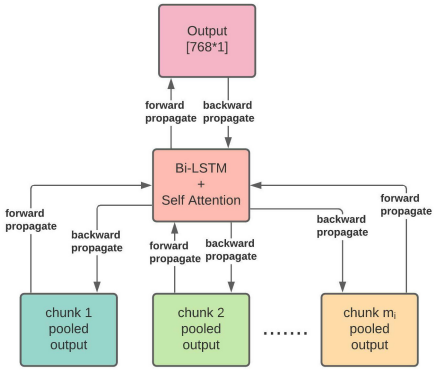

Fig. 4. Attention pooling via Bi-LSTM for document embeddings.

this using a recurrent Bi-LSTM model. However, not all chunks in a long document are equally important. It is crucial to score the chunks based on their importance in the document. For this, we introduce chunk-level attention within the Bi-LSTM model. Given a document, we input its chunk feature vectors $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{m_i}\}$ into the Bi-LSTM model. The output and hidden state vectors of the Bi-LSTM for chunk i are then obtained by concatenating the outputs and the hidden states in forward and backward pass, respectively. Formally,

$$\mathbf{o}_i = \overrightarrow{\mathbf{o}_i} \oplus \overleftarrow{\mathbf{o}_i}, \qquad \mathbf{h}_i = \overrightarrow{\mathbf{h}_i} \oplus \overleftarrow{\mathbf{h}_i},$$

where $\oplus$ denotes concatenation, $\rightarrow$ denotes forward and $\leftarrow$ denotes backward models, and the $\mathbf{h}_i$ and $\mathbf{o}_i$ denote the hidden and output state vectors for chunk c_i, respectively (i also denotes the ith element of the chunk sequence). The attention score α_i for each chunk is calculated by taking softmax over the product of outputs with the hidden state context vector. The document feature vector $\mathbf{d}_i$ (of size 768) is obtained by taking the weighted sum of the chunks according to their attention scores, normalized by the number of chunks for that document. Formally,

$$\alpha_i = \texttt{softmax}\left(\left\{\mathbf{o}_1^T\mathbf{h}_i, \mathbf{o}_2^T\mathbf{h}_i, \ldots, \mathbf{o}_{m_i}^T\mathbf{h}_i\right\}\right),$$
$$\mathbf{d}_i = \frac{\sum_{j=1}^{m_i} \alpha_j \cdot \mathbf{c}_j}{m_i}.$$

Figure 4 shows an illustration of the document level attention pooling step. At the bottom are the chunk embedding vectors $\mathbf{c}_i$ as inputs, which are passed to the Bi-LSTM and attention modules to create the document embedding $\mathbf{d}_i$.

3.4 Model Training

In the final stage of training, we feed each 768-dimensional document feature vector $\mathbf{d}_i$ to two additional fully connected linear layers FC_1 and FC_2 (see Figure 1), with size 601 and 1, respectively, with a leaky ReLU activation and a dropout layer applied to FC_1. The last layer FC_2 represents the output neuron to predict a target numeric variable. In other words, we concatenate the historic score y^{hist} (e.g., the previous year's value for stock volatility or return on assets, etc.) with the document vector $\mathbf{d}_i$ to use both the numerical and textual features. Formally,

$$\mathbf{d}_i = \mathbf{o}_{FC_1} \oplus y^{hist},$$

where $\mathbf{o}_{FC_1}$ denotes the output features vector from FC_1. Hence, FC_1 has 601 neurons, the first 600 of which are textual features, and the last one is the historical numeric value, all of which are

input to FC_2 to predict the target numeric score $\hat{y}$. The loss function is MSE or mean-squared error between the predicted and true target value.

4 EXPERIMENTS

We now showcase the effectiveness of our FETILDA framework on text regression tasks on very long financial documents. All of our experiments were conducted on a machine with 2.5 Ghz Intel Xeon Gold 6248 CPU, 768 GB memory, and a NVIDIA Tesla V100 GPU with 32 GB memory. The neural network models are implemented using PyTorch v1.10 (pytorch.org) and the HuggingFace library (huggingface.co). Our code and datasets are publicly available on github via https://github.com/Namir0806/FETILDA.

4.1 Data Description: 10-K Reports

A 10-K is a comprehensive report filed annually by a publicly traded company about its financial performance and is required by the U.S. SEC [18]. The SEC requires this report to keep investors aware of a company's financial condition and to allow them to have enough information before they buy or sell shares in the corporation, or before investing in the firm's corporate bonds.

10-Ks thus give a clearer picture of everything a company does and what kinds of risks it faces [18]. However, the length of 10-K reports has generally increased dramatically in recent years. According to a Wall Street Journal article [26], the average 10-K report is getting longer, from about 30,000 words in 2000 to about 42,000 words in 2013. In the article, GE finance chief Jeffrey Bornstein is reported to have said that not a retail investor on planet earth could get through it, let alone understand it. Our goal, therefore, is to extract the soft information contained in the textual data of these extremely lengthy 10-K reports, to better our predictions of forward-looking KPIs.

While the entire 10-K report is a very long disclosure document, Items 7/7A and 1A are considered as the important subsections in a 10-K report [2]. Item 7 (MD&A) gives the company's perspective on the business results of the past financial year. It is meant for the management to relate in its own words the analysis of their financial condition. Item 7A (Quantitative and Qualitative Disclosures about Market Risk) provides information about the company's exposure to market risk, such as interest rate risk, foreign currency exchange risk, commodity price risk, or equity price risk. Item 1A (Risk Factors) includes information about the most significant risks for a company or its securities. The risk factors are typically reported in order of their importance. However, it focuses on the risks themselves, and not necessarily on how the company addresses those risks. Some risks apply to the entire economy, some only to the specific industry sector or region, and some are directly related to the company.

We thus focus on Item 7/7A and Item 1A of the 10-K reports, which contain a treasure trove of soft information that can be leveraged for predictive analytics tasks. Since the industry standard is to only use quantitative data to predict future KPIs, we want to add the qualitative data coming from text into the mix, to achieve better predictions.

4.2 Datasets and Target Metrics

4.2.1 FIN10K Dataset [22]. The FIN10K dataset [22], contains Item 7 of 10-K reports of U.S. companies from 1996 to 2013, and the stock return volatilities 12 months before and after each report. Table 1 shows the statistics for this dataset. Following earlier work [38], we use the reports from 1996 to 2000 as training and validation data, and reports for each year from 2001 to 2006 as separate testing data. Further, we choose the first 80% of the reports from 1996 to 2000 as training data, and the remaining 20% as validation data. In the testing data, from 2001 to 2006, the number of documents is generally increasing, as well as the average document length, which almost doubles from 2001 to 2006.

Table 1. FIN10K Dataset [22] Statistics

Year	1996–2000	2001	2002	2003	2004	2005	2006
Number of total documents	8,703	1,825	2,023	2,866	2,861	2,698	2,564
Average document length	5,079.4	6,245.6	8,414.3	10,324.7	11,499.6	12,528.1	12,198.1

Table 2. U.S. Bank Dataset Statistics

	Item 7/7A	Item 1A
Number of total documents	5,321	
After extracting items	3,396	
Target data available	2,500	2,479
Average document length	12,589.75	4,435.69

Prediction Task: Volatility. The regression task is to predict the stock return volatilities, based on data from 12 months before and after each report. Volatility [38] is a common risk metric defined as the standard deviation of a stock's returns over a period of time. Historical volatilities are derived from time series of past stock market prices as a proxy for financial risk. Let S_t be the price of a stock at time t. Holding the stock for one period from time $t-1$ to time t results in a simple net return of $R_t = \frac{S_t}{S_{t-1}} - 1$ [40]. Therefore, the volatility of returns for a stock from time $t-n$ to t is defined as

$$v_{[t-n,t]} = \sqrt{\frac{\sum_{i=t-n}^{t}(R_i - \bar{R})^2}{n}}, \tag{1}$$

where $\bar{R} = \sum_{i=t-n}^{t} R_i/(n+1)$.

4.2.2 U.S. Banks Dataset. We collected the 10-K filings for all U.S. banks for the period between 2006 and 2016 (from the SEC EDGAR website: www.sec.gov/edgar), as well as the corresponding quantitative target data from the WRDS Center for Research in Security Prices (wrds-www.wharton.upenn.edu). While the entire 10-K report is a very long disclosure document, as noted above, Items 7/7A and 1A are considered as the important subsections in a 10-K report [2]. These subsections are themselves also quite long. The dataset statistics for the 10-K reports for all U.S. Banks for the period of 2006–2016 are reported in Table 2.

The 10-K reports for U.S. Banks (2006–2016) total 5,321 documents, but not all reports have both the Item 7/7A subsection. Of the total, 3,396 10-K reports have this important subsection. Furthermore, we found that not all banks have all the eight target KPI values that we need for regression. Of the 3,396 documents, we have 2,500 Item 7/7A and 2,479 Item 1A, with their eight metrics in full as target data, which makes up the final document set used in our experiments. The average document length (in terms of the number of words) is 12,590 for Item 7/7A, though the average length of 4437 is considerably shorter for Item 1, as noted in Table 2. We sort the documents chronologically from 2006 to 2016, and choose the first 80% of the data for training, and the remaining 20% as validation and testing data, with a 50/50 split between the latter two. In terms of target data normalization, for each of the eight target metrics, we performed min-max scaling to normalize the data for training.

Prediction Task: Bank KPIs. For U.S. banks our goal is to predict several KPI metrics using the 10-K reports. In particular, we focus on eight metrics that indicate either the performance or risk of a given bank: ROA, EPS, ROE, Tobin's Q Ratio, Tier 1 Capital Ratio, Leverage Ratio, Z-Score, and Market-to-Book Ratio. The target metrics are defined below.

— **Return on Assets (ROA):** ROA is calculated by dividing a company's net income by total assets:

$$\text{ROA} = \frac{\text{Net Income}}{\text{Total Assets}}. \tag{2}$$

Higher ROA shows more asset efficiency and productivity.

— **Return on Equity (ROE):** According to Reference [13], ROE is a measure of financial performance calculated by dividing net income by shareholders' equity:

$$\text{ROE} = \frac{\text{Net Income}}{\text{Total Equity}}. \tag{3}$$

— **Earning per share (EPS):** EPS is an indicator of a company's profitability. It is calculated as a company's profit divided by the outstanding shares of its common stock:

$$\text{EPS} = \frac{\text{Net Income} - \text{Preferred Dividends}}{\text{End-of-Period Common Shares Outstanding}}. \tag{4}$$

The higher a company's EPS, the more profitable per share it is.

— **Tobin's Q Ratio (TQR):** TQR represents the ratio of the market value of a firm's assets to the replacement cost of the firm's assets:

$$\text{TQR} = \frac{\text{Equity Market Value} + \text{Liabilities Book Value}}{\text{Equity Book Value} + \text{Liabilities Book Value}}. \tag{5}$$

This ratio indicates how the market views the managers' prospects of using firm's asset to generate future value for investors of the firm.

— **Leverage Ratio (LR):** The Leverage Ratio measures the extent of debt financing for a firm, therefore assesses the ability of a company to meet its financial obligations. It is given as

$$\text{LR} = \frac{\text{Average Total Assets}}{\text{Average Equity}}. \tag{6}$$

— **Tier 1 Capital Ratio (T1CR):** The Tier 1 capital ratio is the ratio of a bank's core Tier 1 capital to its total risk-weighted assets:

$$\text{T1CR} = \frac{\text{Tier 1 Capital}}{\text{Total Risk-Weighted Assets}}. \tag{7}$$

These risk-weighted assets include all assets that are systematically weighted for credit risk.

— **Z-score (Z):** The Z-score links a bank's capitalization with its return (ROA) and risk (volatility of returns).

$$\text{Z-Score} = \frac{\text{ROA} + \text{CAR}}{\sigma(\text{ROA})}, \tag{8}$$

where σ(ROA) is the standard deviation of ROA for a specific time period, and CAR is the capital-to-assets ratio.

— **Market-to-Book Ratio (MBR):** The Market-to-Book Ratio is used to evaluate a company's current market value relative to its book value, and is calculated by dividing the current stock price of all outstanding shares by the book value:

$$\text{MBR} = \frac{\text{Market Capitalization}}{\text{Total Book Value}}. \tag{9}$$

4.3 Methods

We now outline the results of our framework on both the FIN10K and U.S. Banks datasets. Overall, we experiment with four different types of methods, three of which being baseline methods with which we compare the FETILDA framework to evaluate the performance of our approach. To effectively compare different methods, all results report the MSE. The methods are as follows:

- **Numeric Regression** [49]: We compare with linear regression (**LR**) and support vector regression (**SVR**) models. These baselines methods take the historical score y^{hist} and run (bivariate) regression model on it to predict the target variable. These method therefore utilize only numerical data. The SVR method (from Reference [38]) uses the logarithm of the historic volatility for the prior 12 months.
- **Word Feature Vector Models**: These are traditional but still quite effective baseline methods, based on the TF-IDF or LOG1P word vector representations, as given below:
  TF-IDF [17]: In this baseline method, we use the term frequency–inverse document frequency features, as the input document embeddings, which help in scoring important words.
  - **LOG1P+** [38]: This is the method used in the volatility regression task proposed by Tsai and Wang [38]. The word features are formed using LOG1P, calculated as LOG1P = $log(1 + TC(t, \mathbf{d}))$, where $TC(t, \mathbf{d})$ denotes the term count of a word t in a given document $\mathbf{d}$. Furthermore, the logarithm of the stock return volatility 12 months before each report is used as an additional numeric feature, and together, the word features and numeric features are input into a Support Vector Regression model.
- **FETILDA Framework**: Here, we use our framework, detailed in Section 3, with different large language models, including domain specific ones, as listed below. All methods use the default pooling strategy.
  - **FETILDA w/BERT**: We use [10] as the underlying language model, setting the chunk size to 512 tokens.
  - **FETILDA w/FinBERT**: Here, we use FinBERT [16] as the LM, which was pre-trained on 10-K, 10-Q, and analyst reports, with chunk size of 512 tokens.
  - **FETILDA w/Longformer**: To test the effectiveness of a bigger block size with a pretrained model, we use our approach with Longformer [4] as the underlying language model, setting the chunk size to 4,096 tokens.
  - **FETILDA w/Nystromformer**: For an even bigger block size, but without a pretrained model (that is, training from scratch), we use our approach with Nystromformer [44] as the underlying language model, setting the chunk size to 8,192 tokens, the number of layers to one, and the number of attention heads to eight.
- **Truncated LMs**: Here, we experiment with truncated LM baselines, where the each model uses only the first chunk of each document and discards the rest of the document. The rest of the model training is identical to the process detailed in Section 3.4. These baselines thus serve as ablated versions of our chunking approach for each of the corresponding LMs.
  - **BERT Truncate**: Here, we simply use the pretrained BERT [10] model with chunk size of 512.
  - **FinBERT Truncate**: Here, we use the financial domain pretrained FinBERT model [16] as the underlying language model to learn on the first chunk of the document, with chunk size of 512 tokens.
  - **Longformer Truncate**: This baseline uses pretrained Longformer [4], with 4,096 as the chuck size, which is considerably larger than the BERT/FinBERT baselines.

- **Nystromformer Truncate**: For Nystromformer [44], we set the chuck size as 8,192, which is double that for Longformer, so that we can evaluate the effect of truncating on with a very large context or chunk size. We further use eight attention heads, and one tranformer layer.
- **BigBird Truncate**: Here, we use the large context BigBird [48] model, with a chunk size of 4,096. This also shows the comparison with a sparse-attention-based approach.

With all four versions of FETILDA, namely, using BERT, FinBERT, Longformer, and Nystromformer, and baseline numerical regressions, word feature vector models, and truncated LMs, namely, BERT, FinBERT, Longformer, Nystromformer, and BigBird, we performed an extensive set of experiments, evaluating our approach in predicting stock return volatility for the FIN10K dataset [22], and all eight different KPI metrics for the U.S. banks dataset. For the U.S. banks dataset, the historical scores are numeric values of each of the eight metrics in the previous year of the report. For the FIN10K dataset, they are the stock return volatilities 12 months before each report. In addition to applying our approach as described in Section 3.2 with fully unfrozen LM layers, enabling model fine-tuning, we also report the effect of freezing all the LM layers and freezing only the last layer in FETILDA when we apply it on both the U.S. banks dataset and the FIN10K dataset [22]. This allows us to compare the effect of fine-tuning versus the default pre-training approach.

4.4 Comparative Performance Results

In all four versions of FETILDA and truncated LMs, we train the model with varying learning rates from 10^{-1} to 10^{-8}, and pick the epoch and parameters with the best validation loss. Due to the memory constraint of 32 GB, for a given document, the GPU can only process up to around 20,480 tokens at a time, so we truncate the rest if a document goes beyond that length. However, this only happens for a minority of cases in our experiments, and we do not truncate at all in our experiments with fully frozen language models. As mentioned above, we use the default pooling strategy to extract chunk embedding vectors, and among the various FinBERT alternatives, we use the Huang et al. FinBERT [16] model. We empirically show below that both these choices are in fact the best ones among the different pooling and FinBERT variants, respectively. Finally, for both FETILDA (w/BERT, w/FinBERT, w/LongFormer, and w/Nystromformer), all truncated LMs, and TF-IDF we select the best among the following regression models based on the validation data: (1) Linear Regression, (2) Support Vector Regression, using a RBF Kernel with $C = 0.1$ and $\epsilon = 0.0001$, and (3) Kernel Ridge Regression, using a RBF Kernel with $\alpha = 0.1$ and $\gamma = 0.1$, in addition to the variant based on the predicted output (from FC_2) with MSE loss.

4.4.1 FIN10K Dataset. Table 3 compares the performance of FETILDA variants with the other baseline methods listed above. Note that LOG1P+:ALL refers to the model trained on the entire original text using the LOG1P features, and LOG1P+:SEN refers to the model trained on only the sentiment bearing words taken from the L&M dictionary [24]. For LOG1P+:ALL and LOG1P+:SEN, we report the results for these methods directly from their paper [38]. We also include the results for the TF-IDF baseline. Among their methods, LOG1P+:SEN performs the best for all years, except 2001. For the average over all test years, LOG1P+:SEN performs better than TF-IDF, even though TF-IDF performs better than LOG1P+:SEN for 2001, 2002, and 2006. However, as we can observe, with the exception of 2003, FETILDA outperforms LOG1P+:SEN by a large margin. Interestingly, FETILDA w/FinBERT outperforms both BERT and Longformer on all the metrics. It is the best performing model over all the years, with the exception of 2003. *Looking at the last column, which shows the average performance across the years 2001–2006, FETILDA w/FinBERT is the best; it outperforms all previous baselines by a significant margin, establishing new SOTA results.*

Table 3. MSE Results on FIN10K

Model\Year	2001	2002	2003	2004	2005	2006	**Average**
SVR	0.174700	0.160020	0.187340	0.144210	0.136470	0.146380	0.150860
LOG1P+:ALL	0.180820	0.171750	0.171570	0.128790	0.130380	0.142870	0.154360
LOG1P+:SEN	0.185060	0.163670	**0.157950**	0.128220	0.130290	0.139980	0.150860
TF-IDF	0.123816	0.121450	0.218520	0.176087	0.148645	0.138113	0.154438
Truncated LMs							
BERT Truncate (Fully Unfrozen)	0.123519	0.109091	0.188643	0.117910	0.099342	0.094656	0.122193
FinBERT Truncate (Fully Unfrozen)	0.123174	0.108862	0.187908	0.117288	0.098827	0.094146	0.121701
Longformer Truncate (Fully Unfrozen)	0.123086	0.108659	0.187317	0.116490	0.097997	0.093773	0.121220
BigBird Truncate (Fully Unfrozen)	0.124220	0.108205	0.185940	0.115778	0.097545	0.092892	0.120763
Nystromformer Truncate (Fully Unfrozen)	0.123561	0.108854	0.187848	0.116714	0.098098	0.093215	0.121382
FETILDA w/ LMs							
FETILDA w/BERT (Fully Unfrozen)	0.128406	0.111145	0.180670	0.111339	0.094401	0.091456	0.119569
FETILDA w/FinBERT (Fully Unfrozen)	0.123321	**0.108134**	0.172562	**0.106124**	**0.090766**	**0.088401**	**0.114885**
FETILDA w/Longformer (Fully Unfrozen)	0.124797	0.109595	0.183509	0.113019	0.094623	0.090408	0.119325
FETILDA w/Nystromformer (Fully Unfrozen)	0.120945	0.108224	0.174019	0.109716	0.095050	0.093098	0.116842
FETILDA w/BERT (Last Layer Frozen)	0.129132	0.111559	0.181691	0.110962	0.093300	0.089595	0.119373
FETILDA w/FinBERT (Last Layer Frozen)	0.125969	0.109420	0.176483	0.108349	0.092103	0.089228	0.116925
FETILDA w/Longformer (Last Layer Frozen)	0.135215	0.114627	0.193750	0.117404	0.096162	0.089970	0.124521
FETILDA w/BERT (Fully Frozen)	0.121354	0.108529	0.175446	0.108837	0.093004	0.090500	0.116278
FETILDA w/FinBERT (Fully Frozen)	**0.118620**	0.113750	0.159487	0.108527	0.097878	0.095545	0.115635
FETILDA w/Longformer (Fully Frozen)	0.126380	0.109627	0.169686	0.108116	0.091884	0.089902	0.115932

Best results in bold.

Table 3 also shows what happens to the FETILDA variants if we freeze the layers of the language model and use only the pre-trained embeddings, compared to fine-tuning through unfreezing all the layers or only freezing the last layer. Interestingly, for the larger FIN10K dataset, fine-tuning results in a much better model for FinBERT, although fully frozen FinBERT does well for 2001 and 2003, but not so much for BERT and Longformer. As such, the domain-specific pre-training in FinBERT followed by fine-tuning results in the best overall model.

Comparing with Truncated Models. Table 3 also shows how our evaluation framework compares with LMs in Finance and Long Document LMs. Since these models have a fixed context/chuck size and do not create document level representations, we truncate the models to use only the first chunk from each document. However, we study the effect of models with larger chuck sizes, ranging from 512 used for BERT/FinBERT, to 4,096 used in Longformer and BigBird (which also uses sparse attention), to 8,192 used in Nystromformer. As we can see, none of the truncated LM baselines methods outperform FETILDA w/FinBERT. Thus, truncating the documents is not an effective strategy. This provides strong evidence on the advantage of our framework that considers all the chunks so that long documents can be processed without significant information loss, and that further allows fair comparisons between different underlying LMs. For example, FinBERT Truncate achieves almost the same performance as Longformer Truncate, which may lead one to prematurely conclude the futility of pre-trained domain-specific models such as FinBERT. However, truncated FinBERT can only "see" the first 512 tokens of the document, whereas truncated Longformer uses the first 4,096 tokens. However, when utilizing our chunking framework, we observe that FinBERT can unleash its full potential and outperform Longformer, showcasing the effectiveness of pre-trained domain-specific LMs in finance. This due to the fact that now FinBERT and Longformer, when used through FETILDA, can exploit the full text from the long document.

4.4.2 U.S. Banks Dataset: Item 7/7A. Table 4 shows the performance comparison between the four versions of our approach on Item 7/7A and baseline methods: truncated versions of all BERT,

Table 4. MSE Results on Item 7/7A

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
TF-IDF	0.000879	0.010422	0.001022	0.022000	0.000767	0.002594	0.028926	0.005765
Linear Regression	0.001432	0.010096	0.001564	0.022587	0.000306	0.002441	0.030760	0.005757
Truncated LMs								
BERT Truncate (Fully Unfrozen)	0.000769	0.007790	0.000873	0.020440	0.000396	0.002607	0.029508	0.005641
FinBERT Truncate (Fully Unfrozen)	0.000827	0.007458	0.000855	0.016940	0.000308	0.002599	0.028715	0.005655
Longformer Truncate (Fully Unfrozen)	0.000817	0.007462	0.001056	0.020103	0.000272	0.002446	0.033579	0.005889
BigBird Truncate (Fully Unfrozen)	0.000781	0.007422	0.000909	0.017403	0.000240	0.002576	0.029046	0.005617
Nystromformer Truncate (Fully Unfrozen)	0.000879	0.008459	0.000853	0.016444	**0.000239**	0.002596	**0.027976**	0.005591
FETILDA w/ LMs								
FETILDA w/BERT (Fully Unfrozen)	0.000796	0.009227	0.000897	0.021409	0.000325	0.002502	0.029505	0.005651
FETILDA w/FinBERT (Fully Unfrozen)	**0.000746**	0.008901	0.000932	0.019150	0.000317	0.002535	0.029516	0.005657
FETILDA w/Longformer (Fully Unfrozen)	0.000813	0.008507	0.000858	0.017358	0.000296	0.002467	0.028697	0.005683
FETILDA w/Nystromformer (Fully Unfrozen)	0.000788	**0.007338**	**0.000835**	0.016862	0.000310	0.002503	0.029397	0.005735
FETILDA w/BERT (Last Layer Frozen)	0.000850	0.009903	0.000960	0.021728	0.000306	0.002469	0.029203	0.005798
FETILDA w/FinBERT (Last Layer Frozen)	0.000844	0.008543	0.000988	0.021425	0.000304	0.002445	0.029637	0.005678
FETILDA w/Longformer (Last Layer Frozen)	0.000849	0.008356	0.000851	**0.016436**	0.000291	**0.002419**	0.029011	**0.005481**
FETILDA w/BERT (Fully Frozen)	0.000890	0.010052	0.001109	0.022748	0.000328	0.002581	0.028966	0.005950
FETILDA w/FinBERT (Fully Frozen)	0.001093	0.009401	0.001906	0.021882	0.000447	0.002514	0.030094	0.005695
FETILDA w/Longformer (Fully Frozen)	0.000801	0.008501	0.000876	0.019053	0.000308	0.002436	0.028965	0.005957

Best results in bold per KPI.

FinBERT, Longformer, Nystromformer, and BigBird, TF-IDF for textual modeling with historic scores, and linear regression for numerical modeling. *For most metrics, our method outperforms the baseline methods (TF-IDF and linear regression), with FETILDA w/Longformer and FETILDA w/Nystromformer performing the best in a majority of cases.* As such, FETILDA variants using all of the chunks outperform all other baselines on six of the eight metrics. In addition, we also see a significant edge in the performance of FETILDA w/FinBERT in the prediction of ROA target values.

Table 4 also shows the comparisons between applying FETILDA w/ LMs, and using LMs directly in truncated mode. As we can see, the shorter window size models, namely, BERT and FinBERT, when using all chunks, performed better in some metrics and worse in some metrics compared to their truncated counterparts. This may be due to the chunk attention weights not being well assigned in the document-level attention pooling phase in a portion of shorter documents, as we examine in Section 4.6.1. As for longer window size LMs, namely, Longformer and Nystromformer, we can observe that for the majority of the metrics, our framework is able to further improve the performance of longer window LMs for long documents on six of the eight metrics. On the two metrics where Nystromformer truncate performs best, it demonstrates the benefit of having a larger window size, as the window size in this case is 8,196.

4.4.3 U.S. Banks: Item 1A Section. Next, we report results on Item 1A. Table 5 shows the performance comparison between our approach and baseline methods, namely, truncated versions of BERT, FinBERT, Longformer, Nystromformer, and BigBird; TF-IDF for textual modeling with historic scores, and linear regression for numerical modeling. In five of eight metrics, FETILDA w/ LMs with our framework outperform other methods, with FETIDA w/Longformer [4] performing the best in two metrics, and FETILDA w/BERT and FinBERT performing the best in the other three metrics. Longformer truncate and BigBird truncate, both with a window size of 4,096, performed best on two of the metrics, demonstrating the benefits of a larger context size. The Item 1A section is generally a shorter document, as shown in Table 2, and FETILDA is designed for long documents, so when our framework trains on this corpus, it may suffer from the uneven spread of attention due to the document-level attention pooling focusing on a few select chunks while ignoring the rest, as demonstrated in the case study analysis below (see Section 4.6.2).

Table 5. MSE Results on Item 1A

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
TF-IDF	0.001153	0.009350	0.000970	0.018660	0.000322	0.003117	**0.029103**	0.005922
Linear Regression	0.001407	0.010174	0.001577	0.022500	0.000299	0.002534	0.032102	0.005802
Truncated LMs								
BERT Truncate (Fully Unfrozen)	0.000783	0.009169	0.000955	0.019806	0.000324	0.002655	0.031074	0.005991
FinBERT Truncate (Fully Unfrozen)	0.000774	0.007453	0.000861	0.017555	0.000290	0.002597	0.031737	0.005979
Longformer Truncate (Fully Unfrozen)	0.000925	0.008200	0.001645	0.017472	**0.000221**	0.002553	0.034199	0.005923
BigBird Truncate (Fully Unfrozen)	0.000816	**0.006908**	0.000963	0.015997	0.000318	0.002632	0.031801	0.006205
Nystromformer Truncate (Fully Unfrozen)	0.000790	0.007385	0.000882	0.017392	0.000275	0.002614	0.030956	0.006388
FETILDA w/ LMs								
FETILDA w/BERT (Fully Unfrozen)	0.000811	0.008520	**0.000820**	0.019151	0.001353	0.002559	0.029614	0.004944
FETILDA w/FinBERT (Fully Unfrozen)	0.000867	0.008671	0.001171	0.017383	0.000385	0.002560	0.030583	0.004937
FETILDA w/Longformer (Fully Unfrozen)	0.000790	0.007940	0.000826	**0.015620**	0.000937	0.002527	0.030130	**0.004555**
FETILDA w/Nystromformer (Fully Unfrozen)	0.000780	0.007659	0.000925	0.016263	0.000226	0.002831	0.029426	0.005640
FETILDA w/BERT (Last Layer Frozen)	**0.000774**	0.007803	0.000824	0.017883	0.000726	0.002751	0.029729	0.004943
FETILDA w/FinBERT (Last Layer Frozen)	0.000850	0.008814	0.000834	0.018282	0.000485	0.002612	0.030115	0.004967
FETILDA w/Longformer (Last Layer Frozen)	0.000795	0.007409	0.000821	0.018100	0.000242	0.002715	0.030415	0.004894
FETILDA w/BERT (Fully Frozen)	0.000856	0.008788	0.001076	0.018572	0.000315	0.010919	0.030225	0.004908
FETILDA w/FinBERT (Fully Frozen)	0.000976	0.008626	0.001274	0.018254	0.000428	**0.002471**	0.032155	0.004911
FETILDA w/Longformer (Fully Frozen)	0.000811	0.008053	0.000854	0.018429	0.000930	0.002619	0.034284	0.004955

Best results in bold per KPI.

Table 5 also shows the comparisons between applying FETILDA with LMs and using LMs directly in truncate mode. As for the shorter window size models, namely, BERT and FinBERT, when applied in FETILDA, these LMs performed better in a majority of metrics compared to using the it directly in truncate mode, with BERT performing better in FETILDA for all metrics compared to BERT in truncate mode. As for longer window size LMs, namely, Longformer and Nystromformer, we can observe that for the majority of the metrics, our framework is again able to deliver extra performance on longer window LMs for long documents. Finally, even for these shorter documents, using all of the chunks via the FETILDA approach results in the best performing model on five of eight metrics, and a second best on the remaining three metrics.

4.5 Algorithmic Choices

Having shown the effectiveness of our FETILDA framework, we now present some results to justify some of the algorithmic choices, such as which document-level pooling strategy and which chunk-level pooling strategy does the best, and which FinBERT model performs the best.

4.5.1 Effectiveness of Document-level Attention Pooling. As elaborated in Section 3.3, in the FETILDA framework, our approach to generating a document-level embedding from chunk-level embeddings is to pool them together using a Bi-LSTM layer with self attention. In this way, each chunk is weighted by its importance and aggregated together. To showcase the effectiveness of this approach on a large corpus of long documents, we performed three sets of additional experiments on the FIN10K dataset: (i) truncating the document, so that we use only the first chunk of every document, (ii) mean-pooling, and (iii) max-pooling the chunks together into one document-level embedding. As we can see in Table 6, compared to these three simpler approaches, the advantage and gains in using our document-level attention pooling approach are evident. Moreover, we can observe that mean-pooling all the chunks together even performs slightly worse than just using the first chunk of every document. This means that our attention-based pooling approach effectively captures information from all the chunks, whereas simpler pooling methods are not able to do so.

Table 6. Document Level Attention Pooling: MSE Results on FETILDA w/FinBERT (Fully Unfrozen) Variants

Model\Year	2001	2002	2003	2004	2005	2006	Average
FETILDA w/FinBERT (Bi-LSTM + Self Attention)	0.123321	**0.108134**	**0.172562**	**0.106124**	**0.090766**	**0.088401**	**0.114885**
FETILDA w/FinBERT (Only Using First Chunk)	0.123174	0.108862	0.187908	0.117288	0.098827	0.094146	0.121701
FETILDA w/FinBERT (Mean-Pooling Chunks)	0.122950	0.108655	0.187268	0.116554	0.098074	0.093308	0.121135
FETILDA w/FinBERT (Max-Pooling Chunks)	**0.122072**	0.108241	0.186018	0.115826	0.097591	0.092913	0.120444

Best results in bold.

Table 7. Comparison of Different Chunk-level Pooling Methods

Results\Methods	Mean pooling		Max pooling		Default pooling
	Second-to-last layer	Last four layers	Second-to-last layer	Last four layers	Last layer
Validation MSE	0.0011465	0.0012064	0.0011102	0.0011188	**0.0010205**
Testing MSE	0.0008547	0.0008221	0.0007686	0.0008820	**0.0007458**

Table 8. Comparison of Three Different Models of FinBERT

Results\Models	Araci [3]	DeSola et al. [9]	Huang et al. [16]
Validation loss	0.0011482	0.0010539	**0.0010205**
Testing error	0.0007781	0.0008682	**0.0007458**

4.5.2 Chunk-level Pooling Strategy. Recall that in Section 3.2, we outlined several chunk-level pooling strategies to create the final chunk embeddings. These include: (1) the default pooling method (default pooler output) using the hidden state of the first token of the last layer, (2) mean pooling method using the hidden states of the second-to-last layer, (3) mean pooling method using the hidden states of the last four layers, (4) max pooling method using the hidden states of the second-to-last layer, and (5) max pooling method using the hidden states of the last four layers. In Table 7, we present the comparative MSE results for these alternatives on Item 7/7A for predicting ROA. We observe that the default pooler output yields the best results for both validation and testing datasets. We thus chose the default pooling method using the hidden state of first token of the last layer, and this is used for the different versions of FETILDA in our experiments above.

4.5.3 FinBERT Variants. As discussed in related work, there are four different FinBERT approaches proposed recently. Out these, the implementation for Liu et al. FinBERT [23] is not publicly available. We therefore compare the three FinBERT implementations that are available: Araci [3], DeSola [9], and Huang et al. [16]. Table 8 shows the MSE results when predicting ROA using both textual data from Item 7/7A and numeric historic data (using a learning rate of 0.001) for the U.S. Banks dataset. The results show that Huang et al. implementation results in the best performance. We thus choose the Huang et. al FinBERT [16] as the underlying FinBERT model for FETILDA. Recall that this FinBERT model was pre-trained on a very huge financial corpus containing 10-K and 10-Q reports, earnings call transcripts, and analyst reports.

4.6 Qualitative Analysis and Case Study

4.6.1 Sentence Sentiment and Document-level Attention Pooling. To discover insights into the nature of the contents in the 10-K reports we have trained on, we first performed a sentiment analysis of the the sentences contained in these documents, both in the FIN10K dataset and U.S. Banks dataset, using the finbert-tone model developed by Huang et al. [16]. This model was fine-tuned on 10,000 manually annotated (positive, negative, neutral) sentences from analyst reports, and classifies a given sentence with one of these three labels with a score between 0 and 1. Tables 9

Table 9. Sentiment Analysis of Documents from the FIN10K Dataset

	1996–2000	2001	2002	2003	2004	2005	2006
Number of neutral sentences	969,556	252,398	396,105	691,779	734,407	752,487	701,754
Percentage of neutral sentences	75.1%	75.5%	76.7%	78.0%	76.7%	77.3%	79.4%
Neutral average score	0.969249	0.967979	0.969975	0.971859	0.970663	0.971246	0.973817
Number of positive sentences	155,546	34,816	41,887	68,310	88,937	94,444	91,481
Percentage of positive sentences	12.0%	10.4%	8.1%	7.7%	9.3%	9.7%	10.3%
Positive average score	0.931751	0.928670	0.927051	0.927983	0.935404	0.937857	0.940044
Number of negative sentences	166,200	46,983	78,609	127,057	133,900	126,587	90,814
Percentage of negative sentences	12.9%	14.1%	15.2%	14.3%	14.0%	13.0%	10.3%
Negative average score	0.924655	0.920516	0.921983	0.921626	0.919780	0.917041	0.913382

Table 10. Sentiment Analysis of Documents from the U.S. Banks Dataset

	Item 7	Item 1A
Number of neutral sentences	977.863	216.876
Percentage of neutral sentences	79.9%	52.6%
Neutral average score	0.971906	0.945267
Number of positive sentences	129.278	14.170
Percentage of positive sentences	10.6%	3.4%
Positive average score	0.923350	0.892079
Number of negative sentences	116.839	181.097
Percentage of negative sentences	9.5%	43.9%
Negative average score	0.909770	0.944259

and 10 show the results of this analysis. As we can see, in the Item 7 section of 10-K reports, which forms the entirety of the training data for the FIN10K experiments and one portion of the training data for the U.S. Banks experiments, the majority of sentences, around 70% to 75%, are all neutral sentences, with over 90% confidence (i.e., the average score value) from the classifier. This indicates that the majority of the textual content that we are training on is neutral content. Only around 10% to 15% of the sentences are negative, and a smaller percentage of sentences are positive. And in the Item 1A section of 10-K reports in the U.S. Banks dataset, which is a shorter type of document focusing on risk factors, over half of the sentences are neutral, with around 40% of the sentences negative and a small percentage of sentences positive. This brings up the question of what the model will decide to focus on when making quantitative predictions, that is, whether attention will be paid mostly to neutral sentences, or sentences that indicate some sort of positive or negative sentiment. We explore this more in Section 4.6.2.

We also examined the chunk attention weights in the document-level attention pooling phase (see Section 3.3). Since this phase evaluates different chunks based on their importance and weighs the tokens in each chunk accordingly, the process can affect which parts of the document the model ends up focusing on to capture signals from, to make predictions. We decided to focus on the FIN10K dataset for this analysis as it is a larger dataset, which can produce more representative conclusions. In examining documents where FETILDA w/FinBERT did not perform well, we discovered that on account of the shortness of the document, the chunk attention weights focused on one of the chunks with a very high weight, and on others with very low weights, and some chunks even had zero weight. This was due to the fact that some chunks had mundane regulatory content or padding at the end, but they also contained sentences that could be of importance to

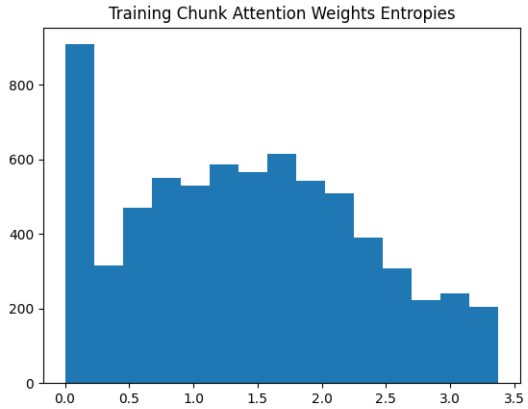

Fig. 5. Entropies of chunk attention weights produced by FETILDA w/FinBERT after training on the FIN10K dataset.

volatility change. It was evident in this case that much of the information contained in the less weighed chunks was lost.

This led us to further investigate the relationship between the length of the document and the entropy of the chunk attention weights, and how much this factor leads to information loss that affects performance. We therefore calculated the entropy of all the chunk attention weights produced after training. Given a list of chunk attention weights $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ (see Section 3.3) for document d, which all sum up to 1, since they are the result of a softmax function, the chunk entropy is given as

$$H = -\sum_{i=1}^{m} \alpha_i \log(\alpha_i). \tag{10}$$

Higher entropy would mean that the weights are more spread out, giving attention to each chunk more evenly, and lower entropy would mean that the weights are skewed, giving attention to few chunks while ignoring the rest.

Figure 5 shows a histogram of the results. Overall, the entropies ranged between 0 and 3.5, and though we see a rough normal distribution for the majority of documents, a significant portion of the documents had low entropies between 0 and 0.5. For these documents, the chunk attention focuses only on a single or a few chunks. To examine the implications of this further, we divided the training set into seven buckets based on the chunk entropy value, and calculated their respective average document lengths and the mean training losses (in terms of MSE), as shown in Table 11. As we can observe, the length of the document has a proportional relationship with the entropy of the chunk attention weights, which in turn has an inverse relationship with the MSE loss (or proportional relationship with model performance). The longer documents have higher entropies, and the shorter documents have lower entropies. In turn, lower entropy documents have worse performance, producing higher losses, and higher entropy documents have better performance, producing lower losses.

From this analysis, we conclude that our approach performs better on longer documents than on shorter documents, because its document-level attention pooling weighs chunks more evenly in longer documents than in shorter documents. This makes sense, as our approach is designed for long documents, and in the training process, documents of varying lengths are fed to the framework, resulting in a generalized model that performs well on long documents. Therefore, to handle shorter documents, a corpus could be divided into several buckets based on document length, and

Table 11. Analysis of Chunk Entropies Produced by FETILDA w/FinBERT on the FIN10K

	Number of documents	Average document length	Mean training loss (MSE)
Entropy within [0.0, 0.5]	1,343	1,893.3	0.281587
Entropy within [0.5, 1.0]	1,116	2,792.5	0.247949
Entropy within [1.0, 1.5]	1,273	3,594.5	0.221757
Entropy within [1.5, 2.0]	1,299	4,746.3	0.177632
Entropy within [2.0, 2.5]	991	6,478.5	0.171708
Entropy within [2.5, 3.0]	566	9,187.0	0.181916
Entropy within [3.0, 3.5]	374	14,200.0	0.148555

separate models can be trained on different buckets of documents, which can fine-tune each model more specifically tailored to the document length. This is part of future investigation.

4.6.2 Case Study Analysis. To gain insights into what our most effective model, FETILDA w/FinBERT, learned from the FIN10K dataset, we designed a method to extract the "important" sentences learned from the data. For a given document, we rated the "importance" of each unique word in the following way. For a given word-token t_i that appears in chunk k of size b, we sum up the attention scores it gets from all other words in the chunk, that is, how much attention it was paid to when querying each word in the chunk. Thereafter, we take the weight α_k that has been given to chunk k by the self-attention mechanism detailed in Section 3.3, which determines how "important" each chunk is, and multiply it with the sum above to get the final score for this instance of t_i, given as follows:

$$s(t_i) = \alpha_k \sum_{j=1}^{b} a_{ji},$$

where a_{ji} is the attention that t_i gets from token t_j (in the given chuck k). Next, for each sentence, we aggregate the scores of each word together into a sentence attention score by averaging them over the length of the sentence.

Using this approach, we generated the top 30 sentences based on their attention score, from a document that our method performed well on in the year that it performed best on, namely, the 10-K report of *Microtune Inc. in 2006*, and a document that it performed poorly on in the year that it performed worst on, namely, the 10-K report of *Federal Screw Works in 2003*, in terms of squared error when predicting price log volatility. Some example sentences are shown in Table 12. Looking at these sentences, we can see a clear difference. For the good performance case, the top sentences were more informative to the company's future prospects, whereas for the bad performance case, the top sentences were more mundane accounting-related sentences.

To analyze these sentences more objectively, we applied sentiment analysis using the finbert-tone model developed by Huang et al. [16]. Table 13 shows the general statistics from the result of this sentiment analysis. In both cases, as a matter of course, a good number of neutral sentences garnered attention, since most of the 10-K reports are neutral sentences that are part of mundane regulatory filings. In the case of good performance for *Microtune, Inc., 2006*, attention was paid more to negative sentences that are classified to be negative with a high score, and some attention was paid to positive sentences that are classified to be positive with a medium score, but in the case of bad performance for *Federal Screw Works, 2003*, we see that attention was paid mostly to neutral sentences that are classified to be neutral with a very high score, and to just two negative sentences with a medium score, and to one positive sentence with a low score. Thus, for better performance, it is important for the attention in the framework to focus on the key parts of the document that are indicative of future trends.

Table 12. Examples of Top Attention Sentences from the 10-K Report of Microtune Inc. in 2006 (Good Performance) and Federal Screw Works in 2003 (Poor Performance), from the FIN10K Dataset, Using FETILDA w/FinBERT

Microtune Inc., 2006: Good Performance	**Federal Screw Works, 2003: Poor Performance**
"Further, several existing and potential customers have substantial internal technological capabilities and could develop products internally that compete with or replace our products."	"The decrease resulted primarily from a decrease in inventories."
"A decision by any of our significant customers to internally design and manufacture products that compete with our products could have a material adverse effect on our business and results of operations."	"Sales price increases in each of these years were insignificant."
"We believe that our future results of operations will continue to depend on the success of our largest customers, on our ability to sell existing and new products to these customers in significant quantities."	"Critical accounting policies the accompanying financial statements have been prepared in conformity with accounting principles."
"We compete with, or may in the future compete with, a number of major domestic and international suppliers of integrated circuit and system modules in the cable, digital TV and automotive markets."	"In an effort to increase the plan assets of the qualified pension plans, the company contributed $2,850,000 to the plans' funding in the fourth quarter of fiscal 2003."
"Our international operations, including our operations in Germany, Taiwan, Japan, China and Korea, the operations of our international suppliers and our overall financial results may be adversely affected by events that occur in or otherwise affect these countries."	"Accordingly, in the fourth quarter of fiscal 2003, the company recorded a non-cash charge of $5,080,000, after-tax, related to the additional minimum liability for certain underfunded pension plans which increased accumulated other comprehensive loss in shareholders' equity."
"We cannot assure you that any acquisition or joint venture will be successfully integrated with our operations and the failure to avoid these or other risks associated with such acquisitions or investments could have a material adverse effect on our business, financial condition and results of operations."	"Inventories were reduced to reflect lower demand from our automotive customers and also to reflect the elimination of strike banks required earlier but no longer necessary with the signing of a new four year contract with the employees of our Romulus division effective February 1, 2003."
"Many of these technologies compete effectively with cable modem and cable telephony services and do not require RF tuners like the ones that we sell."	"Further, the charge did not impact net income, and will reverse should the fair value of the pension plans' assets again exceed the accumulated benefit obligations at March 31, 2004."

Table 13. Sentiment Analysis of Top 30 Key Sentences Generated by FETILDA w/FinBERT from the 10-K Reports of Microtune Inc. in 2006 and Federal Screw Works in 2003, for the FIN10K Dataset

	Microtune Inc., 2006	Federal Screw Works, 2003
Number of neutral sentences	19	27
Neutral average score	0.957843	0.983552
Number of positive sentences	3	1
Positive average score	0.858830	0.618559
Number of negative sentences	8	2
Negative average score	0.935866	0.863191

Table 14. Sentiment Analysis of all the Sentences from the 10-K Reports of Microtune Inc. in 2006 and Federal Screw Works in 2003, from the FIN10K Dataset

	Microtune Inc., 2006	Federal Screw Works, 2003
Number of neutral sentences	540	75
Percentage of neutral sentences	70.6%	79.8%
Neutral average score	0.957420	0.976197
Number of positive sentences	47	7
Percentage of positive sentences	6.1%	7.4%
Positive average score	0.914375	0.977220
Number of negative sentences	178	12
Percentage of negative sentences	23.3%	12.8%
Negative average score	0.932533	0.921442

Table 15. Entropy Analysis of the Chunk Attention Weights Generated by FETILDA w/FinBERT from the 10-K Reports of Microtune Inc. in 2006 and Federal Screw Works in 2003, from the FIN10K Dataset

	Document length	Entropy of chunk attention weights	Testing loss (squared error)
Microtune Inc., 2006	24,172	3.094900	0.029381
Federal Screw Works, 2003	2,587	0.495800	0.441628

We further analyze the sentiment of all the sentences in both documents, as well as the document lengths and how that affects the entropy of the chunk attention weights produced in the document-level attention pooling stage. Tables 14 and 15 show the results of our analyses. Overall, the good performance document is about 10 times the length as the bad performance document, and contains more sentiment-bearing sentences both in terms of sentence count and percentage, which resulted in a big discrepancy in the entropies of their respective chunk attention weights. This means that the model focused on just select parts of the bad performance document that were mostly neutral, while focusing on each part of the good performance document more evenly. As a result, the testing loss was much higher in the shorter document than the longer one.

5 CONCLUSIONS

In this article, we examined the efficacy of various different types of language models using the FETILDA framework in generating effective document embeddings for very long financial text documents, such as 10-K public disclosures to the SEC, for which just one section, such as Item 7/7A, contains over 12,000 words on average. In our extensive set of experiments, we applied FETILDA with various different language models to the task of predicting eight different KPIs for U.S. Bank performance, as well as stock volatility prediction for U.S. companies from FIN10K. Our approach is shown to outperform previous baselines, yielding SOTA results on the various regression tasks for the two datasets used, thus testifying to the efficacy of language models in representing long financial documents. With the FIN10K dataset especially, we demonstrated quite evidently the significance of the improvement we get from taking a domain-specific LM such as FinBERT and fine-tuning it on our particular downstream task. We show this not only by how much FETILDA with fully unfrozen FinBERT outperforms the baseline methods but also by how fine-tuning FinBERT through unfreezing all its layers during training yields better performance than using the frozen pretrained embeddings that the LM produces. Our work also shows that using the whole document via chunk attention outperforms the standard approach that truncates the document to only one chunk, even over LMs that have large context windows.

Our work opens avenues for follow-on research. For example, while the contextual models in FETILDA can learn more effective document representations compared to baselines like TF-IDF, there is still scope for more improvement. One could consider learning even larger domain-specific pre-trained models for financial text, with larger blocks (e.g., using Longformer or Nystromformer instead of BERT for pre-training). We also plan to explore alternative approaches to learn better document representations. For example, instead of using the entire text, we can focus on the most important words, phrases, and sentences (e.g., sentiment bearing elements within the text). We can derive better chunk-level and document-level embeddings in this manner. How to select these informative elements from text remains an open challenge. Another promising avenue is to leverage domain-specific generative models in finance, such as FinGPT [45], and study how we can utilize generative approaches for textual regression tasks.

REFERENCES

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'20)*. Association for Computational Linguistics, 268–284. https://doi.org/10.18653/v1/2020.emnlp-main.19

[2] Amir Amel-Zadeh and Jonathan Faasse. 2016. The information content of 10-K narratives: comparing MD&A and footnotes disclosures. Retrieved Nov. 2, 2019 from https://dx.doi.org/10.2139/ssrn.2807546

[3] Dogu Araci. 2019. *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. Master's thesis. University of Amsterdam.

[4] Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. Retrieved from https://arXiv:2004.05150

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, 1877–1901. Retrieved from https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[6] Ching Yun Chang, Yue Zhang, Zhiyang Teng, Zahn Bozanic, and Bin Ke. 2016. Measuring the information content of financial news. In *Proceedings of the 26th International Conference on Computer Linguistics (COLING'16)*. 3216–3225.

[7] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*. Retrieved from https://openreview.net/pdf?id=r1xMH1BtvB

[8] Sanjiv Ranjan Das et al. 2014. Text and context: Language analytics in finance. *Found. Trends Finance* 8, 3 (Nov. 2014), 145–261.

[9] Vinicio Desola, Kevin Hanna, and Pri Nonis. 2019. FinBERT: Pre-trained model on SEC filings for financial natural language tasks. Retrieved from https://doi.org/10.13140/RG.2.2.19153.89442

[10] 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 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[11] Travis Dyer, Mark Lang, and Lorien Stice-Lawrence. 2017. The evolution of 10-K textual disclosure: Evidence from Latent Dirichlet Allocation. *J. Account. Econ.* 64, 2 (2017), 221–245. https://doi.org/10.1016/j.jacceco.2017.07.002

[12] Sophie Emerson, Ruairí Kennedy, Luke O'Shea, and John O'Brien. 2019. Trends and applications of machine learning in quantitative finance. In *Proceedings of the 8th International Conference on Economics and Finance Research (ICEFR'19)*.

[13] Jason Fernando. 2021. How return on equity (ROE) works. Retrieved from https://www.investopedia.com/terms/r/returnonequity.asp

[14] Hongyu Gong, Yelong Shen, Dian Yu, Jianshu Chen, and Dong Yu. 2020. Recurrent chunking mechanisms for long-text machine reading comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 6751–6761. https://doi.org/10.18653/v1/2020.acl-main.603

[15] Quentin Grail, Julien Perez, and Eric Gaussier. 2021. Globalizing BERT-based transformer architectures for long document summarization. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 1792–1810. https://doi.org/10.18653/v1/2021.eacl-main.154

[16] Allen H. Huang, Hui Wang, and Yi Yang. 2023. FinBERT: A large language model for extracting information from financial text. *Contemp. Account. Res.* 40, 2 (2023), 806–841. https://doi.org/10.1111/1911-3846.12832

[17] Dan Jurafsky and James H. Martin. 2021. *Speech and Language Processing* (3rd ed.). Retrieved from https://web.stanford.edu/~jurafsky/slp3/

[18] Will Kenton. 2021. What you should know About 10-KS. Retrieved from https://www.investopedia.com/terms/1/10-k.asp

[19] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. 2016. Predicting the direction of stock market prices using random forest. Retrieved from https://arXiv:1605.00003

[20] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. Retrieved from https://arXiv:2001.04451

[21] Shimon Kogan, Dimitry Levin, Bryan R. Routledge, Jacob S. Sagi, and Noah A. Smith. 2009. Predicting risk from financial reports with regression. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics*. 272–280.

[22] Yu-Wen Liu, Liang-Chih Liu, Chuan-Ju Wang, and Ming-Feng Tsai. 2016. FIN10K: A web-based information system for financial report analysis and visualization. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM'16)*. ACM, New York, NY, 24412444. https://doi.org/10.1145/2983323.2983328

[23] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. 2020. FinBERT: A pre-trained financial language representation model for financial text mining. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4513–4519. https://doi.org/10.24963/ijcai.2020/622

[24] Tim Loughran and Bill McDonald. 2011. When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *J. Finance* 66, 1 (Feb. 2011), 35–65.

[25] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR'13)*, Yoshua Bengio and Yann LeCun (Eds.).

[26] Vipal Monga and Emily Chasan. 2015. The 109,894-word annual report. Retrieved from https://www.wsj.com/articles/the-109-894-word-annual-report-1433203762

[27] Paraskevi Nousi, Avraam Tsantekidis, Nikolaos Passalis, Adamantios Ntakaris, Juho Kanniainen, Anastasios Tefas, Moncef Gabbouj, and Alexandros Iosifidis. 2019. Machine learning for forecasting mid-price movements using limit order book data. *IEEE Access* 7 (May 2019), 64722–64736.

[28] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1532–1543.

[29] Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training. *OpenAI Blog* (June 11, 2018).

[30] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog* (Feb. 14, 2019).

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67. http://jmlr.org/papers/v21/20-074.html

[32] Mustafa A. Sakarwala and Anthony Tanaydin. 2019. Use advances in data science and computing power to invest in stock market. *SMU Data Sci. Rev.* 2, 1 (2019), 17.

[33] Marcelo Sardelich and Suresh Manandhar. 2018. Multimodal deep learning for short-term stock volatility prediction. Retrieved from https://arXiv:1812.10479

[34] Marina Sedinkina, Nikolas Breitkopf, and Hinrich Schütze. 2019. Automatic domain adaptation outperforms manual domain adaptation for predicting financial outcomes. In *Proceedings of the 57th Annual Meeting of the Association for Computer Linguistics*. 346–359.

[35] Paul C. Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. 2008. More than words: Quantifying language to measure firms' fundamentals. *J. Finance* 63, 3 (June 2008), 1437–1467.

[36] Christoph Kilian Theil, Sanja Štajner, and Heiner Stuckenschmidt. 2018. Word embeddings-based uncertainty detection in financial disclosures. In *Proceedings of the 1st Workshop on Economics and Natural Language Processing*. 32–37.

[37] Christoph Kilian Theil, Sanja Štajner, and Heiner Stuckenschmidt. 2020. Explaining financial uncertainty through specialized word embeddings. *ACM Trans. Data Sci.* 1, 1 (Mar. 2020), 1–19.

[38] Ming-Feng Tsai and Chuan-Ju Wang. 2017. On the risk prediction and analysis of soft information in finance reports. *Eur. J. Oper. Res.* 257, 1 (Feb. 2017), 243–250.

[39] Ming-Feng Tsai, Chuan-Ju Wang, and Po-Chuan Chien. 2016. Discovering finance keywords via continuous-space language models. *ACM Trans. Manage. Inf. Syst.* 7, 3 (Aug. 2016), 1–17.
[40] Ruey S. Tsay. 2005. *Analysis of Financial Time Series*. Vol. 543. John Wiley & sons.
[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. 5998–6008.
[42] Yan Wang and Xuelei Sherry Ni. 2019. A XGBoost risk model via feature selection and Bayesian hyper-parameter optimization. Retrieved from https://arXiv:1901.08433
[43] Han Xiao. 2018. bert-as-service. Retrieved Mar. 3, 2020 from https://github.com/hanxiao/bert-as-service
[44] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. 2021. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
[45] Hongyang Yang, Xiao-Yang Liu, and Christina Dan Wang. 2023. FinGPT: Open-source financial large language models. In *Proceedings of the FinLLM Symposium at the International Joint Conference on Artificial Intelligence (IJCAI'23)*.
[46] Liu Yang, Mingyang Zhang, Cheng Li, Michael Bendersky, and Marc Najork. 2020. Beyond 512 tokens: Siamese multi-depth transformer-based hierarchical encoder for long-form document matching. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. ACM, New York, NY, 17251734. https://doi.org/10.1145/3340531.3411908
[47] Linyi Yang, Zheng Zhang, Su Xiong, Lirui Wei, James Ng, Lina Xu, and Ruihai Dong. 2018. Explainable text-driven neural network for stock prediction. In *Proceedings of the 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS'18)*. 441–445.
[48] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, 17283–17297. Retrieved from https://proceedings.neurips.cc/paper/2020/file/c8512d142a2d849725f31a9a7a361ab9-Paper.pdf
[49] Mohammed J. Zaki and Wagner Meira Jr. 2020. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2nd ed.). Cambridge University Press.

Received 1 November 2022; revised 8 November 2023; accepted 1 April 2024