

ARI5121

Applied Natural Language Processing

Text Processing

Dr Claudia Borg

Karsten Guenther

The paper that I chose to discuss and replicate covered the attempt to develop a language model based on BERT, to tackle NLP tasks in financial domain [1]. The objective of this paper is polarity analysis, which is classifying text as positive, negative or neutral which denotes positive, negative or neutral financial news respectively.

It aimed to address two challenges:

- 1) The most sophisticated classification methods that make use of neural nets require vast amounts of labelled data and labelling financial text snippets requires costly expertise.
- 2) The sentiment analysis models trained on general corpora are not suited to the task, because financial texts have a specialized language with unique vocabulary and have a tendency to use vague expressions instead of easily-identified negative/positive words.

NLP transfer learning methods was proposed as a promising solution to both of the challenges mentioned above, and are the focus of this paper.

The core idea behind these models is that by training language models on very large corpora and then initializing down-stream models with the weights learned from the language modelling task, a much better performance can be achieved. This approach should, in theory, be an answer to the scarcity of labelled data problem. Language models don't require any labels, since the task is predicting the next word. They can learn how to represent the semantic information. That leaves the fine-tuning on domain specific labelled data only the task of learning how to use this semantic information to predict the labels.

One particular component of the transfer learning methods is the ability to further pre-train the language models on domain specific unlabelled corpus. Thus, the model can learn the semantic relations in the text of the target domain, which is likely to have a different distribution than a general corpus. This approach is especially promising for a niche domain like finance, since the language and vocabulary used is dramatically different than a general one

The goal of this paper is to test these hypothesized advantages of using and fine-tuning pre-trained language models for financial domain. For that, sentiment of a sentence from a financial news article towards the financial actor depicted in the sentence will be tried to be predicted, using the Financial PhraseBank created by Malo et al. (2014) and FiQA.

In summary the paper provided the following contributions:

- The introduction of FinBERT, which is a language model based on BERT for financial NLP tasks.
- The evaluation of FinBERT on two financial sentiment analysis datasets. State-of-the-art achievements on FiQA sentiment scoring and Financial PhraseBank.
- Comparison of two other pre-trained language models, ULM-Fit and ELMo for financial sentiment analysis and compare these with FinBERT.
- Implementation of experiments to investigate several aspects of the model, including:
 - effects of further pre-training on financial corpus
 - training strategies to prevent catastrophic forgetting
 - fine-tuning only a small subset of model layers for decreasing training time without a significant drop in performance.

Analysis: A detailed analysis of the system used in the paper.

FinBERT is a pre-trained NLP model based on BERT, Google's revolutionary transformer model. [2]

Understanding the transformer's model architecture:

The Transformer is an attention-based architecture for modelling sequential information. The attention mechanism allows for learning contextual relations between words and the encoder and decoder mechanisms aim to solve sequence-to-sequence tasks.

The encoder consists of multiple identical transformer layers. Each layer has a multi-headed self-attention layer and a fully connected feed-forward network. For one self-attention layer, three mappings from embeddings (key, query and value) are learned. Using each token's key and all tokens' query vectors, a similarity score is calculated with dot product. These scores are used to weight the value vectors to arrive at the new representation of the token. With the multi-headed self-attention, these layers are concatenated together, so that the sequence can be evaluated from varying "perspectives". Then the resulted vectors go through fully connected networks with shared parameters.

Understanding bidirectional encoder representations from transformers (BERT):

BERT is a language model that consists of a set of transformer encoders stacked on top of each other.

The training of Bert is done in two phases, the first phase is pre-training where the model understands language and context and the second phase is fine-tuning where the model learns how to solve specific problems, in this case sentiment prediction from financial text.

BERT was trained by masking 15% of the tokens with the goal to guess them. An additional objective was to predict the next sentence. BERT "masks" a randomly selected 15% of all tokens for the second phase BERT is trained on is "next sentence prediction". Given two sentences, the model predicts whether or not these two actually follow each other. The input sequence is represented with token embeddings, segment embeddings and position embeddings. Two tokens denoted by [CLS] and [SEP] are added to the beginning and end of the sequence respectively. For all classification tasks, including the next sentence prediction, [CLS] token issued. With a SoftMax layer over vocabulary on top of the last encoder layer the masked tokens are predicted.

Implementation of BERT for financial domain (FinBert):

The author suggested new approaches and improvements for the implementation of BERT, he suggested further pre-training on domain corpus, suggested new ways to implement BERT for classification and regression and suggested new training strategies for fine-tuning to prevent catastrophic forgetting. [3] [4]

Further pre-training.

It was suggested that further pre-training a language model on a target domain(finance) corpus improves the eventual classification performance.

- Another pre-training stage for the model on a relatively large corpus from the target domain (finance). For that, they further pre-trained a BERT language model on a financial corpus TRC2-financial.
- Next, fine-tuning the model only on the sentences from the training classification dataset.

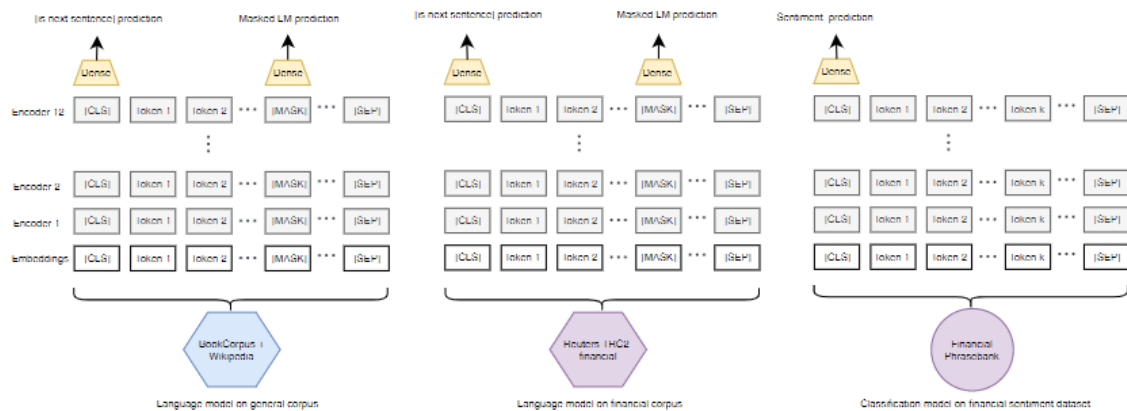


Figure 1 BERT base, pre-training and fine-tuning process

FinBERT for text classification:

- Sentiment classification is conducted by adding a dense layer after the last hidden state of the [CLS] token. Then, the classifier network is trained on the labelled sentiment dataset.

FinBERT for regression:

- While the focus of this paper is classification, the authors also implemented a regression with almost the same architecture on a different dataset with continuous targets. The only difference is that the loss function being used is mean squared error instead of the cross-entropy loss.

Training strategies to prevent catastrophic forgetting.

Catastrophic forgetting is a significant danger with this fine-tuning approach. Because the fine-tuning procedure can quickly cause model to “forget” the information from language modelling task as it tries to adapt to the new task.

In order to deal with this phenomenon, three techniques proposed by Howard and Ruder (2018) were applied:

- **slanted triangular learning rates:**
 - Slanted triangular learning rate applies a learning rate schedule in the shape of a slanted triangular, that is, learning rate first linearly increases up to some point and after that point linearly decreases.
- **discriminative fine-tuning:**
 - Discriminative fine-tuning is using lower learning rates for lower layers on the network.
 - The assumption behind this method is that the lower layers represent the deep-level language information, while the upper ones include information for actual classification task. Therefore, we fine-tune them differently.
- **and gradual unfreezing:**
 - With gradual freezing, we start training with all layers but the classifier layer as frozen.
 - During training we gradually unfreeze all of the layers starting from the highest one, so that the lower-level features become the least fine-tuned ones.
 - Hence, during the initial stages of training it is prevented for model to “forget” low-level language information that it learned from pre-training.

Parameters/Datasets:

Dataset:

TRC2-financial.

In order to further pre-train BERT, we use a financial corpus we call TRC2-financial. It is a subset of Reuters' TRC24, which consists of 1.8M news articles that were published by Reuters between 2008 and 2010. We filter for some financial keywords in order to make corpus more relevant and in limits with the compute power available.

The resulting corpus, TRC2-financial, includes 46,143 documents with more than 29M words and nearly 400K sentences.

Financial PhraseBank.

The main sentiment analysis dataset used in this paper is Financial PhraseBank⁵ from Malo et al. 2014. Financial Phrasebank consists of 4845 English sentences selected randomly from financial news found on LexisNexis database.

These sentences then were annotated by 16 people with background in finance and business. The annotators were asked to give labels according to how they think the information in the sentence might affect the mentioned company stock price. The dataset also includes information regarding the agreement levels on sentences among annotators.

We set aside 20% of all sentences as test and 20% of the remaining as validation set. In the end, our trainset includes 3101 examples. For some of the experiments, we also make use of 10-fold cross validation.

FiQA Sentiment:

FiQA is a dataset that was created for WWW '18 conference financial opinion mining and question answering challenge.

We use the data for Task 1, which includes 1,174 financial news headlines and tweets with their corresponding sentiment score.

Unlike Financial Phrasebank, the targets for this dataset are continuous ranging between $[-1,1]$ with 1 being the most positive. Each example also has information regarding which financial entity is targeted in the sentence. We do 10-fold cross validation for evaluation of the model for this dataset.

Method: Model and Parameters

Author's FinBert Implementation:

The author's implementation BERT, we use a dropout probability of $p=0.1$, warm-up proportion of 0.2, maximum sequence length of 64 tokens, a learning rate of $2e-5$ and a mini-batch size of 64. The model was trained for 4 epochs, evaluated the validation set and chose the best one. For discriminative fine-tuning the discrimination rate as set at 0.85. The training was done with only the classification layer unfrozen, after each third of a training epoch the authors unfroze the next layer.

The last two parameters discriminate and gradual unfreeze determine whether to apply the corresponding technique against catastrophic forgetting.

```
config = Config( data_dir=cl_data_path,
                 bert_model=bertmodel,
                 num_train_epochs=4.0,
                 model_dir=cl_path,
                 max_seq_length = 64,
                 train_batch_size = 32,
                 learning_rate = 2e-5,
                 output_mode='classification',
                 warm_up_proportion=0.2,
                 local_rank=-1,
                 discriminate=True,
                 gradual_unfreeze=True )
```

Figure 2: Authors configuration of BERT

For contrastive experiments, the authors considered baselines with three different methods:

1. *LSTM classifier with GLoVe embeddings:*

2. *LSTM classifier with ELMo embeddings:*

- The implemented two classifiers using bidirectional LSTM models. In both of them, a hidden size of 128 was used, with the last hidden state size being 256 due to bidirectionality.
- A fully connected feed-forward layer maps the last hidden state to a vector of three, representing likelihood of three labels.
- The difference between two models is that one uses GLoVe embeddings, while the other uses ELMo embeddings.
- A dropout probability of 0.3 and a learning rate of $3e-5$ is used in both models.
- We train them until there is no improvement in validation loss for 10 epochs

3. *ULMFit classifier:*

- Classification with ULMFit consisted of three steps.
- The first step of pre-training a language model was already done and the pre-trained weights are released by Howard and Ruder (2018).
- Further pre-training AWD-LSTM language model on TRC2-financial corpus for 3 epochs.
- Fine-tune the model for classification on Financial PhraseBank dataset, by adding a fully-connected layer to the output of pre-trained language model.

Evaluation Metrics:

- For evaluation of classification models, the author used three metrics:
 - Accuracy
 - Cross entropy loss
 - Macro F1 average.

The results of FinBERT, the baseline methods and state-of-the-art on Financial PhraseBank dataset classification task can be seen below.

Table 2: Experimental Results on the Financial PhraseBank dataset

Model	All data			Data with 100% agreement		
	Loss	Accuracy	F1 Score	Loss	Accuracy	F1 Score
LSTM	0.81	0.71	0.64	0.57	0.81	0.74
LSTM with ELMo	0.72	0.75	0.7	0.50	0.84	0.77
ULMFit	0.41	0.83	0.79	0.20	0.93	0.91
LPS	-	0.71	0.71	-	0.79	0.80
HSC	-	0.71	0.76	-	0.83	0.86
FinSSLX	-	-	-	-	0.91	0.88
FinBERT	0.37	0.86	0.84	0.13	0.97	0.95

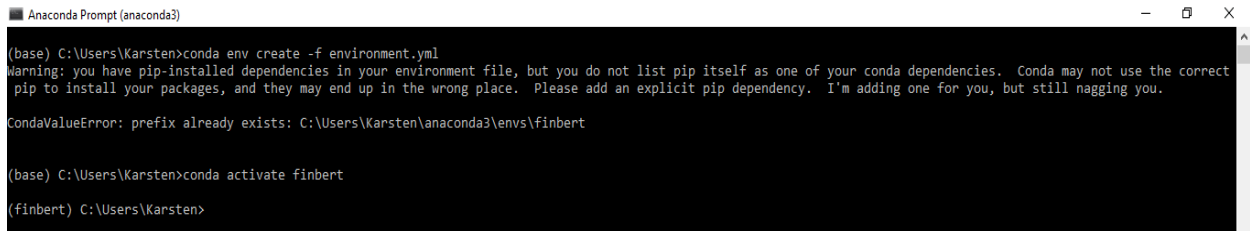
Figure 3: Author's model Results.

Implementation:

What worked/failed, did you obtain the same results?

Code Replication:

I installed all the dependencies by creating the Conda environment finbert from the given environment.yml file and activating it.



```
Anaconda Prompt (anaconda3)
(base) C:\Users\Karsten>conda env create -f environment.yml
Warning: you have pip-installed dependencies in your environment file, but you do not list pip itself as one of your conda dependencies. Conda may not use the correct
pip to install your packages, and they may end up in the wrong place. Please add an explicit pip dependency. I'm adding one for you, but still nagging you.
CondaValueError: prefix already exists: C:\Users\Karsten\anaconda3\envs\finbert

(base) C:\Users\Karsten>conda activate finbert
(finbert) C:\Users\Karsten>
```

Models Used:

Models used in this analysis are Language model trained on TRC2 and Sentiment analysis model trained on Financial PhraseBank. [5] [6] For both of these models I proceeded to do the following:

(NB: to run files please follow these steps as the 2 .bin will not be included in the zip file (420mb each).

- Create a directory for the model. For example: models/sentiment/<model directory name>
- Download the model and put it into the directory you just created.
- Put a copy of config.json in this same directory.

There are two datasets used for FinBERT:

The language model further training is done on a subset of Reuters TRC2 dataset [7]

For sentiment analysis, Financial PhraseBank is used from [8]

To train the model on the same dataset, after downloading it, three files should be created under the data/sentiment data folder as train.csv, validation.csv, test.csv.

Following steps to create these files:

- Download the Financial PhraseBank.
- Get the path of Sentences_AllAgree.txt file in the FinancialPhraseBank-v1.0 zip.

Prepare the model:

Setting path variables:

cl_path: the path where the classification model is saved.

cl_data_path: the path of the directory that contains the data files of train.csv, validation.csv, test.csv.

Configuring training parameters:

I implemented various training parameters exploring different epochs, classification verities and freezing layers. The parameters I experimented with are presented below together with their metrics.

config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=4, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=True, gradual_unfreeze=True)	Loss:0.41 Accuracy:0.91				
	Classification Report:				
		precision	recall	f1-score	support
	0	0.34	0.86	0.49	14
	1	0.99	0.89	0.94	220
	2	0.94	0.97	0.96	105
	accuracy			0.91	339
	macro avg	0.76	0.90	0.80	339
	weighted avg	0.95	0.91	0.93	339

Figure 4: Model with 6 layers frozen:

config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=4, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=True, gradual_unfreeze=True)	Loss:0.43 Accuracy:0.91				
	Classification Report:				
		precision	recall	f1-score	support
	0	0.29	0.79	0.42	14
	1	1.00	0.89	0.94	220
	2	0.95	0.96	0.96	105
	accuracy			0.91	339
	macro avg	0.75	0.88	0.77	339
	weighted avg	0.96	0.91	0.92	339

Figure 5: Model with 8 layers frozen.

config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=4, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=True, gradual_unfreeze=True)	Loss:0.53 Accuracy:0.87				
	Classification Report:				
		precision	recall	f1-score	support
	0	0.26	0.86	0.39	14
	1	1.00	0.83	0.91	220
	2	0.93	0.97	0.95	105
	accuracy			0.87	339
	macro avg	0.73	0.89	0.75	339
	weighted avg	0.95	0.87	0.90	339

Figure 6: Model with 10 layers frozen.

<pre> config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=4, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=False, gradual_unfreeze=True) </pre>	<p>Loss:0.43 Accuracy:0.90</p> <p>Classification Report:</p> <table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.31</td> <td>0.79</td> <td>0.44</td> <td>14</td> </tr> <tr> <td>1</td> <td>1.00</td> <td>0.88</td> <td>0.94</td> <td>220</td> </tr> <tr> <td>2</td> <td>0.93</td> <td>0.96</td> <td>0.94</td> <td>105</td> </tr> <tr> <td>accuracy</td> <td></td> <td></td> <td>0.90</td> <td>339</td> </tr> <tr> <td>macro avg</td> <td>0.74</td> <td>0.88</td> <td>0.77</td> <td>339</td> </tr> <tr> <td>weighted avg</td> <td>0.95</td> <td>0.90</td> <td>0.92</td> <td>339</td> </tr> </tbody> </table>		precision	recall	f1-score	support	0	0.31	0.79	0.44	14	1	1.00	0.88	0.94	220	2	0.93	0.96	0.94	105	accuracy			0.90	339	macro avg	0.74	0.88	0.77	339	weighted avg	0.95	0.90	0.92	339
	precision	recall	f1-score	support																																
0	0.31	0.79	0.44	14																																
1	1.00	0.88	0.94	220																																
2	0.93	0.96	0.94	105																																
accuracy			0.90	339																																
macro avg	0.74	0.88	0.77	339																																
weighted avg	0.95	0.90	0.92	339																																

Figure 7: Model with 8 layers frozen and not discriminating.

<pre> config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=2, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=False, gradual_unfreeze=True) </pre>	<p>Loss:0.85 Accuracy:0.51</p> <p>Classification Report:</p> <table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.05</td> <td>0.64</td> <td>0.10</td> <td>14</td> </tr> <tr> <td>1</td> <td>0.97</td> <td>0.32</td> <td>0.48</td> <td>220</td> </tr> <tr> <td>2</td> <td>0.92</td> <td>0.90</td> <td>0.91</td> <td>105</td> </tr> <tr> <td>accuracy</td> <td></td> <td></td> <td>0.51</td> <td>339</td> </tr> <tr> <td>macro avg</td> <td>0.65</td> <td>0.62</td> <td>0.50</td> <td>339</td> </tr> <tr> <td>weighted avg</td> <td>0.92</td> <td>0.51</td> <td>0.60</td> <td>339</td> </tr> </tbody> </table>		precision	recall	f1-score	support	0	0.05	0.64	0.10	14	1	0.97	0.32	0.48	220	2	0.92	0.90	0.91	105	accuracy			0.51	339	macro avg	0.65	0.62	0.50	339	weighted avg	0.92	0.51	0.60	339
	precision	recall	f1-score	support																																
0	0.05	0.64	0.10	14																																
1	0.97	0.32	0.48	220																																
2	0.92	0.90	0.91	105																																
accuracy			0.51	339																																
macro avg	0.65	0.62	0.50	339																																
weighted avg	0.92	0.51	0.60	339																																

Figure 8: Model with 8 layers frozen, not discriminating and running for 2 epochs.

<pre> config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=2, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=True, gradual_unfreeze=True) </pre>	<p>Loss:0.95 Accuracy:0.45</p> <p>Classification Report:</p> <table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.05</td> <td>0.71</td> <td>0.10</td> <td>14</td> </tr> <tr> <td>1</td> <td>0.98</td> <td>0.21</td> <td>0.34</td> <td>220</td> </tr> <tr> <td>2</td> <td>0.92</td> <td>0.92</td> <td>0.92</td> <td>105</td> </tr> <tr> <td>accuracy</td> <td></td> <td></td> <td>0.45</td> <td>339</td> </tr> <tr> <td>macro avg</td> <td>0.65</td> <td>0.62</td> <td>0.45</td> <td>339</td> </tr> <tr> <td>weighted avg</td> <td>0.92</td> <td>0.45</td> <td>0.51</td> <td>339</td> </tr> </tbody> </table>		precision	recall	f1-score	support	0	0.05	0.71	0.10	14	1	0.98	0.21	0.34	220	2	0.92	0.92	0.92	105	accuracy			0.45	339	macro avg	0.65	0.62	0.45	339	weighted avg	0.92	0.45	0.51	339
	precision	recall	f1-score	support																																
0	0.05	0.71	0.10	14																																
1	0.98	0.21	0.34	220																																
2	0.92	0.92	0.92	105																																
accuracy			0.45	339																																
macro avg	0.65	0.62	0.45	339																																
weighted avg	0.92	0.45	0.51	339																																

Figure 9: Model with 8 layers frozen running for 2 epochs.

<pre> config = Config(data_dir=cl_data_path, bert_model=bertmodel, num_train_epochs=4, model_dir=cl_path, max_seq_length = 48, train_batch_size = 32, learning_rate = 2e-5, output_mode='classification', warm_up_proportion=0.2, local_rank=-1, discriminate=False, gradual_unfreeze=True) </pre>	<p>Loss:0.67 Accuracy:0.84</p> <p>Classification Report:</p> <table border="1"> <thead> <tr> <th></th> <th>precision</th> <th>recall</th> <th>f1-score</th> <th>support</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0.22</td> <td>0.79</td> <td>0.34</td> <td>14</td> </tr> <tr> <td>1</td> <td>1.00</td> <td>0.79</td> <td>0.88</td> <td>220</td> </tr> <tr> <td>2</td> <td>0.87</td> <td>0.95</td> <td>0.91</td> <td>105</td> </tr> <tr> <td>accuracy</td> <td></td> <td></td> <td>0.84</td> <td>339</td> </tr> <tr> <td>macro avg</td> <td>0.70</td> <td>0.84</td> <td>0.71</td> <td>339</td> </tr> <tr> <td>weighted avg</td> <td>0.93</td> <td>0.84</td> <td>0.87</td> <td>339</td> </tr> </tbody> </table>		precision	recall	f1-score	support	0	0.22	0.79	0.34	14	1	1.00	0.79	0.88	220	2	0.87	0.95	0.91	105	accuracy			0.84	339	macro avg	0.70	0.84	0.71	339	weighted avg	0.93	0.84	0.87	339
	precision	recall	f1-score	support																																
0	0.22	0.79	0.34	14																																
1	1.00	0.79	0.88	220																																
2	0.87	0.95	0.91	105																																
accuracy			0.84	339																																
macro avg	0.70	0.84	0.71	339																																
weighted avg	0.93	0.84	0.87	339																																

Figure 4: Model with 6 layers frozen and not discriminating.

Applying all three of the strategies produce the best performance in terms of test loss and accuracy.

Gradual unfreezing and discriminative fine-tuning have the same reasoning behind them, higher level features should be fine-tuned more than the lower-level ones, since information learned from language modelling are mostly present in the lower levels resulting in gradual unfreezing being the most important technique for our case

One way that catastrophic forgetting can show itself is the sudden increase in validation loss after several epochs. As model is trained, it quickly starts to overfit when no measure is taken accordingly. The model achieves the best performance on validation set after the first epoch and then starts to overfit. While with all three techniques applied, model is much more stable. The other combinations lie between these two cases

BERT has 12 Transformer encoder layers. It is not necessarily a given that the last layer captures the most relevant information regarding classification task during language model training. This might be indicative of two factors:

- When the higher layers are used the model that is being trained is larger, hence possibly more powerful,
- The lower layers capture deeper semantic information; hence they struggle to fine-tune that information for classification

Future Work: A way forward, slight or major improvements to the system.

Financial sentiment analysis is not a goal on its own, it is as useful as it can support financial decisions. One way that the model might be extended, could be using FinBERT directly with stock market return data (both in terms of directionality and volatility) on financial news.

FinBERT is good enough for extracting explicit sentiments, but modelling implicit information that is not necessarily apparent even to those who are writing the text should be a challenging task.

Another possible extension can be using FinBERT for other natural language processing tasks such as named entity recognition or question answering in financial domain.

Bibliography

- [1] D. Araci, "FinBERT: Financial Sentiment Analysis with Pre-trained Language Models," 2019-06-25.
- [2] M.-W. C. K. L. K. T. Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 24 May 2019.
- [3] D. Araci, "github.com," 25 06 2019. [Online]. Available: <https://github.com/ProsusAI/finBERT>.
- [4] d. araci, "github.com/ProsusAI/finBERT," ProsusAI, 2019. [Online]. Available: https://github.com/ProsusAI/finBERT/blob/master/notebooks/finbert_training.ipynb.
- [5] "finbert language model," [Online]. Available: https://prosus-public.s3-eu-west-1.amazonaws.com/finbert/language-model/pytorch_model.bin.
- [6] f. s. model. [Online]. Available: https://prosus-public.s3-eu-west-1.amazonaws.com/finbert/finbert-sentiment/pytorch_model.bin.
- [7] Reuters, "Reuters," [Online]. Available: <https://trec.nist.gov/data/reuters/reuters.html>.
- [8] FinancialPhraseBank, "researchgate," [Online]. Available: https://www.researchgate.net/publication/251231364_FinancialPhraseBank-v10.