

# FETILDA: A FRAMEWORK FOR FIN-TUNED EMBEDDINGS OF LONG FINANCIAL TEXT DOCUMENTS

**Bolun “Namir” Xia**

Submitted in Partial Fullfillment of the Requirements  
for the Degree of

*MASTER OF SCIENCE*

Approved by:

Dr. Mohammed J. Zaki, Chair

Dr. Sibel Adalı

Dr. Aparna Gupta

Dr. Tomek Strzalkowski



*Department of Computer Science*  
Rensselaer Polytechnic Institute  
Troy, New York

[May 2022]  
Submitted March 2022

# CONTENTS

LIST OF TABLES . . . . .	iii
LIST OF FIGURES . . . . .	iv
ACKNOWLEDGMENT . . . . .	v
ABSTRACT . . . . .	vi
1. Introduction . . . . .	1
2. Related Work . . . . .	5
2.1 Textual Data: Sentiment Analysis . . . . .	5
2.2 Language Models in Finance . . . . .	6
2.3 Long Document Language Models . . . . .	7
3. FETILDA: Methodology . . . . .	9
3.1 Chunk Generation . . . . .	9
3.2 Chunk-Level Language Model Pooling . . . . .	10
3.3 Document-Level Attention Pooling . . . . .	12
3.4 Model Training And Fine-Tuning . . . . .	13
3.5 FETILDA: Overall Summary . . . . .	14
4. FETILDA: Experiments . . . . .	16
4.1 Datasets and Target Metrics . . . . .	16
4.1.1 US Banks Dataset . . . . .	16
4.1.2 FIN10K Dataset [24] . . . . .	20
4.2 Methods . . . . .	22
4.3 Comparative Performance Results . . . . .	23
4.4 Algorithmic Choices . . . . .	31
5. Conclusion and Future Work . . . . .	34

## LIST OF TABLES

4.1	US bank dataset statistics. . . . .	19
4.2	FIN10K dataset [24] statistics. . . . .	20
4.3	MSE results on Item 7/7A with historical scores. Best results in bold. . . . .	24
4.4	The effect of freezing the last or all layers of the underlying LM in FETILDA for Item 7/7A. Best results in bold. . . . .	24
4.5	MSE results on Item 1A with historical scores. Best results in bold. . . . .	26
4.6	The effect of freezing the last or all layers of the underlying LM in FETILDA for Item 1A. Best results in bold. . . . .	26
4.7	Performance comparison (MSE) of FETILDA with BL (SVR) LOG1P+ [40], and TF-IDF. Best results in bold. . . . .	28
4.8	The effect of freezing the last or all layers of the underlying LM in FETILDA for the FIN10K [24] dataset. Best results in bold. . . . .	29
4.9	Text only: MSE results on Item 7/7A. Best results in bold. . . . .	30
4.10	Text only: MSE results on Item 1A. Best results in bold. . . . .	30
4.11	A comparison of three different models of FinBERT. . . . .	31
4.12	A comparison of different pooling methods. . . . .	32
4.13	Best regression model for FETILDA w/Longformer with last layer frozen. . . . .	32

## LIST OF FIGURES

3.1	Chunk generation . . . . .	9
3.2	Chunk-level language model pooling for chunk embeddings. . . . .	11
3.3	Attention pooling via Bi-LSTM for document embeddings. . . . .	12
3.4	Final linear neural network. . . . .	14
3.5	FETILDA: Overall framework. . . . .	15
4.1	The average document length (number of words) increases with time for both Item 1A and Item 7/7A. . . . .	20
4.2	The average document length (number of words) increases with time for Item 7 of 10-K reports in FIN10K [24]. . . . .	21
4.3	Improvements for FETILDA on Item 7/7A over TF-IDF, LOG1P, and linear regression. . . . .	25
4.4	Improvements for FETILDA on Item 1A over TF-IDF, LOG1P, and linear regression. . . . .	27
4.5	Best improvements of our approach compared with the LOG1P+ methods used in Tsai and Wang [2017]. . . . .	28

## ACKNOWLEDGMENT

I would like to thank my advisor, Prof. Mohammed J. Zaki for leading me and giving me directions in researching such an interesting and cutting-edge topic, and for reviewing and editing the journal paper that we have submitted for publication.

I would like to thank Prof. Aparna Gupta for participating in our weekly discussions and giving us valuable input, and for reviewing and editing the journal paper.

I would also like to thank Prof. Sibel Adalı and Prof. Tomek Strzałkowski for agreeing to be on my committee.

Lastly, I would like to acknowledge my fellow MS student, Vipula Rawte for laying the groundworks for us, in terms of preliminary research, so that we could innovate upon it and actually achieve SOTA results.

## ABSTRACT

Increasingly, unstructured data are being utilized in different domains. In particular, textual data, in recent years, is becoming more important. When it comes to financial applications, unstructured data, such as text from financial documents that companies publish on a consistent basis to government regulators like the Securities and Exchange Commission (SEC), is accumulating more and more. These financial documents are typically quite long, but they usually contain soft information that can be valuable in gauging company performance. They are special in that this soft information is not taken into consideration when trying to perform predictive analysis with only numerical data. Therefore, it would be very beneficial to train predictive models to learn on these long financial documents, in order to forecast metrics that gauge a company’s future performance. And indeed, much progress has been made in the sphere of Natural Language Processing (NLP) in pre-trained language models (LMs) that train on huge corpora of texts. However, that progress is still lacking when it comes to effectively representing long documents. This is the focus of this thesis: we are looking at how to learn better models to utilize the beneficial information contained in long financial text documents and generate more informative features from text, in order to use the soft information for various regression tasks. Towards that end, we propose and implement a novel machine learning framework that divides a long document into chunks, inputs the chunks through different LMs, both pre-trained and from scratch, use the outputs from those chunks to generate chunk-level vector representations, and inputs that representation into a self-attention bi-LSTM network to generate a document-level representation. In order to evaluate our deep learning framework, we experiment on one dataset of 10-K financial reports published annually by banks in the US, and another dataset of 10-K reports published by publicly traded companies in the US. Our experiment results show that our approach outperforms strong baseline approaches in terms of textual modeling and a baseline regression approach utilizing only quantitative data. Our work shows that using pre-trained, domain-specific, and fine-tuned LMs in representing long texts betters the quality of the textual features generated, and improves the performance of prediction tasks.

# CHAPTER 1

## Introduction

Unstructured data such as text is growing very fast in different domains. And especially, textual data from financial documents have 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 [20], 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 [28] and GloVe [30], 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 [31] and BERT [10] are built on top of the very effective Transformer-based attention model [43], 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.

While much progress has been made in NLP, our focus in this thesis is on the relatively

---

This chapter has been submitted to: Bolun “Namir” Xia, Vipula D. Rawte, Aparna Gupta, and Mohammed J. Zaki. 2022. FETILDA: An Effective Framework For Fin-tuned Embeddings For Long Financial Text Documents. In DSAA’2022: IEEE International Conference on Data Science and Advanced Analytics: MLJ special issue on Foundations of Data Science, 2022. DSAA, Shenzhen, China.

under-examined area of financial text documents. Specifically, we are focusing on the 10-K financial reports that companies in the US submit annually to the Securities and Exchange Commission (SEC). These reports describe a company’s activities, progress, and risks in the interest of the company’s stakeholders. The detailed content of these reports is helpful in evaluating the status of the company as well as predicting future metrics from forward-looking statements. One of the key parts of 10-K reports is Item 7, namely, the Management’s Discussion and Analysis (MDA) section. Another key part is Item 1A, namely, the Risk Factors section. Textual data contained in these 10-K reports are predictive of the volatilities that companies have in the stock market, and can be helpful in predicting failures of financial institutions, as well [8, 27]. But, in these approaches, they use an expert-generated sentiment dictionary by Loughran and McDonald (L&M) [2011] to generate the vector of features. For example, Kogan et al. [2009] utilized the MDA section to predict the return volatility of stocks using the L&M word list. In addition, the L&M word list was expanded using word2vec [28]. Then, it was used in Tsai and Wang [2017].

As mentioned above, moving beyond the static embedding methods, it is important that we take into consideration the contextual information of words, since specific words can have different meanings in the financial context than those in the general context, the nuances of which may not be apparent to laypersons. For this, we can utilize the recent contextual language models to represent long documents, such as the 10-K reports, in order to construct effective models for conducting predictive analyses. 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 [49], reaching up to 4,096 tokens. 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, wiki media, 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 in order 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. [2020], Yang et al. [2020], Araci [2019], and DeSola et al. [2019], 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 a novel framework called **Fin-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 novel chunk-based deep learning framework, where a long document is split into several smaller chunks and then each chunk is processed using an appropriate language model (e.g., BERT [10], FinBERT [48] or Longformer [4]). 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 Bi-LSTM model equipped with self-attention mechanism. The pooled chunks are then aggregated into a document level representation, which serves as input for a fully connected linear neural network for target variable prediction. We can also use the document representations as inputs to Support Vector and Kernel regression models. We evaluate our framework using two different corpuses: i) 10-K reports submitted annually to the SEC by US banks for the period from 2006 to 2016, ii) 10-K reports for all US companies from 1996 to 2013 [24]. We have conducted extensive experiments using these datasets and applied

our framework to different predictive analysis regression tasks: i) predictive metrics of a bank’s financial performance, such as 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); ii) analysis of a company’s stock market volatility. Our results compared against the different baseline methods show that FETILDA performs significantly better and yields SOTA results for long financial text regression tasks. In summary, our main contributions are:

- We propose a SOTA deep learning framework 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 showcase the effectiveness of our FETILDA approach. We applied the model on two different 10-K datasets, and on 9 different regression tasks. We show that FETILDA outperforms several different baseline methods, and achieves state-of-the-art results on long financial documents.

## CHAPTER 2

### Related Work

Machine learning plays an important role in financial analytics. And in financial analytics, the goal is to perform stock return forecasting, risk modeling, and KPI forecasting that utilizes chiefly quantitative or numeric data [11]. Various ML models such as regressions and neural networks were utilized to forecast price movements in Nousi et al. [2019]. They used both handcrafted features generated from raw order book data and features generated from ML algorithms. Other models, such as Random Forest [21], Bidirectional Long Short-Term Memory (Bi-LSTM), stacked LSTMs [35], and XGBoost [44], were also built to forecast stock return volatility and business risk. 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. In Tetlock et al. [2008], they generated textual features using the negative words in a hand-crafted dictionary and compiled them into a document-level term matrix from the source data. Then, they used these features to forecast stock returns and future earnings. In Chang et al. [2016], they proposed a tree-like LSTM to access the value of financial news using both text and numerical stock returns. In Yang et al. [2018], they designed an attention-based neural network to forecast price movement utilizing textual data from financial news. Assessing whether the textual data from financial news is useful in predicting quantitative metrics is valuable because it can positively affect the investment strategy of both individual and institutional investors.

Alternatively, textual features can also be constructed using pre-trained language models. However, the general domain LMs usually are not good at extracting effective predictive

---

This chapter has been submitted to: Bolun “Namir” Xia, Vipula D. Rawte, Aparna Gupta, and Mohammed J. Zaki. 2022. FETILDA: An Effective Framework For Fin-tuned Embeddings For Long Financial Text Documents. In DSAA’2022: IEEE International Conference on Data Science and Advanced Analytics: MLJ special issue on Foundations of Data Science, 2022. DSAA, Shenzhen, China.

features in the financial domain. Araci [2019] proposed the FinBERT model, which can be fine-tuned on the financial data from Reuters, and it was shown to outperform the general BERT model. Using other than financial news data, in Kogan et al. [2009], the authors generated their textual features from 10-K reports. Then, these features were utilized to forecast stock return volatilities. And in Sakarwala and Tanaydin [2019], they trained a deep learning model on SEC filings, which improved the prediction of stock returns, compared to traditional ML models.

In Tsai et al. [2016] and Tsai and Wang [2017], they generated extra textual features by expanding the L&M sentiment word list [26] with word2vec [28]. Similarly, in Theil et al. [2018], they also used word2vec to expand out the uncertainty word list in L&M dictionary, for the purpose of forecasting stock return volatilities. In Theil et al. [2020], they expanded the L&M dictionary through industry-specific word embedding models, utilizing word2vec to forecast stock return volatility, analyst forecast error and analyst dispersion. In Sedinkina [2019], the authors demonstrated how adapting the L&M sentiment list using word2vec to a specific domain can improve the prediction of stock return and stock return volatility. In the expansion of the L&M dictionary mentioned above, they all utilized word2vec model to pick the top  $n$  nearest words to the words in the dictionary. However, since word2vec only generates static word embeddings, all these approaches cannot capture the dynamic contextual information contained in those texts.

## 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. [2020], Yang et al. [2020], Araci [2019], and DeSola et al. [2019], 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 [2019] 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. [2020] focused on financial news and dialogues present on websites, and collected three financial corpora: 13 million financial news (15GB) and financial articles (9GB) from Financial Web, totaling 24GB and 6.38 billion words; financial articles from Yahoo! Finance, totaling 19GB and 4.71 billion words; and question-answer pairs about financial issues from Reddit, totaling 5GB 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 [25]. Yang et al. [2020] 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 [48]. They report that their model outperforms BERT [10] in three sentiment analysis tasks, all by significant margins [48]. DeSola et al. [2019] 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 [31], GPT-2 [32], GPT-3 [5], T-5 [33], 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], ETC [1], and BigBird [49], have been proposed, all of which innovate on the self-attention mechanism in order 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 [22] and Nyströmformer [46], 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.

Extended Transformer Construction (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 [49] 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.

# CHAPTER 3

## FETILDA: Methodology

FETILDA first splits a long document into different chunks, then processes each chunk using a language model, all of whose layers can be 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. Finally, we input the document embedding into a fully connected linear neural network followed by a linear, Support Vector, or kernel regression. Overall, 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

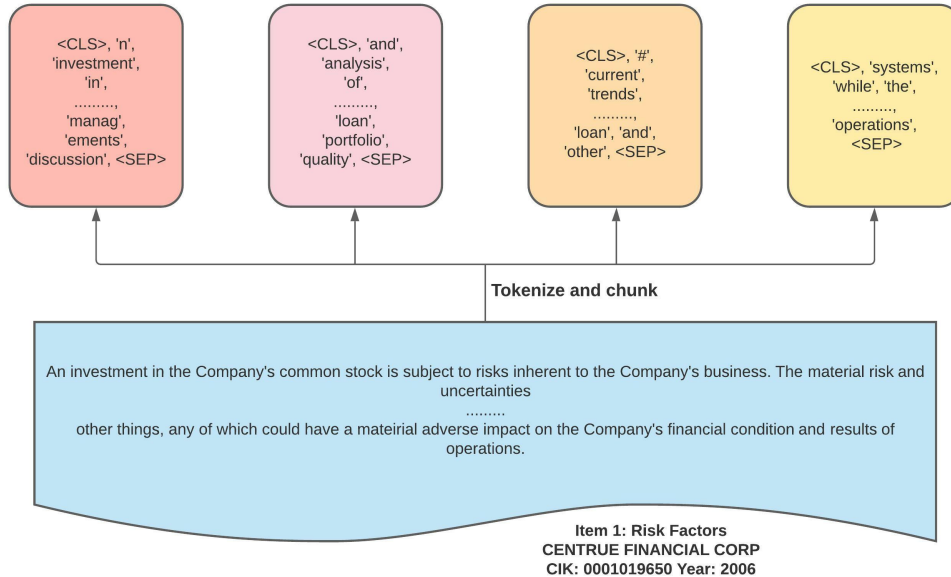


Figure 3.1: Chunk generation

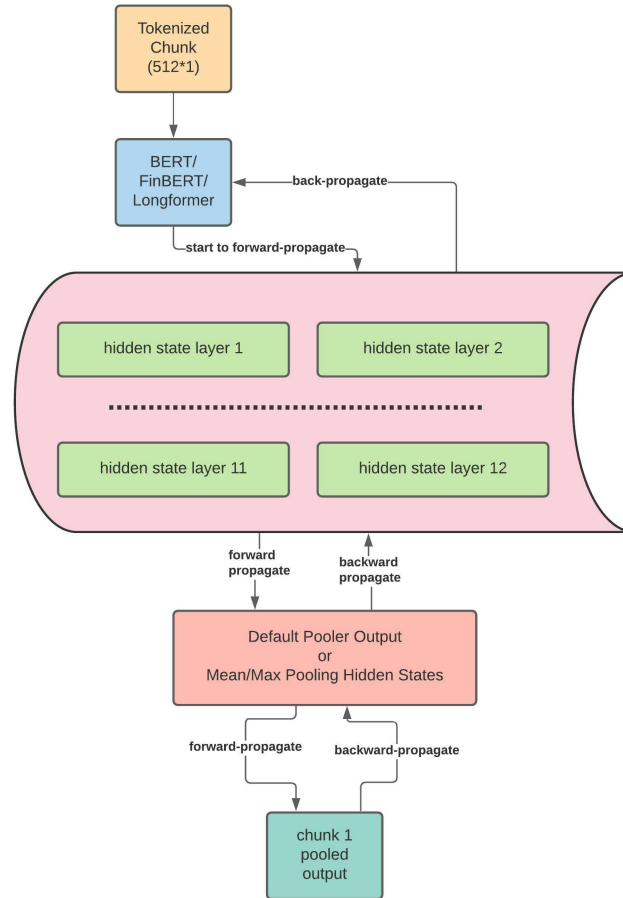
This chapter has been submitted to: Bolun “Namir” Xia, Vipula D. Rawte, Aparna Gupta, and Mohammed J. Zaki. 2022. FETILDA: An Effective Framework For Fin-tuned Embeddings For Long Financial Text Documents. In DSAA’2022: IEEE International Conference on Data Science and Advanced Analytics: MLJ special issue on Foundations of Data Science, 2022. DSAA, Shenzhen, China.

Let  $L = \{d_1, d_2, \dots, d_N\}$  denote a text corpus containing  $N$  long documents, where  $d_i$  denotes the  $i$ -th document in the corpus. We tokenize each document  $d_i$  into a sequence of tokens  $\{t_1, t_2, \dots, t_{n_i}\}$ , 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  $\{c_1, c_2, \dots, c_{m_i}\}$ , with  $m_i$  chunks of length  $b$ . We also prepend and append  $\langle\text{CLS}\rangle$  and  $\langle\text{SEP}\rangle$  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  $c_i = \{t_0, t_1, \dots, t_b, t_{b+1}\}$ , with  $t_0 = \langle\text{CLS}\rangle$  and  $t_{b+1} = \langle\text{SEP}\rangle$ . We experiment with both  $b + 2 = 512$  and  $b + 2 = 4096$ , 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 ( $\langle\text{PAD}\rangle$ )  $(b - k)$  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  $\langle\text{PAD}\rangle$  tokens. Figure 3.1 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,  $\{c_1, c_2, \dots, c_{m_i}\}$ , we need to convert these into features vectors  $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{m_i}\}$ , that represent the token sequence in each respective chunk as a whole. We use SOTA language models like BERT [10], Longformer [4], and FinBERT [48] to generate contextual token and chunk embeddings. We thus input each chunk into the underlying language model, which typically outputs 12 hidden state layers  $\{l_1, l_2, \dots, l_{12}\}$ , where  $l_i$  denotes layer  $i$ . The output of each of these layers contains  $b + 2$  hidden state vectors  $\{\mathbf{z}_0^l, \mathbf{z}_1^l, \dots, \mathbf{z}_{b+1}^l\}$ , 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  $\langle\text{CLS}\rangle$  token, the first token, of the last hidden state layer after processing and activation, denoted by  $\mathbf{z}_0^{12}$ . Figure 3.2 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 mean-



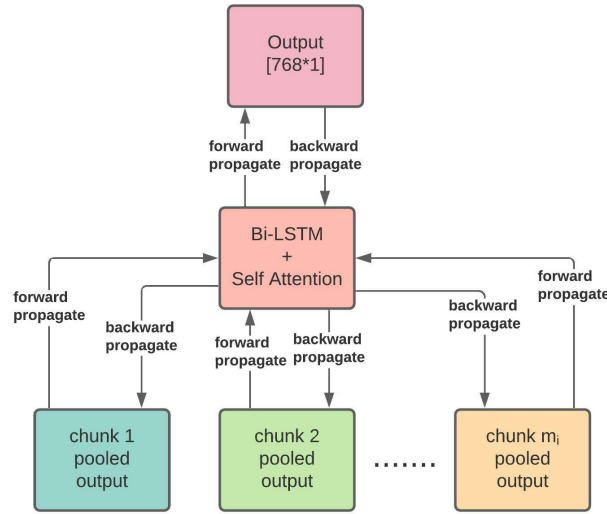
**Figure 3.2: Chunk-level language model pooling for chunk embeddings.**

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

- *Default pooler output:* Since the  $\langle \text{CLS} \rangle$  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 [45] 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,  $\{\mathbf{z}_0^{11}, \mathbf{z}_1^{11}, \dots, \mathbf{z}_{b+1}^{11}\}$  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  $\{\mathbf{z}_0^9, \mathbf{z}_1^9, \dots, \mathbf{z}_{b+1}^9\}$ ,  $\{\mathbf{z}_0^{10}, \mathbf{z}_1^{10}, \dots, \mathbf{z}_{b+1}^{10}\}$ ,  $\{\mathbf{z}_0^{11}, \mathbf{z}_1^{11}, \dots, \mathbf{z}_{b+1}^{11}\}$ , and  $\{\mathbf{z}_0^{12}, \mathbf{z}_1^{12}, \dots, \mathbf{z}_{b+1}^{12}\}$ , 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



**Figure 3.3: Attention pooling via Bi-LSTM for document embeddings.**

Given the chunk embedding vectors  $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{m_i}\}$ , 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 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  $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{m_i}\}$  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,

$$\mathbf{o}_i = \vec{\mathbf{o}}_i \oplus \overleftarrow{\mathbf{o}}_i \mathbf{h}_i = \vec{\mathbf{h}}_i \oplus \overleftarrow{\mathbf{h}}_i \quad (3.1)$$

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  $i$ -th element of the chunk sequence). The attention score  $\alpha_i$  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,

$$\alpha_i = \text{softmax}\left(\{\mathbf{o}_1^T \mathbf{h}_i, \mathbf{o}_2^T \mathbf{h}_i, \dots, \mathbf{o}_{m_i}^T \mathbf{h}_i\}\right) \quad (3.2)$$

$$\mathbf{d}_i = \frac{\sum_{j=1}^{m_i} \alpha_j \cdot \mathbf{c}_j}{m_i} \quad (3.3)$$

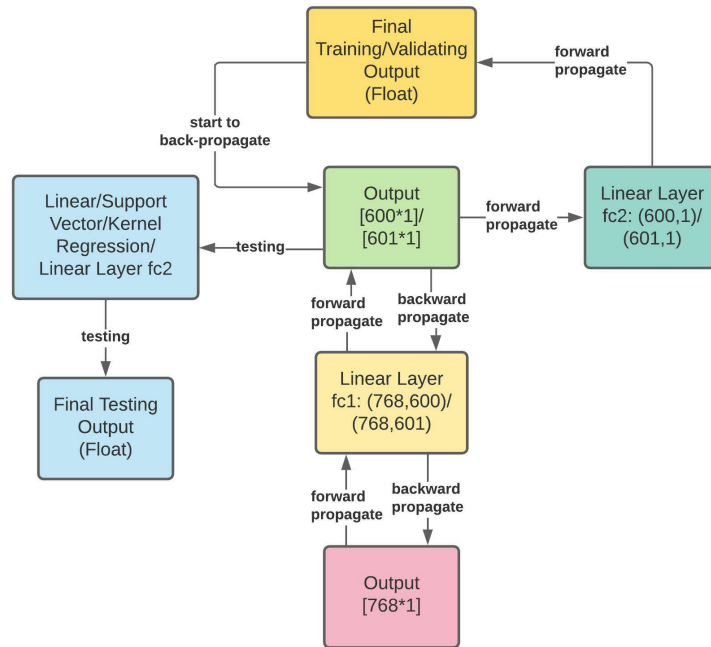
Figure 3.3 shows an illustration of the document level attention pooling step. At the bottom are the chunk embedding vectors  $\mathbf{c}_i$  as inputs, which are passed to the Bi-LSTM and attention modules to create the document embedding  $\mathbf{d}_i$ .

### 3.4 Model Training And Fine-Tuning

In the final stage of training, if we experiment only with textual features, we feed each 768-dimensional document feature vector  $\mathbf{d}_i$  to two additional fully connected linear layers  $f_{c_1}$  and  $f_{c_2}$ , with size 600 and 1, respectively (600 was chosen empirically), with a leaky ReLU activation and a dropout layer applied to  $f_{c_1}$ . The last layer  $f_{c_2}$  represents the output neuron to predict a target numeric variable. In addition to purely textual features, we can also input numeric features at this stage. In particular, if we experiment with previous target data, 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  $\mathbf{d}_i$  so as to use both the numerical and textual features. Formally,

$$\mathbf{d}_i = \mathbf{o}_{f_{c_1}} \oplus y^{hist} \quad (3.4)$$

where  $\mathbf{o}_{f_{c_1}}$  denotes the output features vector from  $f_{c_1}$ . In this case  $f_{c_1}$  has 601 neurons, which are input to  $f_{c_2}$  to predict the target numeric score  $\hat{y}$ . The loss function is MSE or mean squared error between the predicted and true target value.

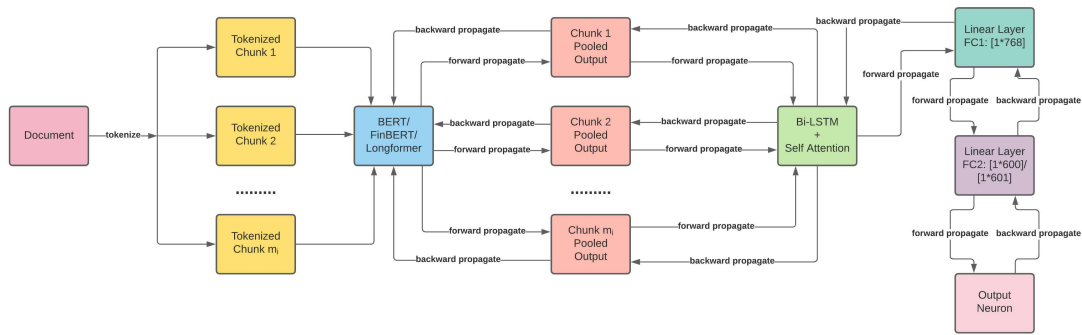


**Figure 3.4: Final linear neural network.**

In the validation step, we take the output features vector from  $\mathbf{o}_{fc_1}$  obtained from  $f_{c_1}$  (with or without concatenating  $y^{hist}$ ), and feed it to one of four final layers to obtain the final output value of the model, depending on which one achieves the best validation loss. These four layers are: (1) the predicted output from the  $f_{c_2}$  output layer, or feeding  $\mathbf{o}_{fc_1}$  into a separate (2) Linear Regression layer, (3) a Support Vector Regression layer, or (4) a Kernel Ridge Regression layer. The testing step uses the best validation model to make the final target variable prediction for each document. Figure 3.4 shows the details of training, validation and testing steps. The document level embedding vector  $\mathbf{d}_i$  is fed into the first linear layer  $f_{c_1}$ , whose output is combined with the historical numeric value (when needed), and then fed into the second linear layer  $f_{c_2}$  to predict the final output. MSE loss is used to train the model. The figure also shows the testing/validation steps involving the four different variants for the regression.

### 3.5 FETILDA: Overall Summary

Figure 3.5 shows the complete FETILDA framework combining all the above steps. We take a very long document and divide it into small fragments or chunks. The chunk representations are extracted from the underlying language model (BERT, FinBERT, or



**Figure 3.5: FETILDA: Overall framework.**

Longformer) 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, or only the last layer can be frozen), using MSE as the loss function.

## CHAPTER 4

### FETILDA: 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 2.5Ghz Intel Xeon Gold 6248 CPU, 768GB memory, and a NVIDIA Tesla V100 GPU with 32GB memory. The neural network models are implemented using PyTorch v1.10 (pytorch.org) and the HuggingFace library (huggingface.co) (for the BERT and Longformer language models). Our code and datasets are publicly available on github via <https://github.com/Namir0806/FETILDA>.

#### 4.1 Datasets and Target Metrics

##### 4.1.1 US Banks Dataset

We collected the 10-K filings for all US banks for the period between 2006 and 2016 (from the SEC EDGAR website: [www.sec.gov/edgar](http://www.sec.gov/edgar)), as well as the corresponding quantitative target data from the WRDS Center for Research in Security Prices. For the US banks dataset, 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: Return on Assets (ROA), Earnings per Share (EPS), Return on Equity (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:

$$ROA = \frac{Net\ Income}{Total\ Assets} \quad (4.1)$$

According to Hargrave [2022], the ROA indicates how profitable a company is in relation to its total assets. Corporate management, analysts, and investors can use ROA

---

Portions of this chapter have been submitted to: Bolun “Namir” Xia, Vipula D. Rawte, Aparna Gupta, and Mohammed J. Zaki. 2022. FETILDA: An Effective Framework For Fin-tuned Embeddings For Long Financial Text Documents. In DSAA’2022: IEEE International Conference on Data Science and Advanced Analytics: MLJ special issue on Foundations of Data Science, 2022. DSAA, Shenzhen, China.

to determine how efficiently a company uses its assets to generate a profit. A higher ROA means a company is more efficient and productive at managing its balance sheet to generate profits while a lower ROA indicates there is room for improvement.

- **Return on Equity (ROE):** According to Fernando [2021], ROE is a measure of financial performance calculated by dividing net income by shareholders' equity:

$$ROE = \frac{Net\ Income}{Total\ Equity} \quad (4.2)$$

Since equity is simply the assets of a company minus the debt, ROE is basically the return on investment to the shareholders of a company. ROE is an indicator of a company's profitability and efficiency in generating its profits.

- **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:

$$EPS = \frac{Net\ Income - Preferred\ Dividends}{End-of-Period\ Common\ Shares\ Outstanding} \quad (4.3)$$

The higher a company's EPS, the more profitable it is considered to be [13].

- **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:

$$Tobin's\ Q\ Ratio = \frac{Equity\ Market\ Value + Liabilities\ Book\ Value}{Equity\ Book\ Value + Liabilities\ Book\ Value} \quad (4.4)$$

According to Hayes [2022], at its most basic level, the Tobin's Q Ratio expresses the relationship between market valuation and intrinsic value. In other words, it is a means of estimating whether a given business or market is overvalued or undervalued.

- **Leverage Ratio (LR):** The leverage ratio measures the extent of debt financing used by a firm:

$$Leverage\ Ratio = \frac{Total\ Liabilities}{Total\ Equity} \quad (4.5)$$

According to Hayes [2021], a high leverage ratio generally indicates that a company has been aggressive in financing its growth with debt. This can result in volatile earnings as a result of the additional interest expense. If the company's interest expense

grows too high, it may increase the company’s chances of a default or bankruptcy. Banks are among the most leveraged institutions in the United States. The combination of fractional-reserve banking and Federal Deposit Insurance Corporation (FDIC) protection has produced a banking environment with limited lending risks. The level of scrutiny paid to leverage ratios has increased since the Great Recession of 2007 to 2009 when banks that were ”too big to fail” were a calling card to make banks more solvent.

- **Tier 1 Capital Ratio (T1CR):** The tier 1 capital ratio is a key measure of a bank’s financial strength that has been adopted as part of the Basel III Accord on bank regulation [18]. It is the ratio of a bank’s core tier 1 capital to its total risk-weighted assets:

$$\textit{Tier 1 Capital Ratio} = \frac{\textit{Tier 1 Capital}}{\textit{Total Risk - Weighted Assets}} \quad (4.6)$$

According to Hayes [2022], the tier 1 capital ratio measures a bank’s core equity capital against its total risk-weighted assets—which include all the assets the bank holds that are systematically weighted for credit risk. For example, a bank’s cash on hand and government securities would receive a weighting of 0%, while its mortgage loans would be assigned a 50% weighting.

- **Z-score (Z):** According to FRED [2019], the Z-score captures the probability of default of a country’s banking system, calculated as a weighted average of the z-scores of a country’s individual banks (the weights are based on the individual banks’ total assets). Z-score compares a bank’s buffers (capitalization and returns) with the volatility of those returns. It is estimated as:

$$\textit{Z-Score} = \frac{\textit{ROA} + \textit{CAR}}{\sigma(\textit{ROA})} \quad (4.7)$$

where,  $\sigma(\textit{ROA})$  is the standard deviation of  $\textit{ROA}$  for a specific time period.

- **Market-to-Book Ratio (MBR):** The MBR 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 (i.e., the price that the market believes the company is

worth) by the book value [19]:

$$\text{Market-to-Book Ratio} = \frac{\text{Market Capitalization}}{\text{Total Book Value}} \quad (4.8)$$

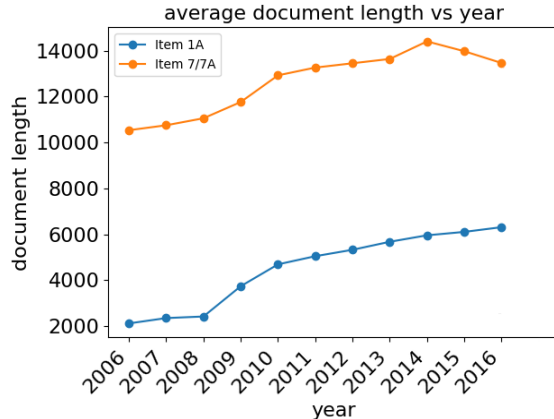
According to Corporate Finance Institute [2022], the market-to-book ratio shows the financial valuation of a company’s stock, and is an indicator of how much equity investors value each share relative to their book value.

While the entire 10-K report is a very long disclosure document, in particular Items 1A and 7/7A are considered as important sections in a 10-K report [2]. Item 1A is titled, “Risk Factors”, and 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. Item 7 is titled, “Management’s Discussion and Analysis of Financial Condition and Results of Operations” (MD&A), and it gives the company’s perspective on the business results of the past financial year. The MD&A section is meant for the management to relate in its own words the analysis of their financial condition. Finally, Item 7A is titled “Quantitative and Qualitative Disclosures about Market Risk” and 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. These sections are themselves also quite long. The dataset statistics for the 10-K reports for all US Banks for the period of 2006-2016 are reported in Table 4.1.

**Table 4.1: US bank dataset statistics.**

	Item 1A	Item 7/7A
number of total documents	5321	
after extracting items	3396	
target data available	2479	2500
average document length	4435.69	12589.75

The 10-K reports for US Banks (2006-2016) total 5321 documents, but not all reports have both the Item 1A or 7/7A sections. Out of the total, only 3396 10-K reports have both these important sections. Furthermore, we found that not all banks have all the eight target KPI values that we need for regression. Out of the 3396 documents, we have 2479



**Figure 4.1:** The average document length (number of words) increases with time for both Item 1A and Item 7/7A.

Item 1A and 2500 Item 7/7A 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 4436 for Item 1A and 12590 for Item 7/7A, as noted in Table 4.1. Furthermore, Figure 4.1 shows how the average document length increases in time. 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 and every one of the eight target metrics, we performed min-max scaling to normalize the data for training.

#### 4.1.2 FIN10K Dataset [24]

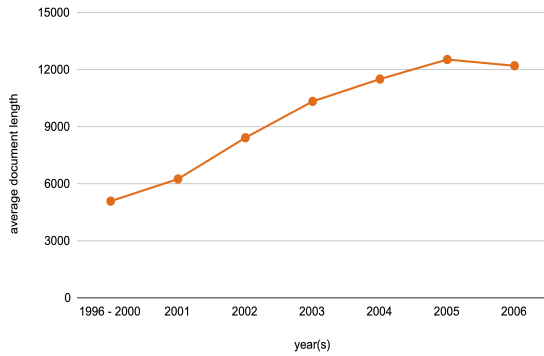
In addition to our dataset of 10-K reports filed by US banks, we also compare our FETILDA approach on the regression task outlined in Tsai and Wang [2017] using the dataset in the FIN10K project [24], which contains Item 7 of 10-K reports of US companies from 1996 to 2013 and the stock return volatilities twelve months before and after each report. Table 4.2 shows the statistics for the part of the FIN10K dataset [24] used in our comparative experiments against the LOG1P+ approach taken in Tsai and Wang [2017].

**Table 4.2:** FIN10K dataset [24] statistics.

	1996 - 2000	2001	2002	2003	2004	2005	2006
number of total documents	8703	1825	2023	2866	2861	2698	2564
average document length	5079.4	6245.6	8414.3	10324.7	11499.6	12528.1	12198.1

To replicate the regression experiment, we also 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. In addition, we did not perform any target data normalization, in order to replicate the experiment completely. Further, we choose the first 80% of reports from 1996 to 2000 as training data, and the remaining 20% as validation data. As we can see, the number of documents in the training and validation data from 1996 to 2000 is more than three times as many as that of the US banks dataset, but the average document length is significantly smaller than that of Item 7/7A in the US banks dataset. In the testing data, from 2001 to 2006, the number of documents is generally increasing, as well as the average document length, as shown in Figure 4.2.



**Figure 4.2: The average document length (number of words) increases with time for Item 7 of 10-K reports in FIN10K [24].**

In terms of the regression task, Tsai and Wang [2017] experimented on stock return volatilities twelve months before and after each report, and we use the same target values in our experiment. According to Tsai and Wang [2017], volatility 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  $R_t = \frac{S_t}{S_{t-1}} - 1$  [42]. Therefore, the volatility of returns for a stock from time  $t - n$  to  $t$  is defined as

$$v_{[t-n,t]} = \sqrt{\frac{\sum_{i=t-n}^t (R_i - \bar{R})^2}{n}} \quad (4.9)$$

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

## 4.2 Methods

We now outline the results of our framework on both the US banks and FIN10K datasets. Overall, we experiment with seven different methods, the first three being baseline methods with which we compare the last four to evaluate the performance of our approach. In order to effectively compare different methods, all results report the mean squared error (MSE). The methods are as follows:

- **TF-IDF** [20]: In this baseline method, we use the term frequency - inverse document frequency features, which help in scoring important words, either concatenated with the historical score  $y^{hist}$  or not, and then apply regression on the features to predict the target values. There are three regression methods we experiment with: (1) Linear Regression, (2) Support Vector Regression and (3) Kernel Ridge Regression. We use validation data to choose the best regression method among the three, according to which one achieves the lowest validation loss.
- **Linear Regression** [50]: In this baseline method, we simply take the historical score  $y^{hist}$  and run (bivariate) linear regression on it to predict the target variable. This method therefore only utilizes numerical data.
- **LOG1P+** [40]: This is the method used in the volatility regression task proposed by Tsai and Wang [2017]. 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 twelve 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 w/ BERT**: In this method, we use our approach, detailed in section 3, with plain BERT[10] as the underlying language model, setting the chunk size to 512 tokens and using the default pooling method.
- **FETILDA w/ FinBERT**: Here we used our approach with FinBERT [48] as the LM, which was pre-trained on 10-K, 10-Q, and analyst reports, setting the chunk size to 512 tokens and using the default pooling method.
- **FETILDA w/ Longformer**: Now, 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 4096 tokens and using the default pooling method.

- **FETILDA w/ Nyströmformer [46]**: Finally, to test the effectiveness of a even bigger block size, but without a pretrained model (that is, training from scratch), we use our approach with Nyströmformer [46] as the underlying language model, setting the chunk size to 8192 tokens, the number of layers to one, the number of attention heads to eight, and using the default pooling method.

With all four versions of FETILDA, namely using BERT, FinBERT, Longformer, and Nyströmformer, we performed an extensive set of experiments, evaluating our approach in predicting all eight different KPI metrics for the US banks dataset, and stock return volatility for the FIN10K dataset [24]. These experiments fall into two main categories: those using both textual data and numeric data, and those using only textual data. In the former category, we add historical scores as a feature to the textual features in predicting future metrics. For the US banks dataset, the historical scores are numeric values of each of the eight metrics in the previous year of the report, and for the FIN10K dataset, they are the stock return volatilities twelve months before each report. In the latter category of our experiments, we use only textual features to predict future metrics, with the aim to assess how much textual data alone is predictive of the quantitative target variable, without using historical values. In this category, we only conducted experiments on the US banks dataset and not on the FIN10K dataset [24], since the LOG1P+ model in Tsai and Wang [2017] includes the historical stock return volatilities as additional features, and our aim was to compare directly with their approach. In addition to applying our approach as described in section 3 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 the US banks dataset, with both Item 1A and Item 7/7A.

### 4.3 Comparative Performance Results

In all four versions of FETILDA, we train the model with eight varying learning rates from 0.0006 to 0.0013, and four different final layer options detailed in Section 3, 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 then use the Yang et al. [2020] FinBERT 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/Nyströmformer) and TF-IDF/LOG1P+ 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  $\gamma = 0.1$ . For FETILDA, we also include the variant based on the predicted output (from  $f_{c_2}$ ) with MSE loss.

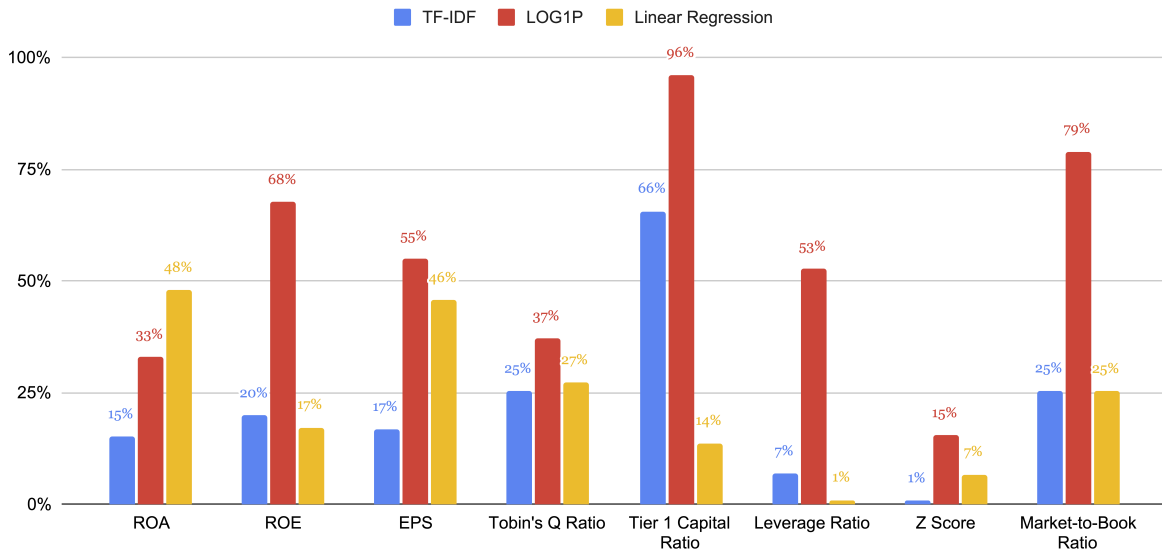
**Table 4.3: MSE results on Item 7/7A with historical scores. Best results in bold.**

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
LOG1P+	0.001112	0.025760	0.001887	0.026116	0.006582	0.005101	0.033905	0.020450
Linear Regression	0.001432	0.010096	0.001564	0.022587	0.000306	0.002441	0.030760	0.005757
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)	<b>0.000746</b>	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	<b>0.028697</b>	0.005683
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
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	<b>0.008356</b>	<b>0.000851</b>	<b>0.016436</b>	0.000291	0.002419	0.029011	0.005481
FETILDA w/Nyströmformer	0.000815	0.008989	0.000869	0.017554	<b>0.000264</b>	<b>0.002417</b>	0.030462	<b>0.004302</b>

**Table 4.4: The effect of freezing the last or all layers of the underlying LM in FETILDA for Item 7/7A. Best results in bold.**

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
FETILDA w/BERT (Fully Unfrozen)	<b>0.000796</b>	<b>0.009227</b>	<b>0.000897</b>	<b>0.021409</b>	0.000325	0.002502	0.029505	<b>0.005651</b>
FETILDA w/BERT (Fully Frozen)	0.000890	0.010052	0.001109	0.022748	0.000328	0.002581	<b>0.028966</b>	0.005950
FETILDA w/BERT (Last Layer Frozen)	0.000850	0.009903	0.000960	0.021728	<b>0.000306</b>	<b>0.002469</b>	0.029203	0.005798
FETILDA w/FinBERT (Fully Unfrozen)	<b>0.000746</b>	0.008901	<b>0.000932</b>	<b>0.019150</b>	0.000317	0.002535	<b>0.029516</b>	<b>0.005657</b>
FETILDA w/FinBERT (Fully Frozen)	0.001093	0.009401	0.001906	0.021882	0.000447	0.002514	0.030094	0.005695
FETILDA w/FinBERT (Last Layer Frozen)	0.000844	<b>0.008543</b>	0.000988	0.021425	<b>0.000304</b>	<b>0.002445</b>	0.029637	0.005678
FETILDA w/Longformer (Fully Unfrozen)	0.000813	0.008507	0.000858	0.017358	0.000296	0.002467	<b>0.028697</b>	0.005683
FETILDA w/Longformer (Fully Frozen)	<b>0.000801</b>	0.008501	0.000876	0.019053	0.000308	0.002436	0.028965	0.005957
FETILDA w/Longformer (Last Layer Frozen)	0.000849	<b>0.008356</b>	<b>0.000851</b>	<b>0.016436</b>	<b>0.000291</b>	<b>0.002419</b>	0.029011	<b>0.005481</b>

**Item 7/7A – Textual Features with Numeric History:** First, we report results using both textual and numeric data, i.e., we combine the document embeddings with the historic



**Figure 4.3: Improvements for FETILDA on Item 7/7A over TF-IDF, LOG1P, and linear regression.**

target value. Table 4.3 shows the performance comparison between the four versions of our approach on Item 7/7A and baseline methods: TF-IDF and LOG1P+ for textual modeling (with historic scores) and linear regression for numerical modeling. We can see that TF-IDF features always do better than LOG1P+, and for all metrics, our method outperforms the baseline methods (TF-IDF, LOG1P+ and linear regression), with FETILDA w/Longformer [4] performing the best in a majority of cases and FETILDA w/Nyströmformer coming as a close second. In addition, we also see a significant edge in the performance of FinBERT in the prediction of ROA target values. Overall, FETILDA w/BERT performs best or second best on three metrics, FETILDA w/FinBERT on four metrics, and FETILDA w/Longformer on six out of the eight metrics.

Table 4.4 shows the effect of freezing either the last layer or all layers for all three underlying LMs. As we can see, the token embeddings from Longformer with the last layer frozen perform the best, and even outperform the fine-tuned Longformer model (with unfrozen layers) on six out of the eight metrics, but fully freezing Longformer is shown not to be beneficial. On the other hand, BERT benefits from fine-tuning for five out of the eight metrics. This may be due to the much longer context blocks used in Longformer, which is able to capture more contextual information and does not need too much fine-tuning on the downstream tasks. FETILDA w/FinBERT benefits from fine-tuning in all of the eight metrics, either with all layers unfrozen or freezing only the last layer, which shows the

effectiveness of taking a domain-specific pretrained language model and then fine-tuning it for a particular downstream task. Overall, FETILDA w/Longformer offers the best or close to best results in predicting seven out of the eight KPI targets for US Banks.

To characterize the improvement due to FETILDA with different language models, for every metric, we select the best performing version of FETILDA in terms of mean squared error and calculate its improvement against the text and numeric baseline methods using the formula:  $(MSE_{baseline} - MSE_{FETILDA})/MSE_{baseline}$ . The results are shown in Figure 4.3. We observe large improvements for ROE, EPS, Leverage Ratio, and Tobin’s Q Ratio, using our approach over TF-IDF. For Tier 1 Capital ratio, our approach achieves the highest improvement over TF-IDF in percentage terms. Overall, our method outperforms TF-IDF, LOG1P+, and linear regression for all metrics.

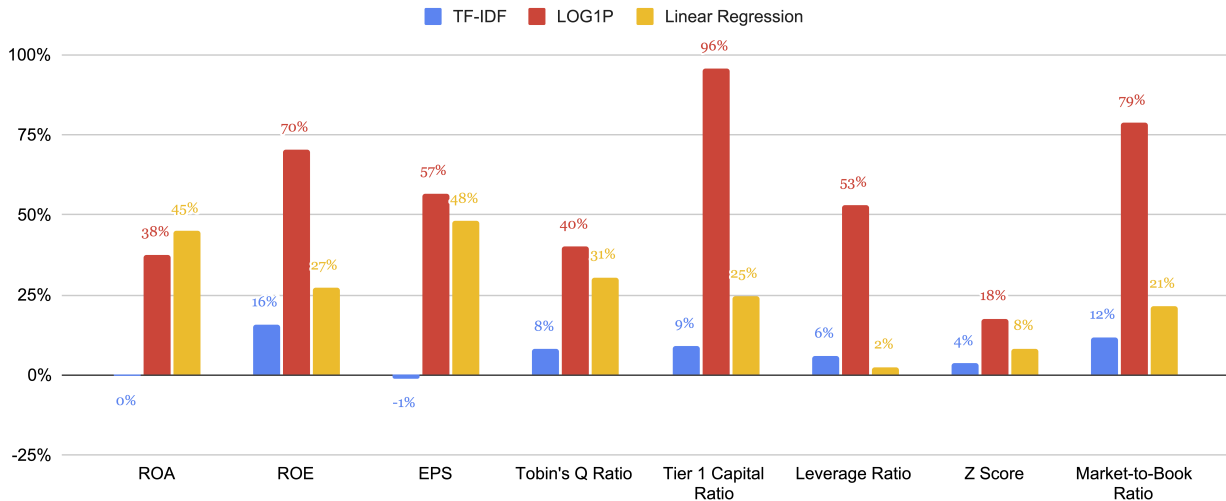
**Table 4.5: MSE results on Item 1A with historical scores. Best results in bold.**

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
TF-IDF	<b>0.000770</b>	0.008785	<b>0.000811</b>	0.016984	0.000248	0.002627	0.030511	0.005165
LOG1P+	0.001239	0.024953	0.001896	0.026026	0.005150	0.005270	0.035720	0.021402
Linear Regression	0.001407	0.010174	0.001577	0.022500	0.000299	0.002534	0.032102	0.005802
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	<b>0.015620</b>	0.000937	0.002527	0.030130	<b>0.004555</b>
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	<b>0.002471</b>	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
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	<b>0.007409</b>	0.000821	0.018100	0.000242	0.002715	0.030415	0.004894
FETILDA w/Nyströmformer	0.000780	0.007659	0.000925	0.016263	<b>0.000226</b>	0.002831	<b>0.029426</b>	0.005640

**Table 4.6: The effect of freezing the last or all layers of the underlying LM in FETILDA for Item 1A. Best results in bold.**

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
FETILDA w/BERT (Fully Unfrozen)	0.000811	0.008520	<b>0.000820</b>	0.019151	0.001353	<b>0.002559</b>	<b>0.029614</b>	0.004944
FETILDA w/BERT (Fully Frozen)	0.000856	0.008788	0.001076	0.018572	<b>0.000315</b>	0.010919	0.030225	<b>0.004908</b>
FETILDA w/BERT (Last Layer Frozen)	<b>0.000774</b>	<b>0.007803</b>	0.000824	<b>0.017883</b>	0.000726	0.002751	0.029729	0.004943
FETILDA w/FinBERT (Fully Unfrozen)	0.000867	0.008671	0.001171	<b>0.017383</b>	<b>0.000385</b>	0.002560	0.030583	0.004937
FETILDA w/FinBERT (Fully Frozen)	0.000976	<b>0.008626</b>	0.001274	0.018254	0.000428	<b>0.002471</b>	0.032155	<b>0.004911</b>
FETILDA w/FinBERT (Last Layer Frozen)	<b>0.000850</b>	0.008814	<b>0.000834</b>	0.018282	0.000485	0.002612	<b>0.030115</b>	0.004967
FETILDA w/Longformer (Fully Unfrozen)	<b>0.000790</b>	0.007940	0.000826	<b>0.015620</b>	0.000937	<b>0.002527</b>	<b>0.030130</b>	<b>0.004555</b>
FETILDA w/Longformer (Fully Frozen)	0.000811	0.008053	0.000854	0.018429	0.000930	0.002619	0.034284	0.004955
FETILDA w/Longformer (Last Layer Frozen)	0.000795	<b>0.007409</b>	<b>0.000821</b>	0.018100	<b>0.000242</b>	0.002715	0.030415	0.004894

**Item 1A – Textual Features with Numeric History:** Next, we report results on Item 1A, using both textual and numeric features. Table 4.5 shows the performance comparison between the four versions of our approach on Item 1A and the baseline methods, TF-IDF, LOG1P+ and linear regression. In six out of eight metrics, our method outperforms both



**Figure 4.4: Improvements for FETILDA on Item 1A over TF-IDF, LOG1P, and linear regression.**

TF-IDF and linear regression, with Longformer [4] performing the best in three cases and Nyströmformer performing the best in two cases. Overall FETILDA is best or second best in all eight cases. The baseline TF-IDF method performs the best in two out of the eight tasks with Item 1A, but is never the best for Item 7/7A. This may be due to the difference in document lengths between Item 1A and Item 7/7A. Table 4.1 shows that Item 7/7A is typically three times the length of Item 1A. With a shorter document length, we posit that TF-IDF feature vectors can represent Item 1A reasonably well, whereas the more complex contextual language models such as BERT, FinBERT, or Longformer do not have too much room for improvement. Nevertheless, FETILDA w/Longformer is still the best or second best in six out of the eight metrics.

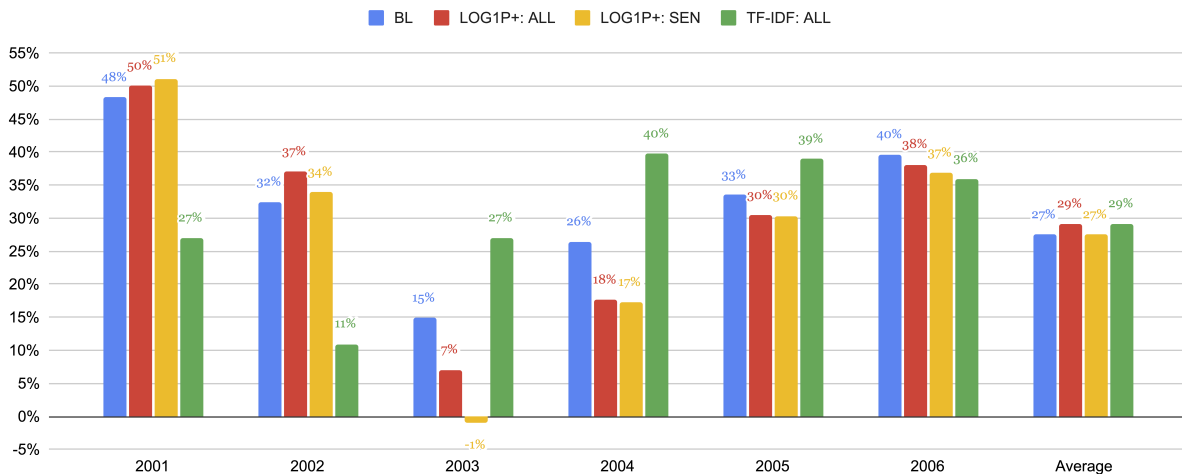
In Table 4.6, the effect of freezing the layers of all three underlying LMs produces different results for different LMs. None of the frozen (pre-trained) embeddings from Longformer outperforms the fine-tuned Longformer with unfrozen layers or only the last layer frozen in all of eight metrics. The frozen (pre-trained) embeddings from BERT performs better than its fine-tuned counterpart in two out of eight metrics, and worse in the remaining six metrics, with more noticeable improvements in predicting Tier 1 Capital Ratio target values. The frozen (pre-trained) embeddings from FinBERT performs better than its fine-tuned counterpart in three out of eight metrics, and worse for the remaining five metrics, with a slightly significant improvement in predicting Leverage Ratio target values. Overall, with frozen

layers, FETILDA is best or second best in five out of the eight metrics.

To see the percentage improvements, in Figure 4.4, for every metric, we select the best performing version of FETILDA with frozen layers in terms of mean squared error and calculate its improvement against the text and numeric baseline methods. We can observe that our approach outperforms LOG1P+ and linear regression on all of the metrics, and outperforms TF-IDF on six of the metrics, the exceptions being ROA and EPS, where TF-IDF has an extremely slight advantage.

**Table 4.7: Performance comparison (MSE) of FETILDA with BL (SVR) LOG1P+ [40], and TF-IDF. Best results in bold.**

Model\Year	2001	2002	2003	2004	2005	2006	Average
BL (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	<b>0.157950</b>	0.128220	0.130290	0.139980	0.150860
TF-IDF: ALL	0.123816	0.121450	0.218520	0.176087	0.148645	0.138113	0.154438
FETILDA w/BERT (Fully Unfrozen)	0.128406	0.111145	0.180670	0.111339	0.094401	0.091456	0.119569
FETILDA w/FinBERT (Fully Unfrozen)	<b>0.090408</b>	<b>0.108134</b>	0.172562	<b>0.106124</b>	<b>0.090766</b>	<b>0.088401</b>	<b>0.109399</b>
FETILDA w/Longformer (Fully Unfrozen)	0.124797	0.109595	0.183509	0.113019	0.094623	0.090408	0.119325
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
FETILDA w/Nyströmformer	0.120945	0.108224	0.174019	0.109716	0.095050	0.093098	0.116842



**Figure 4.5: Best improvements of our approach compared with the LOG1P+ methods used in Tsai and Wang [2017].**

**Table 4.8: The effect of freezing the last or all layers of the underlying LM in FETILDA for the FIN10K [24] dataset. Best results in bold.**

Model\Year	2001	2002	2003	2004	2005	2006	Average
FETILDA w/BERT (Fully Unfrozen)	0.128406	0.111145	0.180670	0.111339	0.094401	0.091456	0.119569
FETILDA w/BERT (Last Layer Frozen)	0.129132	0.111559	0.181691	0.110962	0.093300	<b>0.089595</b>	0.119373
FETILDA w/BERT (Fully Frozen)	<b>0.121354</b>	<b>0.108529</b>	<b>0.175446</b>	<b>0.108837</b>	<b>0.093004</b>	0.090500	<b>0.116278</b>
FETILDA w/FinBERT (Fully Unfrozen)	<b>0.090408</b>	<b>0.108134</b>	0.172562	<b>0.106124</b>	<b>0.090766</b>	<b>0.088401</b>	<b>0.109399</b>
FETILDA w/FinBERT (Last Layer Frozen)	0.125969	0.109420	0.176483	0.108349	0.092103	0.089228	0.116925
FETILDA w/FinBERT (Fully Frozen)	0.118620	0.113750	<b>0.159487</b>	0.108527	0.097878	0.095545	0.115635
FETILDA w/Longformer (Fully Unfrozen)	<b>0.124797</b>	<b>0.109595</b>	0.183509	0.113019	0.094623	0.090408	0.119325
FETILDA w/Longformer (Last Layer Frozen)	0.135215	0.114627	0.193750	0.117404	0.096162	0.089970	0.124521
FETILDA w/Longformer (Fully Frozen)	0.126380	0.109627	<b>0.169686</b>	<b>0.108116</b>	<b>0.091884</b>	<b>0.089902</b>	<b>0.115932</b>

**Comparison with LOG1P+ on FIN10K Dataset:** Table 4.7 compares the performance of our method with the baseline method and the two versions of the LOG1P+ model used in Tsai and Wang [2017], and TF-IDF. The baseline method BL is essentially support vector regression using the logarithm of the historic volatility for the prior twelve months, also from Tsai and Wang [2017]. LOG1P+: ALL refers to the model trained on the entire original text using the LOG1P features. Finally, LOG1P+: SEN refers to the model trained on only the sentiment bearing words taken from the L&M dictionary [26]. We report the results for these methods directly from their paper [40]. 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, LOG1P+:SEN also performs better than TF-IDF, even though TF-IDF performs better than LOG1P+:SEN for three out of the six years. However, as we can observe, with the exception of 2003, FETILDA outperforms LOG1P+:SEN by a large margin. Interestingly, on this much larger dataset, 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 Longformer by 8.3% (which is the second best method on average) and outperforms LOG1P+:SEN by 27%.

Figure 4.5 quantifies the percentage improvement of the best performing version of FETILDA, in terms of mean squared error for volatility prediction, versus the three approaches in Tsai and Wang [2017], as well as against TF-IDF baseline. Our approach outperforms, BL, TF-IDF and LOG1P+:ALL/SEN by a significant margin, on average (last set of bars) over 27% across the years. These results clearly showcase the benefits of contextual modeling of long text documents compared to using simpler textual features based on term

counts as done in LOG1P+.

Table 4.8 shows what happens to the FETILDA variants if we freeze the layers of the language model and used 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, only losing to the fully frozen FinBERT in 2003, but not so much for BERT and Longformer. Furthermore, the domain-specific pre-training in FinBERT followed by fine-tuning results in the best overall model. On average, fine-tuned FinBERT outperforms the frozen FinBERT by 5.4%.

**Table 4.9: Text only: MSE results on Item 7/7A. Best results in bold.**

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
TF-IDF	0.001391	0.026274	0.002983	0.032915	0.007553	0.006963	0.031883	0.020299
LOG1P	0.001111	0.025811	<b>0.001890</b>	0.026157	<b>0.006603</b>	<b>0.005131</b>	0.033935	0.020534
FETILDA w/BERT (Fully Unfrozen)	0.001050	0.026413	0.002961	0.028002	0.008361	0.007454	0.030932	0.029553
FETILDA w/FinBERT (Fully Unfrozen)	0.001154	0.026465	0.003091	0.027761	0.008383	0.007458	0.030910	0.030570
FETILDA w/Longformer (Fully Unfrozen)	0.001076	0.026505	0.002733	0.029851	0.008297	0.007502	0.031569	0.030292
FETILDA w/BERT (Fully Frozen)	0.001316	0.027031	0.004073	0.028120	0.008705	0.007629	<b>0.030625</b>	0.031909
FETILDA w/FinBERT (Fully Frozen)	0.001412	0.026144	0.003972	0.026486	0.008811	0.007240	0.031439	<b>0.019590</b>
FETILDA w/Longformer (Fully Frozen)	0.001282	0.027419	0.002808	0.026703	0.008271	0.007426	0.030885	0.031914
FETILDA w/BERT (Last Layer Frozen)	0.001290	0.026155	0.003222	<b>0.024124</b>	0.008280	0.007461	0.030947	0.033097
FETILDA w/FinBERT (Last Layer Frozen)	0.001059	<b>0.025671</b>	0.003829	0.025871	0.008512	0.007470	0.030912	0.033266
FETILDA w/Longformer (Last Layer Frozen)	<b>0.001041</b>	0.026152	0.002881	0.025567	0.008258	0.007495	0.030829	0.033509
FETILDA w/Nyströmformer	0.001098	0.028015	0.003027	0.032606	0.008233	0.007581	0.031836	0.030441

**Table 4.10: Text only: MSE results on Item 1A. Best results in bold.**

Models\Metrics	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
TF-IDF	0.001107	<b>0.021867</b>	0.002185	0.027620	0.005875	0.005729	0.033247	<b>0.013969</b>
LOG1P	0.001242	0.025091	<b>0.001903</b>	0.026174	<b>0.005185</b>	<b>0.005329</b>	0.035753	0.021562
FETILDA w/BERT (Fully Unfrozen)	<b>0.001082</b>	0.026529	0.002661	0.028673	0.007342	0.007391	0.032465	0.035507
FETILDA w/FinBERT (Fully Unfrozen)	0.001093	0.027217	0.003012	0.028507	0.007457	0.007813	0.032303	0.034169
FETILDA w/Longformer (Fully Unfrozen)	0.001121	0.028252	0.002799	0.028564	0.007274	0.007333	0.032014	0.034463
FETILDA w/BERT (Fully Frozen)	0.001172	0.027577	0.003064	0.027040	0.007243	0.007726	0.032852	0.033668
FETILDA w/FinBERT (Fully Frozen)	0.001398	0.026084	0.003174	<b>0.025455</b>	0.007291	0.007553	0.032445	0.036573
FETILDA w/Longformer (Fully Frozen)	0.001116	0.027497	0.002899	0.027739	0.007174	0.007396	0.034301	0.034219
FETILDA w/BERT (Last Layer Frozen)	0.001165	0.026862	0.002916	0.026215	0.007309	0.007321	0.031910	0.033216
FETILDA w/FinBERT (Last Layer Frozen)	0.001132	0.027768	0.002884	0.027820	0.007374	0.007337	<b>0.031887</b>	0.032898
FETILDA w/Longformer (Last Layer Frozen)	0.001098	0.027683	0.002855	0.028768	0.007226	0.007340	0.032266	0.034334
FETILDA w/Nyströmformer	0.001159	0.028497	0.003159	0.026934	0.007304	0.007888	0.033390	0.037246

**Item 1A and 7/7A – Textual Features Only:** In Tables 4.9 and 4.10 we report the MSE results for FETILDA variants compared to the TF-IDF and LOG1P baselines when we consider a purely text-based approach for predicting the numeric quantitative target variables. In other words, we ignore the numeric historic score for the target, and rely on only textual features. The aim of these experiments is to examine how predictive text alone is of the quantitative targets for regression. The results follow the same trend outlined for the text plus numeric experiments above, namely that on Item 7/7A, which are much longer,

FETILDA variants are best or second best on six of the eight metrics for US Banks dataset. However, the simpler LOG1P approach is competitive and yields the best performance on three of the metrics. On Item 1A, TF-IDF and LOG1P are the best on five metrics. As noted above, Item 1A is much smaller than Item 7/7A, and the simpler TF-IDF features can reasonably extract valuable information from shorter documents. Nevertheless, the main conclusion from the results is that purely text-based methods are not competitive with the combination of text and numeric for target variable regression. We notice a huge performance penalty to using text only versus text plus history. For example, on ROA, FETILDA w/Longformer on Item 7/7A achieves an MSE of 0.000813 with text plus numeric, whereas the MSE is as high as 0.001041 with text only. That is about 28% performance penalty for the purely text based approach. Incidentally, for purely numeric (Linear Regression) the MSE is even worse at 0.001432. This indicates that combining text with historic scores offers the best model. In particular, the FETILDA deep learning framework is able to extract highly informative features that combine both the textual and numeric information to yield the best performance overall.

#### 4.4 Algorithmic Choices

Having show the effectiveness of our FETILDA framework, we now present some results to justify some of the algorithmic choices, such as which chunk-level pooling strategy does the best and which FinBERT model performs the best. We also examine how the four final regression models compare.

**Table 4.11: A comparison of three different models of FinBERT.**

Results\Models	Araci [2019]	DeSola et al. [2019]	Yang et al. [2020]
Validation loss	0.0011482	0.0010539	<b>0.0010205</b>
Testing error	0.0007781	0.0008682	<b>0.0007458</b>

**FinBERT Variants:** As discussed in related work, there are four different FinBERT approaches proposed recently. Out of these, the implementation for Liu et al. [2020] FinBERT is not publicly available. We therefore compare the three FinBERT implementations that are available: Araci [2019], DeSola et al. [2019], and Yang et al. [2020]. Table 4.11 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 US Banks dataset. The results show that Yang et al. implementation results in the best performance. We thus choose the Yang et. al [2020] FinBERT 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.

**Table 4.12: A comparison of different 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	<b>0.0010205</b>
Testing MSE	0.0008547	0.0008221	0.0007686	0.0008820	<b>0.0007458</b>

**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 4.12, 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.

**Table 4.13: Best regression model for FETILDA w/Longformer with last layer frozen.**

FETILDA w/Longformer (Last Layer Frozen)	ROA	ROE	EPS	TQR	T1CR	LR	Z	MBR
Item 7/7A with historical scores	FC2	FC2	SVR	FC2	LR	KR	FC2	FC2
Item 1A with historical scores	FC2	FC2	SVR	FC2	KR	LR	FC2	LR
Item 7/7A without historical scores	FC2	FC2	FC2	FC2	FC2	FC2	FC2	LR
Item 1A without historical scores	FC2	FC2	FC2	FC2	FC2	FC2	FC2	KR

**Regression Model Alternatives:** We mentioned that for FETILDA we select the best regression model among four alternatives: 1) FC2: the output from the last layer of our model  $f_{c_2}$  that predicts the target variable (this is used in the model training step too), 2)

LR: Linear Regression, 3) SVR: Support vector Regression, and 4) KR: Kernel Regression. The best alternative is chosen based on the validation MSE scores. Table 4.13 shows which of these four methods is chosen as the best one for Item 7/7A and Item 1A with and without historical scores, for all of the eight metrics (with the last layer frozen for FETILDA w/Longformer). We can see that typically the final MLP layer  $fc_2$  does the best (FC2), but for some metrics SVR or KR or even LR can yield the best results.

## CHAPTER 5

### Conclusion and Future Work

In this thesis, we presented our novel FETILDA framework to address the main challenge of creating 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 12000 words on average. Even the SOTA language models struggle to create informative document representations for downstream tasks. Our FETILDA framework divides the long documents into smaller chunks, and first learns chunk-level contextual embeddings using SOTA language models like BERT, Longformer, and Nyströmformer, as well as domain-specific LMs like FinBERT. Next, we propose a Bi-LSTM layer with self-attention to pool together the chunk embeddings into the final document level embedding that weights different chunks based on the attention scores. We apply FETILDA to the task of predicting eight different KPIs for US Bank performance, as well as stock volatility prediction for US companies from FIN10K. Our approach is shown to outperform previous baselines, yielding SOTA results on the various regression tasks for the two datasets used. 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, and we have shown 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 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. For instance, we found that the performance is better on longer documents, but the language models lose some edge on shorter documents (e.g., those from Item 1A). We plan to explore this in more detail

---

Portions of this chapter have been submitted to: Bolun “Namir” Xia, Vipula D. Rawte, Aparna Gupta, and Mohammed J. Zaki. 2022. FETILDA: An Effective Framework For Fin-tuned Embeddings For Long Financial Text Documents. In DSAA’2022: IEEE International Conference on Data Science and Advanced Analytics: MLJ special issue on Foundations of Data Science, 2022. DSAA, Shenzhen, China.

on even larger 10-K datasets to confirm the trends. Also, we found that numeric data is important along with the text for the regression tasks. Purely textual features were not found to be competitive with the text plus numeric features. It remains an open question as to how to improve the purely text-based features. That is, to what extent text alone is indicative of quantitative metrics. Along these lines, one could also consider learning even larger domain-specific pre-trained models for financial text, with larger blocks (e.g., using Longformer or Nyströmformer instead of BERT for pre-training). Finally, we 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.

## BIBLIOGRAPHY

- [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 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 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 MDA and footnotes disclosures*.  
<https://dx.doi.org/10.2139/ssrn.2807546> (Retrieved on November, 02, 2019).
- [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. *arXiv:2004.05150* (2020).
- [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, Inc., 1877–1901.
- [6] Ching Yun Chang, Yue Zhang, Zhiyang Teng, Zahn Bozanic, and Bin Ke. Dec. 2016. Measuring the information content of financial news. In *Proc. of COLING 2016, the 26th Int. Conf. on Comput. Linguistics: Tech. Papers*. 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 *ICLR*.
- [8] Sanjiv Ranjan Das et al. Nov. 2014. Text and context: Language analytics in finance. *Found. and Trends® in 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. (08 2019).  
<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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- [11] Sophie Emerson, Ruairí Kennedy, Luke O’Shea, and John O’Brien. May 2019. Trends and applications of machine learning in quantitative finance. In *8th Int. Conf. on Econ. and Finance Res. (ICEFR 2019)*.
- [12] Jason Fernando. 2021. How return on equity (ROE) works.  
<https://www.investopedia.com/terms/r/returnonequity.asp> (Retrieved on April, 13, 2022).
- [13] Jason Fernando. 2022. Earnings per share (EPS).  
<https://www.investopedia.com/terms/e/eps.asp> (Retrieved on April, 13, 2022).
- [14] FRED. 2019. Bank Z-score for United States.  
<https://fred.stlouisfed.org/series/DDSI01USA645NWDB> (Retrieved on April, 13, 2022).
- [15] Marshall Hargrave. 2022. Return on assets (ROA).  
<https://www.investopedia.com/terms/r/returnonassets.asp> (Retrieved on April, 13, 2022).

- [16] Adam Hayes. 2021. Leverage ratio.  
<https://www.investopedia.com/terms/l/leverageratio.asp> (Retrieved on April, 13, 2022).
- [17] Adam Hayes. 2022. How the Q ratio – Tobin’s Q works.  
<https://www.investopedia.com/terms/q/qratio.asp> (Retrieved on April, 13, 2022).
- [18] Adam Hayes. 2022. Understanding tier 1 capital ratio.  
<https://www.investopedia.com/terms/t/tier-1-capital-ratio.asp> (Retrieved on April, 13, 2022).
- [19] Corporate Finance Institute. 2022. Market to book ratio.  
<https://corporatefinanceinstitute.com/resources/knowledge/valuation/market-to-book-ratio-price-book/> (Retrieved on April, 13, 2022).
- [20] Dan Jurafsky and James H. Martin. 2021. *Speech and Language Processing* (3 (draft ed.)).
- [21] Luckyson Khaidem, Snehanshu Saha, and Sudeepa Roy Dey. Apr. 2016. Predicting the direction of stock market prices using random forest. (Apr. 2016). arXiv:1605.00003.
- [22] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Jan. 2020. Reformer: The efficient transformer. (Jan. 2020). arXiv:2001.04451.
- [23] Shimon Kogan, Dimitry Levin, Bryan R Routledge, Jacob S Sagi, and Noah A Smith. Jun. 2009. Predicting risk from financial reports with regression. In *Proc. of Human Lang. Technol.: The 2009 Annu. Conf. of the North Amer. Chapter of the Assoc. for Comput. Linguistics*. 272–280.
- [24] 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 (Indianapolis, Indiana, USA) (CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 2441–2444.  
<https://doi.org/10.1145/2983323.2983328>

- [25] 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 Twenty-Ninth 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> Special Track on AI in FinTech.
- [26] Tim Loughran and Bill McDonald. Feb. 2011. When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The J. of Finance* 66, 1 (Feb. 2011), 35–65.
- [27] Tim Loughran and Bill McDonald. Sep. 2016. Textual analysis in accounting and finance: A survey. *J. of Accounting Res.* 54, 4 (Sep. 2016), 1187–1230.
- [28] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).
- [29] Paraskevi Nousi, Avraam Tsantekidis, Nikolaos Passalis, Adamantios Ntakaris, Juho Kannianen, Anastasios Tefas, Moncef Gabbouj, and Alexandros Iosifidis. May 2019. Machine learning for forecasting mid-price movements using limit order book data. *IEEE Access* 7 (May 2019), 64722–64736.
- [30] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Oct. 2014. Glove: Global vectors for word representation. In *Proc. of the 2014 Conf. on Empirical Methods in Natural Lang. Process. (EMNLP)*. 1532–1543.
- [31] Alec Radford and Karthik Narasimhan. 2018. Improving Language Understanding by Generative Pre-Training.
- [32] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).
- [33] 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. *Journal of Machine Learning Research* 21, 140 (2020), 1–67.

- [34] 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.
- [35] Marcelo Sardelich and Suresh Manandhar. Dec. 2018. Multimodal deep learning for short-term stock volatility prediction. (Dec. 2018). arXiv:1812.10479.
- [36] Marina Sedinkina, Nikolas Breilkopf, and Hinrich Schütze. Jun. 2019. Automatic Domain Adaptation Outperforms Manual Domain Adaptation for Predicting Financial Outcomes. In *Proc. of the 57th Annu. Meeting of the Assoc. for Comput. Linguistics*. 346–359.
- [37] Paul C Tetlock, Maytal Saar-Tsechansky, and Sofus Macskassy. Jun. 2008. More than words: Quantifying language to measure firms’ fundamentals. *The J. of Finance* 63, 3 (Jun. 2008), 1437–1467.
- [38] Christoph Kilian Theil, Sanja Štajner, and Heiner Stuckenschmidt. Jul. 2018. Word embeddings-based uncertainty detection in financial disclosures. In *Proc. of the 1st Workshop on Econ. and Natural Lang. Process.* 32–37.
- [39] Christoph Kilian Theil, Sanja Štajner, and Heiner Stuckenschmidt. Mar. 2020. Explaining financial uncertainty through specialized word embeddings. *ACM Trans. on Data Sci.* 1, 1 (Mar. 2020), 1–19.
- [40] Ming-Feng Tsai and Chuan-Ju Wang. Feb. 2017. On the risk prediction and analysis of soft information in finance reports. *Eur. J. of Oper. Res.* 257, 1 (Feb. 2017), 243–250.
- [41] Ming-Feng Tsai, Chuan-Ju Wang, and Po-Chuan Chien. Aug. 2016. Discovering finance keywords via continuous-space language models. *ACM Trans. on Manage. Inf. Syst. (TMIS)* 7, 3 (Aug. 2016), 1–17.
- [42] Ruey S Tsay. 2005. *Analysis of financial time series*. Vol. 543. John wiley & sons.
- [43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Jun. 2017. Attention is all you need. In *Advances in Neural Inf. Process. Syst.* 5998–6008.
- [44] Yan Wang and Xuelei Sherry Ni. Jan. 2019. A XGBoost risk model via feature selection and Bayesian hyper-parameter optimization. (Jan. 2019). arXiv:1901.08433.

- [45] Han Xiao. Mar 2020. bert-as-service. <https://github.com/hanxiao/bert-as-service> (Retrieved on March, 03, 2020).
- [46] 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. (2021).
- [47] Linyi Yang, Zheng Zhang, Su Xiong, Lirui Wei, James Ng, Lina Xu, and Ruihai Dong. Nov. 2018. Explainable text-driven neural network for stock prediction. In *2018 5th IEEE Int. Conf. on Cloud Comput. and Intell. Syst. (CCIS)*. 441–445.
- [48] Yi Yang, Mark Christopher Siy UY, and Allen Huang. 2020. FinBERT: A Pretrained Language Model for Financial Communications. arXiv:2006.08097
- [49] 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, Inc., 17283–17297.
- [50] Mohammed J Zaki and Wagner Meira Jr. 2020. *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2 ed.). Cambridge University Press.