

DiFiLE: A Knowledge-Distillation Longformer Model for Finance with Ensembling

Diana Hristova
 HWR Berlin, Germany
diana.hristova@hwr-berlin.de

Nirav Satani
 University of Koblenz, Germany
niravssatani@gmail.com

Abstract

10-K reports are a very important source of information in finance. Unfortunately, due to their text length, they can hardly be analyzed by state-of-the-art transformer-based methods. In this paper, we aim to address this by combining the fields of efficient attention mechanisms, knowledge distillation (KD), and ensembling. Our five-step approach, DiFiLE, first pre-processes the data and splits it into data chunks based on the report items. Then, for each chunk, we estimate a teacher Longformer model. This is followed by KD and the generation of the corresponding student models. Finally, we aggregate the results from the chunks with ensembling and in particular stacking. We evaluate DiFiLE on the 10-K reports of the DJIA companies. The results show high performance of the teacher model, which is then well mimicked by its distilled version, requiring 30% fewer resources.

Keywords: knowledge distillation, 10-K, transformer, ensembling, efficient attention mechanisms

1. Introduction

The field of finance focuses on the analysis of data to support different investment decisions. One important source for that analysis is the company's annual report that consists of both quantitative data on the current financial performance and qualitative textual data on other important factors. A 10-K report for instance, which is the annual report required for publicly traded companies in the U.S., contains statements on the company's business environment, legal cases, future risks, strategies and expectations, among others. Extracting those statements is crucial for financial decision-making, as they provide important information on companies' performance (Tsai & Wang, 2017). However, in the past years both the length and complexity of 10-K reports grew substantially (Dyer et al., 2017). This, together with the increasing digitalisation of the reports, has led to approaches for the automated analysis of 10-K's

textual parts using Natural Language Processing (NLP).

The idea behind such approaches is first to transform the texts into a structured form and then use the result to extract features for financial performance prediction. The literature can be split into frequency- and embedding-based approaches. For the former, the structure is derived from the frequencies of occurrence of the words. However, this approach fails to consider the dependency between the words, thus losing the language context. This limitation is addressed by embedding-based approaches, where words are represented by vectors in a high-dimensional vector space. The state-of-the-art embedding-based models are the ones derived from the transformer architecture. Both Glodd & Hristova (2023) and Hsieh & Hristova (2022) use such models to structure 10-Ks, but then compress the embeddings into features, such as topics and sentiment for performance prediction. However, using the embeddings directly can provide richer contextual information and has shown promising results on shorter text, such as news and social media posts (Kraus & Feuerriegel, 2017; Zhang et al., 2022).

The reason why this is hardly possible for 10-Ks is their length. For instance, the average length of a corporate annual report was 42,000 words in 2017 (Lesmy et al., 2019, p. 4) and is expected to grow with time. This is an issue, because transformer-based approaches, such as BERT and GPT, are restricted in the maximum text length they may process. BERT has a maximum input sequence length of 512 tokens (words split for training), while GPT-3 expects a maximum of 2,048 tokens. This may be enough for short texts, such as news, but it is not covering longer documents such as annual reports.

The length restriction is due to two reasons: a) the complex model architecture and b) the self-attention mechanism. Starting with a), the model architecture consists of many layers and neural networks, leading to the need to estimate a large number of parameters. BERT-Large has 340 million parameters (Devlin et al., 2019) and GPT-3 has 175 billion parameters (T. B. Brown et al., 2020). As a result, training such models

takes a very long time and requires dedicated hardware resources that are rarely available. Additionally, the trained models are very large, making their deployment on smaller devices, such as mobile phones, hardly possible. This substantially restricts their application and thus relevance for real-world cases.

To deal with a), the literature has looked into solutions for model compression, such as knowledge distillation (KD) and pruning (Tang et al., 2024). KD in particular has led to the development of lighter models like DistilBERT that achieves a reduction in "...the size of a BERT model by 40%, while retaining 97% of its language understanding capabilities and being 60% faster." (Sanh et al., 2020, p. 1).

Unfortunately, this still has not completely solved the issues with the input sequence length, leading us to b), the self-attention mechanism. It is responsible for examining the dependencies between words and therefore reflecting the language context. This is the only place in the model architecture that cannot be parallelized and thus a bottleneck for calculations. As a result, time and memory complexity increase quadratically with the length of the input text (Beltagy et al., 2020).

This bottleneck was the inspiration for the development of efficient attention mechanisms (Beltagy et al., 2020; N. Brown et al., 2023; Zaheer et al., 2021), which aim to reduce the number of self-attention computations, so that the model complexity becomes linear in the length of the input text. For instance, the trained Longformer and Big Bird models (Zaheer et al., 2021) can take input sequences of up to 4,096 tokens.

N. Brown et al. (2023) examined the benefits of combining both a) and b) for multiple transformer-based models and standard datasets. They showed that the distilled Longformer model led to the highest efficiency gains. This approach could potentially also address the issues with the long texts of 10-Ks, as discussed above, and provide richer contextual information for performance prediction. However, due to the length of the reports, they still need to be split reasonably and then combined again.

The aim of this paper is to explore this path by developing DiFiLE: A Knowledge-Distillation Longformer Model for Finance with Ensembling. It combines both KD and efficient-attention mechanisms for the analysis of 10-Ks, by additionally using ensembling for the aggregation of the results. The paper is structured as follows: in section 2, we present the relevant literature in the field and the corresponding research gap. This is followed by our methodology in section 3, which we then evaluate in

section 4. Finally, section 5 draws main conclusion and proposes directions for future research.

2. Related work

In this section, we first discuss 10-Ks and the literature applying NLP to them. This is followed by an overview of the transformer architecture, KD, efficient attention mechanisms and ensembling.

2.1. 10-Ks and NLP

10-Ks are annual reports required by the U.S. Securities and Exchange Commission for publicly traded companies in the U.S. They consist of 14 Items, with Item 1 (Business), Item 1A (Risk Factors), Item 3 (Legal Proceedings), Item 7 (Management Discussion & Analysis (MD&A)) and Item 7A (Market Risk) considered most important in the NLP literature (Glodd & Hristova, 2023). Most works focus on Item 7, which contains management's view on the company's situation.

As mentioned above, the literature applying NLP to 10-Ks can be split into frequency- and embedding-based approaches. In the first group, being the bigger one, Kogan et al. (2009) extract the frequencies of unigrams and bigrams in Item 7 to predict the volatility of stock returns with a regression model. Hájek (2018) uses the sentiment and the word frequencies in the same item together with quantitative financial data to predict abnormal stock returns with a neural network. Other frequency-based approaches can be found in Loughran & McDonald (2016) and Tsai & Wang (2017). However, as mentioned above, they fail to reflect the context between the words.

Embedding-based approaches address this, but there are few works looking at 10-Ks. For instance, Hsieh & Hristova (2022) combine transformer-based models on text summarization and sentiment together with topic modeling to predict future stock price growth rates. They use the topics and sentiment of the text as an input to a random forest model. Glodd & Hristova (2023) analyze 10-Ks for stock price prediction, by focusing on Forward-Looking Statements (FLS), defined as statements that convey information about the future of the company. They use the sentiment and number of FLS for stock price prediction again with a random forest model.

Those works are impressive and show promising results. However, they do not explore the whole potential of state-of-the-art NLP models. For instance, Kraus & Feuerriegel (2017) analyze financial news for predicting stock returns, directly using the embeddings of RNN and LSTM deep learning models. Zhang et al. (2022) use Twitter data to predict stock price

movements based on the embeddings from a transformer-based model. Such embeddings may also provide richer contextual information for 10-Ks and thus in this paper, we would like to explore this path for transformer-based models. To facilitate the understanding of our approach, in the following, we shortly explain the transformer architecture and KD.

2.2. Transformer architecture and KD

The transformer architecture, consisting of encoders and decoders was developed by Vaswani et al. (2017) and has seen many applications since then. The secret behind its success is the self-attention mechanism. Given an input text, it mathematically determines the relationships between different tokens. In order to allow for multiple relationships for a given token, the architecture contains multiple attention heads.

The result of the final block in transformer-based models is an embedding representation for each token. Depending on whether the task is regression or classification, this is then transformed into output values of appropriate dimension, called logits.

In order to train a transformer-based model, the training data is passed through the above architecture and the parameters are estimated that minimize the following loss:

$$Loss = F(logits, labels) \quad (1)$$

Here *logits* stands for the output of the last block, as discussed above, while *labels* represents the true values. *F* is the corresponding loss function, such as cross-entropy for classification and mean squared error (MSE) for regression.

As mentioned above, this architecture requires for both training and deployment enormous computational resources, hindering its application. Therefore, scholars have proposed different possibilities for model compression, such as KD.

KD compresses a big, teacher model, by training a smaller, student model based on it. The idea is that the student model mimics the teacher model and as such requires fewer resources for a comparable or a slightly lower performance. For classification, the student loss becomes (Gao et al., 2018; Hinton et al., 2015):

$$Loss_{DS} = \alpha F(logits_s, labels) + (1 - \alpha) F_D \left(\frac{logits_s}{T}, \frac{logits_T}{T} \right) * T^2 \quad (2)$$

Here *logits_s* are the student’s logits and *logits_t* are the teacher’s logits. *Loss_{DS}* is essentially a weighted average of the normal student loss from Equation (1) and a distillation loss, that measures the distance between the student’s and the teacher’s logits. The weight is determined by a parameter α . *T* stands for the temperature and has the aim to generate “soft”

targets, that help reduce the focus of the model on one class and provide richer information to the student for learning. Recommended values for *T* in the literature are {2, 3} (Hinton et al., 2015). *F_D* is the corresponding loss function (e.g., cross entropy) that usually is the same as *F*. It is multiplied by *T*² to make the distillation loss comparable in magnitude to the loss of the student model (Hinton et al., 2015).

KD was applied in multiple cases to compress state-of-the-art transformer-based models. The most famous one is the DistilBERT model (Sanh et al., 2020). It was trained as a student model for BERT, mimicking its architecture, but having half the number of encoders and therefore almost half of its parameters with similar performance. Another example is TinyBERT (Jiao et al., 2020) which in addition to the logits of the teacher BERT model considers other elements of the architecture, including the attention values. TinyBERT reaches 97% of the performance of its teacher model, “...being 7.5x smaller and 9.4x faster on inference.” (Jiao et al., 2020, p. 1).

KD addresses some of the issues with the computational requirements of transformer-based models. However, similar to their teacher models, student models are still strongly restricted in their input sequence length. The reason is the self-attention mechanism, as mentioned above. To address this, more efficient attention mechanisms have been developed. We present those in the next subsection.

2.4. Efficient attention mechanisms

The Longformer model, developed by Beltagy et al. (2020), divides the original self-attention mechanism into two parts: local and global attention. The idea is that it is not necessary to determine the self-attention between all tokens in an input sequence. Rather, it is enough to restrict the calculation to a limited fixed-sized window around a given token, reflecting its relevant local context. Additionally, this window can be dilated, meaning that some neighboring tokens are skipped in the calculation. This is the local attention component. However, for downstream tasks, such as classification, the whole context is needed and is often reflected in a single token. To incorporate this, the Longformer model calculates global attention (standard self-attention) only for some task-relevant tokens, such as the CLS-token in BERT. The complexity of the Longformer model is linear in the input sequence length, reducing memory substantially.

Another similar approach that also results in linear complexity is the Big Bird model (Zaheer et al., 2021), which also uses global and local attention, and additionally generates a random attention component.

The currently trained model has the same input sequence length as the Longformer model.

Combining efficient attention mechanisms with KD has the potential to reduce computational requirements even further. Brown et al. (2023) compared multiple models with efficient attention mechanisms, including Longformer and Big Bird, with their distilled versions on different short and long-context tasks, such as GLUE (Wang et al., 2019) Question Answering (QA) and Named Entity Recognition (NER). They found that KD decreases both inference and memory usage with an average of 45.2% and 2.6%, respectively. It also retains on average 94.6% of the teacher model’s performance on GLUE, 94.6% on QA and up to 98.8% on NER. The distilled version of the Longformer shows the greatest reduction in computational resources and the highest retention of performance in all cases for QA and NER and almost all cases for GLUE.

Even though the work by Brown et al. (2023) is impressive, they do not examine the application of the approach to 10-Ks. Keeping in mind their length, a combination of efficient transformer mechanisms and KD may facilitate the direct consideration of embeddings for performance prediction and thus richer contextual information. The purpose of this paper is to explore this path, by focusing on the Longformer model that showed the best results in Brown et al. (2023) and is open-sourced, making reducing the architecture possible. However, since the average annual report is longer than the input sequence length for which this model was trained, we split the text into items and apply ensembling for aggregation.

The idea behind ensembling is to combine multiple “weak” models to get a final, highly performant model. The combination can be done in

parallel (bagging) or sequentially (boosting). Stacking is a bagging approach, where an additional meta-model, such as a linear regression, is estimated for the combination. Stacking is more complex to perform, but has shown very good results in applications (Mohammed & Kora, 2023), so it is our choice here.

This completes the discussion of the related work. In the next section, we present DiFiLE: A Knowledge-Distillation Longformer Model for Finance with Ensembling. It addresses the gap of directly considering transformer-based embeddings for performance prediction from 10-Ks. This is done by combining the fields of KD, efficient attention mechanisms and ensembling.

3. Methodology

Figure 1 shows our methodology, which consists of five steps. The aim is to provide a pipeline, that takes the 10-K of a given company as input and predicts future stock price growth. To train the pipeline, we need a raw dataset of 10-Ks.

In Step 1, the raw dataset is matched with the stock price data and cleaned for further analysis. For each report, we derive the stock price growth based on the filing date. The idea is to examine the effect of the report’s publication on the financial performance of the company. We clean both the report texts and the stock price data for outliers and noise. We additionally select only the FLS statements, based on the approach in Glodd & Hristova (2023), who showed that those play an important role in future stock price growth.

In Step 2, we split the preprocessed dataset into data chunks, based on the report’s items. We do not consider all items, but only Item 1, Item 1A, Item 3,

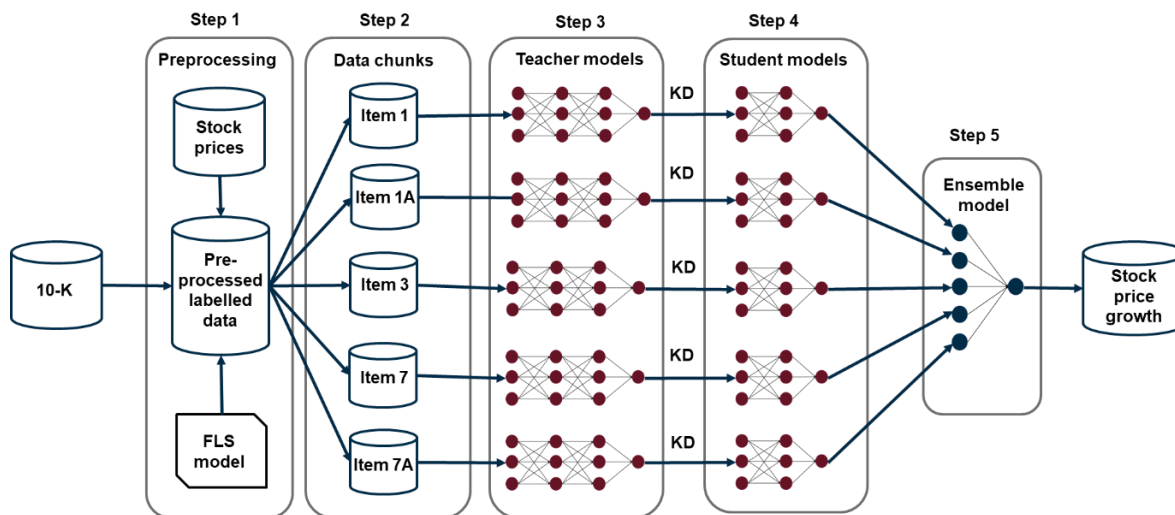


Figure 1. DiFiLE methodology

Item 7 and Item 7A, which as discussed above are expected to have the most informative content for future performance. The reason for this split is threefold. First, those items address different topics and thus different content. Therefore, it is reasonable to examine their language dependencies separately. Second, as the literature has focused mostly on Item 7, this enables us to compare our results with previous findings. Third, even though the combination of KD and efficient attention allows us to increase the input sequence length substantially, this still is not enough to cover a representative 10-K report in its whole length.

The result from Step 2 is a dataset for each chunk. This is then used in Step 3 to build one Longformer teacher model per chunk. The teacher models are estimated following the standard approach in the literature. We apply a train, validation, and test split to measure model performance. We additionally use the number of epochs to control overfitting.

In Step 4, we employ KD to generate the student models for each of the teacher models. Essentially, the student model copies the architecture of the teacher model by reducing its complexity. In our approach, complexity reduction is applied to the number of encoders and the number of attention heads. The former is based on the literature, where it has shown promising results (Sanh et al., 2020). The latter is motivated by the fact that the self-attention mechanism is one of the major causes of the enormous computational complexity in this architecture.

In this step, we first estimate the student models based solely on the data chunks. This provides us with a baseline model for performance comparison. Additionally, it already fine-tunes the model's parameters towards the data, which is important for better performance (Huang et al., 2023). After estimating the baseline student models, we apply KD with the teacher models from Step 3. The result is one KD student model for each data chunk.

To get one prediction per report and not per item (=data chunk), in Step 5 we use ensembling. In particular, we employ stacking to combine the KD student models in a final result. The reason for choosing stacking is that the estimation of the meta-model allows us to train the dependencies between the different data chunks and thus increase performance. This completes the overview of our methodology. In the next section, we present its evaluation.

4. Evaluation

In order to evaluate our methodology, we use 10-Ks from the SEC EDGAR database (SEC, 2024). We focus on companies from the Dow Jones Industrial

Average (DJIA) Index for the period 2006-2022, reflecting the longest available history.

4.1. Step 1: Preprocessing

In Step 1, the 10-Ks are matched with the stock prices of the companies before and after the filing date. Here, we consider two different time ranges: one day (or the first available price) and one month after the filing date. The aim is to examine both short- and medium-term effects as in Glodd & Hristova (2023). We introduce two types of prediction objectives: a relative stock price growth and a class representing positive, neutral and negative growth. The idea is to demonstrate our approach for both a regression and a classification problem. The raw dataset consists of 23,533 reports. After cleaning the data and removing outliers, we ended up with 18,520 reports for the next step of defining the data chunks. Unfortunately, for the classification problem, the neutral class was very small (5%) and thus we excluded it from the further analysis.

4.2. Step 2: Data chunks

As discussed above, the data chunks are generated based on the different items. Table 1 shows the descriptive statistics of the length of the text for each item. We can see that the longest texts are in Item 1, followed by Item 7 and Item 1 A. Items 3 and 7A show a substantially lower text length. The q-90 values are below the maximum input sequence length of the Longformer for all items, except Items 1 and 7, which however are close to it. Thus, we can apply the model to the data chunks. This would lead to the truncation of the texts with a length above 4,096 tokens as shown in the table. Note that not all reports contain all items, which is reflected by the difference in 'count', showing the number of data points.

Table 1. Statistics of text length per item

Statistics	1	1A	3	7	7A
q-90	4950	2425	582	4265	1805
q-75	3239	1599	257	3097	177
mean	2513	1244	261	2427	565
q-25	1175	601	32	1430	45
min	3	3	3	4	3
count	13317	11907	8091	12839	9763
count >4096	2157	203	56	1467	421

4.3. Step 3: Teacher models

For each of the five data chunks above and each of the two objectives, we estimate a separate Longformer model. For the estimation, we used a server with 1.5TB of RAM and four NVIDIA Tesla

V100 32GB. We split the data into 80% training, 10% test and 10% validation sets. Additionally, we chose the number of epochs in a way that prevents overfitting, testing up to 25 epochs. In particular, since there were often local minima/maxima in the loss development, we picked the number of epochs, where there was a clear tendency that the training loss decreases and the validation loss increases. We set the batch size to 8 and adjusted the default learning rate to $3.5e-5$. This was necessary for the models to converge.

As performance measures for the classification objective, we used accuracy and F1-scores. Accuracy represents the percentage of correctly predicted data points, while F1-score is a harmonic mean of the precision and recall of each class. As such, it provides richer information on class over- and underestimation.

To measure performance in the regression objective we used MSE, mean absolute error (MAE) and relative squared error (RSE). While MSE is a standard error measure for regression problems, MAE and RSE are more interpretable. RSE is MSE divided over the MSE of the average value in the data. Thus, it compares the model performance to that of a “lazy predictor”.

Table 2 presents the teacher model performance on the test set for each item and the classification objective. ‘Acc.’ Stands for ‘Accuracy’, ‘1 d’ for ‘1 day’, ‘1 m’ for ‘1 month’, ‘T’ for Time period, (1) and (2) are the classes for negative and positive growth respectively.

Table 2. Teacher performance (classification)

T		Item				
		1	1A	3	7	7A
1 d	Acc.	91%	89%	81%	87%	72%
	F1-(1)	91%	89%	81%	87%	68%
	F1-(2)	91%	89%	81%	87%	75%
1 m	Acc.	80%	80%	65%	62%	57%
	F1-(1)	80%	80%	68%	64%	41%
	F1-(2)	80%	80%	61%	61%	66%

The table shows that the best performance for both 1 day and 1 month is based on Item 1, followed by Item 1A and Item 7. Those three items are also the ones with the longest texts as shown in Table 1. This demonstrates the benefit of big deep learning models, capable of extracting complex contextual relationships within longer texts. Interestingly, Item 7, which is most commonly used in the literature, comes only third.

Both accuracy and F1-scores are very high here. The performance for 1 month is lower than that for 1 day. It shows that the FLS in the reports contain less predictive power for longer time horizons. This makes sense, as 1 month after filing the report, analysts have possibly read them and thus the information contained

in them is reflected in the stock price, as discussed in Glodd & Hristova (2023) and shown in their results.

Table 3. Teacher performance (regression)

T		Item				
		1	1A	3	7	7A
1 d	MAE E-02	4.955	4.874	5.011	4.950	5.001
	MSE E-03	4.430	4.284	4.638	4.358	4.604
	RSE %	96.9	94.6	97.5	97.7	98.6
1 m	MAE E-02	4.988	4.872	5.012	4.952	5.033
	MSE E-03	4.446	4.345	4.705	4.358	4.640
	RSE %	97.2	95.9	98.9	97.7	99.4

Table 3 presents the performance for the regression objective with similar conclusions. The best-performing items for both 1 day and 1 month are Item 1A, followed by Item 1 and Item 7, where the order of the last two depends on the performance metric. It seems that, while the discussions in the Business section matter the most for the direction of the stock price growth (classification), those in Risk Factors determine the exact growth size (regression). Here, again the performance decreases over the one-month horizon. Additionally, all RSE values are below one meaning a value better than the lazy predictor. Finally, the MAE shows a deviation of about 5% in the growth prediction, which is a good performance.

If we compare our results with Glodd & Hristova (2023), we see a similar, but not identical ranking in terms of items. In Glodd & Hristova (2023), Item 3 shows the highest importance, followed by 1A, 7A and 7. However, they consider the number of FLS and not the embeddings for their analysis. This may be an indication, that by solely focusing on the number, important context is lost for the estimation.

The model size for each teacher model, calculated as the number of parameters times the size of the chosen precision in bytes (Belkada & Dettmers, 2022) is 567 MB, as the models have 148,660,225 parameters. The training time is shown in Table 4.

Table 4. Teacher training time (hours:minutes)

Task	T	Item				
		1	1A	3	7	7A
Clas	1 d	07:36	06:50	04:42	07:20	03:01
	1 m	05:04	03:38	04:31	07:17	05:40
Regr	1 d	05:21	07:06	04:50	05:08	05:48
	1 m	07:56	07:05	03:17	05:07	05:47

‘Regr’ stands for the regression objective, ‘Clas’ for the classification one. It is between 7 and 3 hours

with the longest being for Item 1, 1 month and the shortest for Item 3 and 1 month. This completes the estimation of the teacher models. In the next subsection, we turn to the student ones.

4.4. Step 4: Student models

Based on the above teacher models, we estimate the student models using KD. We tested different weights for α (0.05, 0.1, 0.2, 0.3) and obtained the best results with 0.05. Additionally, we reduced the number of encoders in half to six (similar to DistilBERT) and set the number of attention heads to eight. We did experiments with less attention heads, but in those cases, the model failed to converge, indicating not enough complexity. As discussed above, we first trained the baseline student model on the available data and then adjusted it using KD. The training of the student model follows the same approach as that of the teacher model.

For KD, we used different distillation losses for regression and classification. For classification, Equation 3 is applied with cross entropy and $T = 3$. We tested $T = 2$ in pretests, but that showed worse results. For regression, we calculated the MSE loss between the student and the teacher logits. Additionally, we had to reduce the batch size to two due to resource restrictions and therefore the learning rate to $3.5e-6$. The following tables show the student model performance on the test set for each item, the classification and regression objective, and baseline and KD student models.

For classification, the baseline student models in Table 5 show lower performance than the corresponding teacher models in Table 2, for all cases except for Item 3 and 1 month. Item 3 has the shortest texts and therefore it is possible that the teacher’s complexity is not necessary to reflect context dependencies. This is also visible for 1 day, where the teacher model is better, but compared with the other items with a much lower margin. Thus, it is not surprising that for all cases, except Item 3, applying KD substantially increases students’ performance (see Table 6), bringing down the average difference to the accuracy of the teacher from 12% to 6%.

This is confirmed by the regression results in Table 7 and Table 8. The teacher model always outperforms the student and therefore KD substantially improves the student’s performance, achieving a deviation of at most 1% in the RSE of the teacher. In some cases, KD even shows slightly better results than the teacher. This is an interesting observation potentially demonstrating the ability of the KD approach to extract the best dependencies from both the student and the teacher models.

Table 5. Baseline student performance (classification)

		Item				
T		1	1A	3	7	7A
1 d	Acc.	76%	71%	77%	70%	66%
	F1-(1)	74%	64%	75%	63%	62%
	F1-(2)	78%	76%	78%	75%	70%
1 m	Acc.	66%	62%	76%	53%	57%
	F1-(1)	59%	50%	76%	21%	45%
	F1-(2)	71%	69%	76%	66%	64%

Table 6. KD student performance (classification)

		Item				
T		1	1A	3	7	7A
1 d	Acc.	84%	81%	81%	78%	71%
	F1-(1)	83%	79%	81%	76%	63%
	F1-(2)	84%	83%	81%	79%	75%
1 m	Acc.	71%	69%	69%	56%	59%
	F1-(1)	75%	67%	69%	66%	46%
	F1-(2)	65%	71%	68%	36%	67%

Table 7. Baseline student performance (regression)

		Item				
T		1	1A	3	7	7A
1 d	MAE	4.973	4.892	5.084	4.952	5.095
	E-02					
	MSE	4.471	4.352	4.700	4.382	4.642
	E-03					
	RSE %	97.8	96.1	98.8	98.2	99.4
1 m	MAE	5.086	4.907	5.035	4.977	5.094
	E-02					
	MSE	4.559	4.378	4.727	4.395	4.644
	E-03					
	RSE %	99.7	96.7	99.4	98.5	99.4

Table 8. KD student performance (regression)

		Item				
T		1	1A	3	7	7A
1 d	MAE	4.956	4.895	5.084	4.937	5.001
	E-02					
	MSE	4.444	4.331	4.700	4.362	4.610
	E-03					
	RSE %	97.1	95.6	97.5	97.8	98.7
1 m	MAE	4.953	4.869	4.999	4.947	5.024
	E-02					
	MSE	4.395	4.336	4.701	4.378	4.627
	E-03					
	RSE %	96.1	95.8	98.8	98.1	99.1

The model size for all baseline student and KD models is 364 MB, as the models have 95,502,337 parameters. That is 64% the size of the teacher model. Table 9 shows the training time of the student after KD as a percentage of that of the teacher.

Table 9. Student training time as % of teacher

Task	T	Item				
		1	1A	3	7	7A
Clas	1 d	64%	64%	62%	63%	79%
	1 m	71%	87%	68%	66%	46%
Regr	1 d	61%	67%	61%	61%	67%
	1 m	62%	54%	61%	62%	67%

We can see that the training time mostly reflects the reduction in the parameters, being around 70% of that of the teacher. This, together with the smaller size and comparable performance demonstrates the benefits of our approach and that the combination of KD and a Longformer model is a reasonable tool for the stock price growth prediction based on 10-Ks. Since, we now have five different predictions, one for each item, in the following we aggregate those with ensembling.

4.5. Step 5: Ensemble models

In Step 5, we ensemble the KD models in one prediction using stacking. We tested different meta-models, including a fully connected linear layer, a logistic and linear regression and a support vector machine (SVM). Note that no feature selection was applied. We also applied simple averaging and majority voting in pretests. The best results were shown by the regressions and the SVM and since the former are also interpretable, we present them in the following.

As inputs to the ensemblers, we use the predicted stock price growth, while for classification, we extract the corresponding probability for class one of the item models. Note that, due to different text cleanings, the items in 4.4 do not necessarily have the same dataset, as not all items are contained in all reports. This holds true especially in Items 3 and 7A, which are presented in comparatively few reports and barely overlap. Thus, generating one dataset for all items proved challenging as it would have contained too few data points. To address this, we examined different combinations of well-performing items. Among them, considering items 1, 1A and 7 showed the best results. Additionally, for regression, excluding Item 7 improved the results further, pointing at potentially redundant information.

Table 10 shows the performance of the ensemblers. We can see that, the performance improved as compared to Table 6 and Table 8, showing that different items contain different information relevant to the stock price growth direction and thus a combination is beneficial. As far as comparable, our results are better than those in Glodd & Hristova (2023) and Hsieh & Hristova

(2022), who both use a random forest model, thus reflecting the role of deeper contextual information.

Table 10. Ensembler performance

Task	T	Metrics				
		Acc	F1-(1)	F1-(2)		
Clas	1 d	88%	90%	86%		
	1 m	82%	86%	77%		
Regr		MAE E-02	MSE E-03	RSE %	RSE % 1)	MSE 2)
	1 d	4.485	3.624	90.8	97.9	29.61
	1 m	4.282	3.596	90.1	100.4	
1) Best in Glodd & Hristova (2023)						
2) Best in and Hsieh & Hristova (2022), given in E-03, 1 year						

Table 11 provides the weights for the corresponding items in the different ensemblers. For classification and 1 day, all items have comparable importance, with the probability for class 1 of the items increasing that for the same class of the ensemble (logistic regression). For 1 month, Item 7 seems to go in the opposite direction, with a low weight. The reason for that may be that there is an overlap with the information in the other items.

Table 11. Ensembler weights

Task	T	Item		
		1	1A	7
Clas	1 d	-3.16	-3.29	-3.09
	1 m	-3.48	-3.27	0.67
Regr	1 d	0.53	0.83	n/a
	1 m	0.98	0.28	n/a

For regression, the items have different weights and also the order of those weights depends on the time period. This implies that while for the general direction of the stock price growth, the information in different items is equally important, for the exact development, the item number matters. Additionally, for short-term prediction, Risk Factors (1A) are more important, while for medium-term ones, this is Business (1). That makes sense, as in the medium-term general industry characteristics play a bigger role, as opposed to potentially short-term risks.

To sum up, our evaluation generated interesting and promising results. The estimated teacher models had a good performance for both the regression and classification objectives. As far as comparison is possible, we could show an improvement to the literature, by using embeddings directly for stock price growth prediction. Interestingly, the best-performing items are the ones with the longest texts, which again supports our approach as opposed to the ones aggregating the embeddings into a set of features.

The KD was also highly successful. For the cases, where the baseline student underperformed the teacher (all except one), KD could substantially improve its performance, making it close to that of the teacher with around 70% of its size and training time. In some instances, it even outperformed the teacher, showing

its potential to learn the best from the teacher and the student models.

The results for Item 3 and classification deserve a special attention. Here, the student baseline model was better than the teacher, while Item 3 contains the shortest texts. It demonstrates that high model complexity is not always beneficial and that especially for shorter texts, one should always examine possible architecture simplifications, thus saving resources.

In terms of ensembling, we could see an improvement as opposed to the single items, showing that their combination is advantageous. Additionally, for the direction of the stock price, items are considered equally important, while for the exact movement, the item type matters, depending on the time period. This completes our evaluation. In the next section, we draw main conclusions and discuss limitations and paths for future research.

5. Conclusion

10-Ks are an important source of information regarding the financial situation of U.S. companies, used by analysts and investors to support business decisions. A substantial part of those reports consists of text, which has to be extracted and analyzed for future performance prediction. NLP offers the tools to do that, however, most of the existing works focus on frequency-based approaches that lead to a substantial loss of context. Those that apply embedding-based approaches, do not use the embeddings directly, but rather features derived from them.

The reason is the length of the reports, making the application of state-of-the-art transformer-based models hardly possible, due to the enormous computational resources. To address this issue in other fields, researchers have developed methods for model compression and efficient self-attention mechanisms. These methods could potentially improve performance prediction from 10-Ks by reducing the computational complexity.

In this paper, we explored this path by proposing DiFiLE: A Knowledge-Distillation Longformer Model for Finance with Ensembling. DiFiLE provides a pipeline that takes as input a 10-K and generates the stock price growth after the report's filing date. It consists of five steps. Step 1 applies standard preprocessing and data preparation techniques. Step 2 splits the preprocessed text into data chunks based on its items. In Step 3, we estimate a Longformer teacher model for each data chunk. This is then used to estimate the corresponding student models in Step 4 via KD. Finally, in Step 5, we aggregate the results with ensembling.

We evaluated our approach on the 10-Ks for the DJIA companies, for both regression and classification objectives and different time periods after the filing date. The results show a very good performance of the teacher Longformer model, especially for long texts. However, this comes with the need for dedicated hardware resources, long training time and big model size. We could address those points with KD, which reduced the model size and training time by about 30% with a slightly lower performance. This is crucial, as it allows the training and more importantly application of the models with fewer resources, making it more feasible in practical use cases. In particular, many companies and institutions either do not possess the technical infrastructure for heavier computations or do not have the budget to pay for corresponding online resources. Developing lighter model therefore increases their accessibility and practical relevance.

Interestingly, in some cases, KD even showed better results than the teacher model. Additionally, for short texts, the teacher Longformer model was outperformed by the baseline student model. The reason for both could be that for short texts, the complete complexity of the Longformer model is not needed. Thus, KD potentially manages to combine the best of a complex teacher model for long texts and a simpler student model for shorter texts.

Our approach also has some limitations. Since we used the pre-trained Longformer model, we were bound by its maximum sequence length. Future research could aim at training Longformer models with higher sequence length, which however requires hardware resources that are rarely available. Additionally, we focus on the text parts of the report for performance prediction. Similar to the literature, future work could combine it in the ensembling step with quantitative financial data for better results. Finally, other models, such as Jamba (Lieber et al., 2024) could be considered in the future.

6. References

- Belkada, Y., & Dettmers, T. (2022, August 17). *A Gentle Introduction to 8-bit Matrix Multiplication for transformers at scale using Hugging Face Transformers, Accelerate and bitsandbytes*. <https://huggingface.co/blog/hf-bitsandbytes-integration>
- Beltagy, I., Peters, M. E., & Cohan, A. (2020). *Longformer: The Long-Document Transformer* (arXiv:2004.05150). arXiv. <http://arxiv.org/abs/2004.05150>
- Brown, N., Williamson, A., Anderson, T., & Lawrence, L. (2023). *Efficient Transformer Knowledge Distillation: A Performance Review* (arXiv:2311.13657). arXiv. <http://arxiv.org/abs/2311.13657>

- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (arXiv:1810.04805). arXiv. <http://arxiv.org/abs/1810.04805>
- Dyer, T., Lang, M., & Stice-Lawrence, L. (2017). The evolution of 10-K textual disclosure: Evidence from Latent Dirichlet Allocation. *Journal of Accounting and Economics*, 64(2), 221–245. <https://doi.org/10.1016/j.jacceco.2017.07.002>
- Gao, G., Mishra, B., & Ramazzotti, D. (2018). Causal data science for financial stress testing. *Journal of Computational Science*, 26, 294–304. <https://doi.org/10.1016/j.jocs.2018.04.003>
- Glodd, A., & Hristova, D. (2023). Extraction of Forward-looking Financial Information for Stock Price Prediction from Annual Reports Using NLP Techniques. In T. X. Bui (Ed.), *56th Hawaii International Conference on System Sciences, HICSS 2023, Maui, Hawaii, USA, January 3-6, 2023* (pp. 5572–5581). ScholarSpace. <https://hdl.handle.net/10125/103313>
- Gou, J., Yu, B., Maybank, S. J., & Tao, D. (2021). Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6), 1789–1819. <https://doi.org/10.1007/s11263-021-01453-z>
- Hájek, P. (2018). Combining bag-of-words and sentiment features of annual reports to predict abnormal stock returns. *Neural Computing and Applications*, 29(7), 343–358. <https://doi.org/10.1007/s00521-017-3194-2>
- Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network* (arXiv:1503.02531). arXiv. <http://arxiv.org/abs/1503.02531>
- Hsieh, H.-T., & Hristova, D. (2022). Transformer-based Summarization and Sentiment Analysis of SEC 10-K Annual Reports for Company Performance Prediction. *55th Hawaii International Conference on System Sciences, HICSS 2022, Virtual Event / Maui, Hawaii, USA, January 4-7, 2022*, 1–10.
- Huang, A. H., Wang, H., & Yang, Y. (2023). FinBERT: A Large Language Model for Extracting Information from Financial Text*. *Contemporary Accounting Research*, 40(2), 806–841. <https://doi.org/10.1111/1911-3846.12832>
- Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., & Liu, Q. (2020). *TinyBERT: Distilling BERT for Natural Language Understanding* (arXiv:1909.10351). arXiv. <http://arxiv.org/abs/1909.10351>
- Kogan, S., Levin, D., Routledge, B. R., Sagi, J. S., & Smith, N. A. (2009). Predicting risk from financial reports with regression. *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics on - NAACL '09*, 272. <https://doi.org/10.3115/1620754.1620794>
- Kraus, M., & Feuerriegel, S. (2017). Decision support from financial disclosures with deep neural networks and transfer learning. *Decision Support Systems*, 104, 38–48. <https://doi.org/10.1016/j.dss.2017.10.001>
- Lesmy, D., Muchnik, L., & Mugerma, Y. (2019). Doyoureadme? Temporal Trends in the Language Complexity of Financial Reporting. *SSRN Electronic Journal*. <https://doi.org/10.2139/ssrn.3469073>
- Lieber, O., Lenz, B., Bata, H., Cohen, G., Osin, J., Dalmedigos, I., Safahi, E., Meiron, S., Belinkov, Y., Shalev-Shwartz, S., Abend, O., Alon, R., Asida, T., Bergman, A., Glozman, R., Gokhman, M., Manevich, A., Ratner, N., Rozen, N., ... Shoham, Y. (2024). *Jamba: A Hybrid Transformer-Mamba Language Model* (arXiv:2403.19887). arXiv. <http://arxiv.org/abs/2403.19887>
- Loughran, T., & McDonald, B. (2016). Textual Analysis in Accounting and Finance: A Survey. *Journal of Accounting Research*, 54(4), 1187–1230. <https://doi.org/10.1111/1475-679X.12123>
- Mohammed, A., & Kora, R. (2023). A comprehensive review on ensemble deep learning: Opportunities and challenges. *Journal of King Saud University - Computer and Information Sciences*, 35(2), 757–774. <https://doi.org/10.1016/j.jksuci.2023.01.014>
- Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2020). *DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter* (arXiv:1910.01108). arXiv. <http://arxiv.org/abs/1910.01108>
- SEC. (2024). *EDGAR Database* [Dataset]. <https://www.sec.gov/edgar/search-and-access>.
- Tang, Y., Wang, Y., Guo, J., Tu, Z., Han, K., Hu, H., & Tao, D. (2024). *A Survey on Transformer Compression* (arXiv:2402.05964). arXiv. <http://arxiv.org/abs/2402.05964>
- Tsai, M.-F., & Wang, C.-J. (2017). On the risk prediction and analysis of soft information in finance reports. *European Journal of Operational Research*, 257(1), 243–250. <https://doi.org/10.1016/j.ejor.2016.06.069>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. <http://arxiv.org/abs/1706.03762>
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding* (arXiv:1804.07461). arXiv. <http://arxiv.org/abs/1804.07461>
- Zaheer, M., Guruganesh, G., Dubey, A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2021). *Big Bird: Transformers for Longer Sequences* (arXiv:2007.14062). arXiv. <http://arxiv.org/abs/2007.14062>
- Zhang, Q., Qin, C., Zhang, Y., Bao, F., Zhang, C., & Liu, P. (2022). Transformer-based attention network for stock movement prediction. *Expert Systems with Applications*, 202, 117239. <https://doi.org/10.1016/j.eswa.2022.117239>