Engineering Embedding Spaces for Financial Sentiment Regression

Artem Arshakyan
Department of Computer Science
University College London
artem.arshakyan.24@ucl.ac.uk

Abstract

Financial sentiment analysis (FSA) is a common upstream machine learning task, formulated as both a regression and classification problem. Current state-of-the-art (SOTA) models on the former leverage contextual embeddings from pre-trained large language models (LLMs), while integrating domain knowledge from financial lexicons. These methods fail to address the non-uniform (anisotropic) nature of LLM embedding spaces, ignoring powerful partial regularisation techniques through contrastive learning. We reframe sentence embedding learning as a contrastive learning problem, using cosine similarity loss to map an embedding space that reflects sentiment similarity—a process we call "fanning." These engineered embeddings are used for regression through dense neural layers. To achieve this, we construct a dataset of randomly sampled sentence pairs, define a proxy for cosine similarity, and perform two-stage hold-out validation: first for embedding training, then for dense layer regression. Our method achieves SOTA performance on the FiQA 2018 Task 1 dataset, with an average MSE of 0.0483 and an average R^2 of 0.7073, demonstrating the effectiveness of embedding space fanning as a feature engineering technique for FSA.

1 Introduction

The internet is an immense repository for text data, with vast amount of unprocessed information still flowing in daily. In turn, we need sophisticated natural language processing (NLP) pipelines that process this raw information and exploit this information influx. One important area of NLP is sentiment analysis, the task of determining the sentiment conveyed in a piece of text. An improper labeling framework could prove detrimental to many use cases, from erroneously gauging customer satisfaction to introducing false features in a downstream financial problem.

Financial sentiment falls under analytical data. Although considered a lagging indicator, it has proven useful when used as a feature in a downstream financial machine learning task. This can include the generation of trading signals, for instance. [1] argues that financial sentiment—e.g. from news headlines and social media posts—can act as a proxy for investor sentiment, which aggregates to market sentiment, the underlying trading behavior seen in the market. They discuss different methods of obtaining this sentiment in literature, the most used being through LLMs.

Transfer learning methods through pretrained LLMs are considered the SOTA methods for current NLP tasks, including financial sentiment analysis. They leverage contextual word representations from LLMs trained on large corpora, such as the Bidirectional Encoder Representation from Transformers (BERT) model [2], after fine-tuning to their sentiment task. These contextualised representations, called embeddings, are necessary for any NLP task, since static word embeddings are incapable of capturing polysemy. The word "short" has multiple meanings—both short as in "short in length" and short as in "to sell a borrowed asset." Clearly, the sentiment would vary depending on the context the word is used in. In finance, the task of labeling this sentiment is posed either as a classification

problem (positive, negative) or a regression task, where a score in [-1, 1] represents sentiment intensity. In this paper, we will focus on the latter representation.

To tackle such regression problems, [3] presented FinBERT, a model further pretrained through masked language modeling (MLM) on the TRC2 Reuters corpus, which attempts to capture the specialised vocabulary ubiquitous in finance. The financial domain is laden with domain-exclusive jargon and phrases. As noted in [4], "The word 'liability' seemingly expresses negative sentiment, yet in a financial context would be neutral." After fine-tuning, this model achieved SOTA on the leading regression benchmark dataset. Later, [4] beat this SOTA, using a specialised BERT model, Robustly optimised BERT approach (RoBERTa) [5], with a model they named "k-RoBERTa." This model leverages general-purpose and financial lexicons to inject domain sentiment information. It processes lexicon sentiment scores of words within a sentence using CNNs and LSTMs, concatenates them with average pooled RoBERTa embeddings, and labels them using dense neural layers.

Both of these methods ignore the biggest flaw of BERT-like models: they do not produce explicit sentence embeddings [6]. Their word embeddings are situated in a narrow cone in their ambient space, exhibiting immensely high anisotropy [7]. We look to contrastive learning, a technique ubiquitous in sentence textual similarity tasks, to partially remedy this.

The use of contrastive learning in embedding models like SBERT [6] set the SOTA on semantic mapping of the embedding space. Posing the question as "why x, and not y" better constructs the embedding space for your intended purposes. We investigate whether using such sentence transformers to transform the embedding space into one directly structured around sentiment labels could serve as a useful form of feature engineering for regression. This technique should partially regularise anisotropy, which we believe creates a more meaningful embedding space that aids dense layer interpolation on the test set. Just as importantly, it also injects information about labeling bias into the embeddings. To the best of our knowledge, this is the first work to attempt the following method on a sentiment analysis regression task.

We adopt a two-stage hold-out validation approach for model development. First, we fine-tune the embedding model by further splitting the original training data into new train and validation sets composed of sentence pairs, each labeled using a similarity proxy. The model is trained to minimise mean squared error (MSE) between the cosine similarity of paired sentence embeddings and their respective proxy values. Then, we train dense neural layers using the original train and validation fine-tuned embeddings. To note, we approach this problem carefully to mitigate data leakage.

Our method results in a model that slightly outperforms the current SOTA [4] on the FiQA 2018 Task 1 dataset. Although the validation methods differ, our approach is still extremely promising. It is much more interpretable, and rooted in logical, sequential deductions. Due to the immense representation space of dense neural layers, the problem needs to be formulated as exactly as possible, since this representation search is conducted in a locally greedy fashion. This is accomplished in our model. It explicitly creates an embedding space in which sentences with similar sentiment are close together, while those with dissimilar sentiment are far apart. We call this phenomenon "fanning" the embedding space.

2 Methodology

2.1 Preliminaries

2.1.1 Encoder-only Transformer

The encoder-only Transformer [8, 2] focuses mainly on generating contextualised word embeddings. It breaks a sentence $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ into subword units called tokens, and maps these tokens into an embedded sequence $\mathbf{z} = (\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_m})$, where $\mathbf{z}_i \in \mathbb{R}^{d_{\text{model}}}$.[1] Each encoder layer is made up of 2 sub layers—a multi-head attention layer and a stack of two dense neural layers. This structure is completed by applying residual connection and layer normalisation after each sub layer, followed by stacking the desired amount of encoder modules, as seen in Figure 1. Effectively, the output after each sub layer is LayerNorm(x+SubLayer(x)) for each token embedding x.

[1] Notice that m and n are not equal due to special tokens and potential subword tokens.

2.1.2 Self Attention

After the input sequence $\mathbf{x}$ is converted into its corresponding embedding matrix $\mathbf{X} \in \mathbb{R}^{m \times d_{\text{model}}}$, it is processed by H parallel attention heads. For each head $h = 1, \ldots, \text{H}$, the matrix $\mathbf{X}$ is linearly projected into different sets of query, key, and value matrices: $Q_h = \mathbf{X}W_h^Q, K_h = \mathbf{X}W_h^K, V_h = \mathbf{X}W_h^V$, where $W_h^Q, W_h^K, W_h^V \in \mathbb{R}^{d_{\text{model}} \times d_k}$ [8]. In self attention mechanisms, key, query, and value pairs come from the same embedding matrix. Each attention head calculates a similarity score between their query and key pairs by taking the dot product of each token embedding together, scaled by $\frac{1}{\sqrt{d_k}}$, before applying the softmax function below row-wise:

$$\sigma(\mathbf{z})_i = \frac{\exp z_i}{\sum_{j=1}^{K} \exp z_j}.$$

This new matrix is multiplied with the value matrix, resulting in each new token embedding being a weighted sum of all the original token embeddings, where the weights reflect their similarity to the corresponding token. This can be formulated as:

$$\text{Attention}(Q_h, K_h, V_h) = \text{softmax}\left(\frac{Q_h K_h^{\text{T}}}{\sqrt{d_k}}\right) V_h.$$

These attention heads are then concatenated and multiplied by another weight matrix, $W^O \in \mathbb{R}^{H d_k \times d_{\text{model}}}$, to get the final contextual embeddings for all the tokens.

$$\text{MultiHead}(Q, K, V) = [\text{Attention}_1 | \text{Attention}_2 | \ldots | \text{Attention}_H] W^O,$$

$$\text{where Attention}_i = \text{Attention}(Q_i, K_i, V_i) \text{ and [A | B] is an augmented matrix.}$$

Contextualised embeddings are then fed into standard multi-layer perceptrons (MLPs).

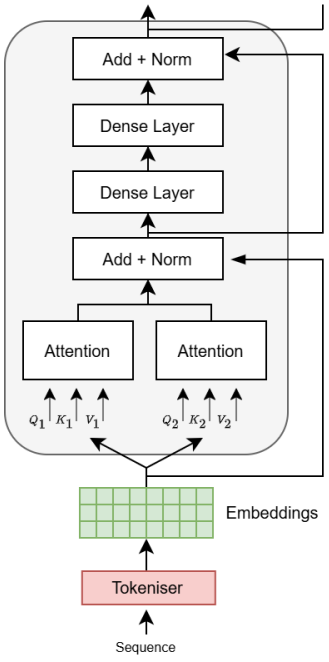

Figure 1: One stack of a two-headed attention encoder block. The grey block is repeated 12 times in BERT.

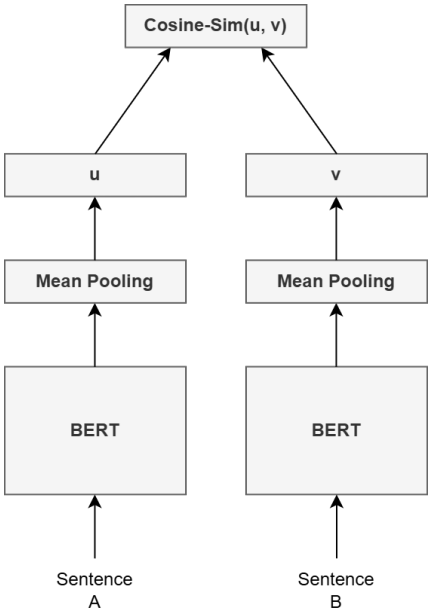

Figure 2: SBERT training. "Cosine-Sim" computes cosine similarity between u, v embeddings and adjusts using MSE.

2.1.3 Dense Neural Layers

MLPs are composed of affine transformations, parametrised by weight and bias vectors, passed through a non-linear function called the activation function. For regression tasks, the activation

function is what prevents the problem from degenerating to a simple gradient descent OLS problem. This activation function can be be a variety of functions. We use the ReLU function, $\text{ReLU}(x) = \max(0, x)$. The two-layer MLP can be written as:

$$\text{ReLU}(XW_1^T + b_1)W_2^T + b_2.$$

2.1.4 BERT Family

BERT: BERT [2], in its base case, is a 12 stack encoder-only Transformer architecture, with $d_{\text{model}} = 768$ and H = 12. It is pretrained using MLM and next sentence prediction (NSP) and is allowed to attend to tokens in both directions. It uses a tokeniser called WordPiece to format up to two sentences into tokens. Each token is labeled with a corresponding token ID, defined by the tokenisers vocabulary, v. This vocabulary is learned on separate training data and in BERT consists of about 30k tokens. The first token is always the [CLS] token, which represents the sentence embedding for classification tasks, while a [SEP] token splits sentences. Next, each token is projected into its high-dimensional embedding, which begin as static representations found in a dictionary, $\text{D} \in \mathbb{R}^{v \times d_{\text{model}}}$, learned during pretraining. This is summed with learned positional and segment embeddings and fed into the first encoder block. Contextual embeddings are then extracted through the self attention layers.

RoBERTa: RoBERTa [5] uses BPE instead of WordPiece, which uses a combination of subword and byte-level tokens to make more efficient use of its vocabulary size. It is trained on 10x the data, with longer sequences and bigger batches, and gets rid of the NSP task in BERT's pretraining. It generally outperforms BERT on NLP benchmark datasets and is therefore used in our report over BERT, which is included solely for comparison's sake.

SBERT: SBERT [6] is a modification of BERT, specifically for modeling embeddings. It is a sentence transformer that tackles the lack of semantic sentence embeddings in BERT through a siamese architecture, where two sentences run through an identical BERT model in parallel, as seen in Figure 2. The token embeddings are average pooled, and the cosine similarity is calculated and optimised using MSE over true similarity labels.

2.2 Sentence Embeddings

Encoder-only Transformer embedding spaces suffer from anisotropy [7]. Consequently, the sentence representations fail to proportionately capture sentiment differences. We look to sentence transformers to remedy this issue.

Although sentence transformers are generally trained for extracting semantics, the context of sentence similarity is arbitrary. Sentiment can very well replace semantics. This would leverage the SBERT architecture to map a sentiment embedding space, where sentences with similar sentiment are closer than those with differing. This is accomplished by optimising cosine similarity loss,

$$\left|\left| y - \frac{e_n \cdot e_m}{||e_n||_2||e_m||_2} \right|\right|_2,$$

using a similarity proxy to define the new label y over sentence embedding pairs e_n, e_m.

Let $s_1, s_2 \in [-1, 1]$ represent the sentiment scores of two distinct sentences. By computing $1 - |s_1 - s_2|$, we obtain a value in the cosine similariy range [-1, 1] that can serve as a proxy for the cosine of the angle between sentence embeddings. Cosine similarity loss will ensure that points closer in sentiment point in a similar direction, while points of opposing sentiment, e.g. -1 and 1, point in opposite directions, e.g. cosine similarity of -1. This is what we referred to as fanning the embedding space, depicted in Figure 3. We leverage the all-mpnet-base-v2 model for this training—a pretrained sentence embedding model fine-tuned using MPNet [9], a BERT-like architecture that uses a different pretraining strategy than MLM. This is done as its embedding space already establishes some structure, albeit semantic in nature.

Training this model is not simple due to the risk of overfitting between the embedding and regression training. We perform two-stage hold-out validation, since training and validating on the same data used for the embedding and dense layers risks overfitting. First, we split the original training set into an 80/20 train-validation split and create a new dataset containing sentence pairs (referred to as the new sets). Inspired by the Hugging Face SetFit library [10], we pair each sentence in the new training

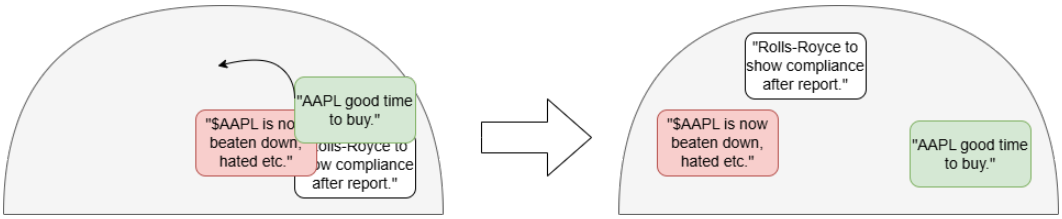

Figure 3: Visualisation of embedding space fanning. Pushes embeddings away based on sentiment.

set with "n" other samples from within the same set, without replacement. For the new validation set, however, we fix the number of paired samples at 30. We treat n as a hyperparameter and tune it using Bayesian optimisation with an early stopping patience of two. We also search for optimal learning rate and training epochs over the Pearson cosine score. During experimentation, our new validation Pearson cosine score was consistently higher when using 100 warmup steps, so we preset this value to save time.

At the end, we train our optimal model only on the original training set, not the original validation. Otherwise, the embedding space may overfit to noise in the full training data required for the dense layer, resulting in poor generalisation to the test set.

2.2.1 Regressive Dense Layer Training

The second hold-out validation stage involves a standard machine learning approach, where we train a dense neural layer with these fine-tuned embeddings, tuning hyperparameters on the true validation set. We conduct this tuning with a common framework: random search followed by grid search. The random search shortlists well performing hyperparameters, then we harshly shortlist the best performing ones for a final grid search. We purposefully keep the number of trials as low as possible, however, to avoid the validation error drifting too far from the generalisation error in the PAC bound.

Finally, we train on the validation and training set for the number of epochs observed in the hyperparameter search, scaled by the ratio of the validation set size to the train set size. All our model testing is conducted 20 times over random seeds, giving us valuable statistics for the MSE and R^2. Note, we did not perform cross validation due to time and computation constraints.

2.3 Baselines

We use two non LLMs as baselines:

- **Mean:** Find the mean label value on the train and validation data. Then, calculate the MSE over the test data.
- **XGBoost:** An optimised library for gradient boosted decision trees. It sequentially constructs weak learners that are trained on the residuals of the previous trees and scales them using a learning rate parameter. We approximate the generalisation loss using a bag-of-trigrams representation.

We also compare our model with BERT, a non-finetuned all-mpnet-base-v2, RoBERTa, both our rendition and [4]'s, and k-RoBERTa.

2.4 Datasets

The data we use in this paper comes from the FiQA 2018 Task 1 challenge (`https://sites.google.com/view/fiqa/home`). It consists of microblog messages, news statements and news headlines, with their sentiment labeled on a continuous range from [-1, 1]. We import this dataset from the Hugging Face library "datasets." This dataset is already split into train, validation and test sets of size 770, 111, and 230, resectively (70/10/20 split). We shuffle this data and remove all the duplicate rows, ensuring there is no data leakage. We then plot the histogram of the labels, observing a negatively skewed distribution, with a mean label of 0.1222 and a mode in the 0.5 range.

Table 1: Test MSE and R^2 (mean $\pm$ std) after 20 seeded runs (apart from *). The std of models from [4] (†) are calculated using 10-fold CV, therefore will be higher and should not be compared.

Model	**MSE**	**R^2**
Mean	0.1725 ± 0.0000	-0.0216 ± 0.0000
XGBoost	0.1178 ± 0.0012	0.3025 ± 0.0072
RoBERTa	0.0558 ± 0.0037	0.6693 ± 0.0218
BERT	0.0852 ± 0.0056	0.4956 ± 0.0330
RoBERTa†	0.0548 ± 0.0173*	0.6770
k-RoBERTa†	0.0490 ± 0.0185*	0.7110
Base Embedding	0.1066 ± 0.0039	0.3350 ± 0.0262
Embedding	$\mathbf{0.0483 \pm 0.0014}$	$\mathbf{0.7073 \pm 0.0073}$

3 Results

As Table 1 shows, our method of fanning the embedding space outperforms all other models on our test set. This includes a 1.4% improvement in the average MSE over k-RoBERTa's 10-fold cross validation average. We achieve this with a comparatively much simpler model. We also display the 95% confidence intervals for our model's MSE and R^2, shown in Figure 4, calculated using a t-distribution with $\nu = 19$. We get intervals of (0.0476, 0.0489) and (0.7039, 0.7107) for the MSE and R^2, respectively. Both confidence intervals fail to overlap with the k-RoBERTa mean MSE and R^2 . However, while our MSE shows better performance, our R^2 is worse. Our model vastly outperforms our baseline models, seeing a **3.57**$\times$ improvement compared to the naive mean method, and a **2.44**$\times$ improvement compared to the popular "xgboost" library.

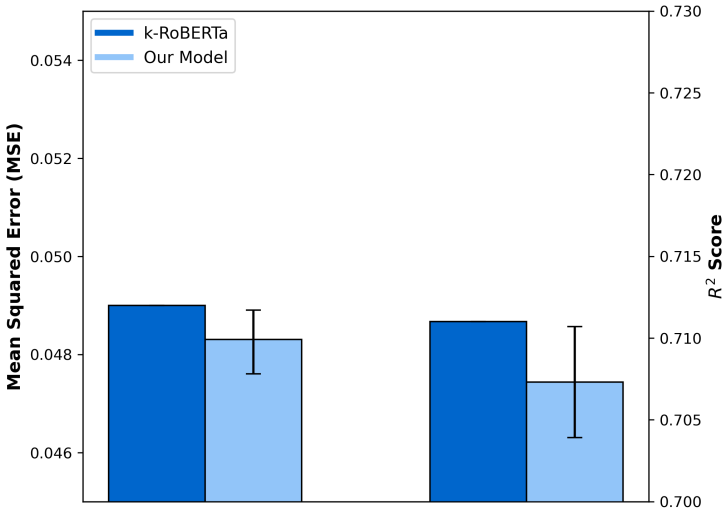

Figure 4: Bar chart for mean MSE (left) and mean R^2 (right) with confidence intervals.

We see that fine-tuning our model greatly improved our average MSE over the base embedding model. This mapping of the sentiment space, consequently an improvement in anisotropy, is exhibited in the average cosine similarity of the sentence embeddings. We observe an exceptionally high average cosine similarity of 0.8613 for the base RoBERTa model, compared to a much more uniform embedding space in our model, with an average similarity of 0.5086. For the base RoBERTa model, sentence embeddings were obtained by average-pooling token embeddings.

Notably, the base RoBERTa model—both ours and [4]'s—achieves a low average MSE of 0.0558 and 0.0548, respectively. However, it does not remain robust to random initialisations, similar to all other models—except for our embedding model. Despite this strong performance, RoBERTa's MSE remains higher than our embedding model's MSE.

4 Discussion

The cosine similarity of 0.5086 in our embedding model, compared to 0.8613 in the base RoBERTa model, demonstrates vast reduction in anisotropy, showing a more evenly distributed embedding space. We successfully fan the embedding space and achieve SOTA performance, the lowest average MSE, under hold-out validation testing. Consequently, our results support the idea that framing our problem as a sentiment contrastive learning problem, and sequentially constructing our model, improves performance on our sentiment regression task.

The lower average MSE, together with non-overlapping confidence intervals, suggests our model outperforms k-RoBERTa on the test set, robustly to random initialisations. Nonetheless, the non-overlapping R^2 confidence intervals, below k-RoBERTa's average, indicate that our model fails to provide the same proportional improvement in explaining variance as k-RoBERTa. This could be due to lower sample variance in our test set compared to the average test set. That is, the total sum of squares is smaller, and therefore a lower MSE is required to explain the same proportion of variance as in the case of k-RoBERTa.

We believe this success can be attributed to two interrelated reasons: (1) our method directly optimises for labeling group bias (2) embedding spacing acts as a form of feature engineering, guiding a more accurate search through the large dense layer representation space.

4.1 Label Bias

Labeling sentiment in finance is non-trivial, especially for $y \in [-1, 1]$. The data providers have not disclosed the methodology used to assign sentiment scores. Regardless, these sentiment labels represent some form of bias, whether collective or singular. The new embedding space directly captures this bias, as the similarity proxy is computed using these exact labels. Recall,

$$1 - |s_n - s_m|, \text{ where } s_n, s_m \text{ come from FiQA labels.}$$

By optimising the cosine similarity loss on this proxy, our embedding space effectively aligns with the labeling bias. Therefore, if the test set is labeled with the same priors, it should consistently generalise in the embedding space.

In contrast, k-RoBERTa uses general and financial lexicons as a means to inject sentiment knowledge. Its knowledge embeddings contain 25 sentiment scores for each word in the lexicon, each representing the bias of a different labeling group. While these lexicon-based representations capture useful sentiment information, they perform worse than our model with fewer parameters. With such a granular task, these bias differences could be more pronounced.

4.2 Feature Engineering

Dense neural layers have immensely large representation spaces. While known for not requiring feature engineering, carefully crafted features could aid with the representation search, as gradient descent only takes locally optimal steps. Naturally, this new embedding representation has been engineered such that small deviations in the embedding space directly represent small shifts in sentence sentiment. This makes the task more intuitive for the user and for the interpolating dense layers. The dense layers now have to learn simpler, consistent mappings, unlike RoBERTa's pooled and anisotropic embeddings, where small embedding shifts do not necessarily reflect similar labels.

4.3 Robustness

Although our predicted labels seem to be robust to random model initialisations, as seen by the comparably low MSE and R^2 standard deviations, we would need to conduct cross validation to assess its robustness across different test sets. Due to computational and time constraints, we were unable to perform 10-fold cross validation over the data and instead relied on a standard 20% hold out

test set. Given the relatively small sample size of 1111, there is potential for high variance in test set MSE, raising uncertainty whether our test set is truly representative of the population. Given enough computational resources, we would want to perform 10-fold cross validation over the embedding model and dense layers together. We would train and select the hyperparameters all in one go.

We believe our model has strong potential to remain the SOTA. Before conducting any hyperparameter tuning, our validation error—10% of the collective data—already exhibited a surprisingly low MSE in the 0.032-0.035 range. This displays the potential of our method to generalise over multiple test sets, but also that test set MSE standard deviation will be high over cross validation, likely comparable to k-RoBERTa's, at 0.0185. This implies cross validation is necessary to better assess general performance.

4.4 Limitations & Further Research

To note, the partitioning of the training sets in the cross validation approach is not straightforward, and model performance heavily depends on the specific sentence pairs generated. We suspect tuning the number of sentence pairs generated as a hyperparameter again would be adequate, although alternative approaches could be explored. For example, we investigated whether performance is improved if each sentence is paired with a set amount of other sentences in different label bins. This vastly underperformed over random sampling on the validation set, hence was omitted. We believe this was due to overfitting the embedding space to the training data.

Furthermore, one would have to be careful in excluding the domain-specific vocabulary. Our method is incapable of generalising for sentences including financial phraseology absent in the training data. In contrast, k-RoBERTa has access to multiple financial lexicons, which could act as a type of "safety net," giving better generalisations. We suggest exploring using our improved embedding model together with the lexicons as used by [4]. These embedding models could be different to our all-mpnet-base-v2.

Finally, even when accounting for labeling bias, there is likely a high lower bound to the test set MSE, in relation to the scale. Even within the same labeling group, there could be labeling inconsistencies due to granularity. For example, it is unclear how to differentiate between sentences of sentiment scores 0.134 and 0.137.

5 Conclusion

We leveraged the pretrained all-mpnet-base-v2 embedding model, fine-tuned on a cosine similarity proxy, to map out an embedding space directly related to labeled sentiments. This acted as a form of feature engineering, defining the problem explicitly as a biased sentiment problem for the dense layers. Our approach achieved SOTA results on hold-out validation testing, although a higher computational budget is needed to make this statement generally.

Our simple framework can be translated into any sentiment problem with labels defined in [-1, 1]. Such bias-optimised predictions could prove hybrid lexicon methods obsolete, given a population representative dataset. If the production data is drawn from the same latent distribution, this representative dataset could rid the need for a safety net, and a more accurate and intuitive approach can be implemented.

References

[1] Kelvin Du, Frank Xing, Rui Mao, and Erik Cambria. Financial Sentiment Analysis: Techniques and Applications. *ACM Computing Surveys*, 56(9):1–42, October 2024.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. arXiv:1810.04805.

[3] Dogu Araci. FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, 2019. arXiv:1908.10063.

[4] Kelvin Du, Frank Xing, and Erik Cambria. Incorporating Multiple Knowledge Sources for Targeted Aspect-based Financial Sentiment Analysis. *ACM Transactions on Management Information Systems*, 14(3):1–24, September 2023.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, July 2019. arXiv:1907.11692.

[6] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, August 2019. arXiv:1908.10084.

[7] Kawin Ethayarajh. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings, September 2019. arXiv:1909.00512.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, August 2023. arXiv:1706.03762.

[9] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. MPNet: Masked and Permuted Pre-training for Language Understanding, November 2020. arXiv:2004.09297.

[10] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. Efficient Few-Shot Learning Without Prompts, September 2022. arXiv:2209.11055.