Beyond Bert-based Financial Sentimental Classification: Label Noise and Company Information

Lizhong Wang
NYU Center for Data Science
lw2350@nyu.edu

Hong Gong
NYU Center for Data Science
hg1153@nyu.edu

Yi Xu
NYU Center for Data Science
yx2090@nyu.edu

Abstract

In our work, we propose two methods to deal with label noise in training data for Fine-tuned BERT financial sentiment classification model. In addition, we also experiment on incorporating company information (i.e.: description of the main entity for a slop) into our BERT model, trying to provide a context information to our language model to further improve model performance. We will show in our results that two methods to deal with label noise do not show an extremely promising result, but incorporating company information into language model outperforms the baseline performance by a decent amount.

1 Introduction

Financial sentiment analysis has raised much attention since the commencement of deep learning. Along with the development of Natural Language Processing and extensive applications of transfer learning, it is a natural trend to apply state-of-the-arts language model, such as BERT **[1]**, to financial sentiment classification. However, for many classification tasks, lack of labeled data and existing label noise have always been two major challenges which hinder the model from further improving its performance **[4]**. In addition, the ultimate goal of sentiment analysis on financial text data is to predict how a piece of text information affects the movement of stock price for an entity **[6]**. Solely making a decision based on a language model seems not ideal and not accurate enough.

In our project, we built a financial sentiment classification model by fine-tuning pre-trained BERT model. Then, we mainly focus on further improving the model performance by (1) incorporating noisy labels into training data and (2) proposing an innovative method called "Company Embedding", which provides an extra company background information into the language model. In section 3, we show the details of our algorithm. In section 4, we present the experimental results showing that incorporating the noisy labels by inverting the noisy label and by including 'non-boundary' labels slightly improve the model performance. Also, adding company embedding into our language model by various ways consistently outperforms the baseline language model.

2 Related Work

Label noise has been an enduring problem for many classifier models. The consequences brought by the presence of label noise are the deterioration of model performance and increase in the necessary number of samples for learning. Traditionally, there are three ways to deal with label noise: design algorithm naturally robust to label noise, directly model label noise during learning and filter out noisy labels **[4]**. Although it has been shown that deep learning is comparatively more robust to label noise, model still suffers from performance deterioration when the ratio between correct and noisy

Preprint. Under review.

labels are not substantially high **[5]**. Considering the scarcity of labeled training data in the financial sentiment analysis task, dealing with label noise is necessary and unavoidable.

Different from traditional sentiment analysis task, the end goal for most financial sentiment analysis tasks is to predict the movement and volatility of stock market based on the sentence sentiment. In traditional machine learning field, **[6]** has given a comprehensive survey for sentiment and text analysis for financial text data. There have also been many attempts to use deep neural network and BERT-based language models to conduct sentiment analysis in finance domain **[7] [8] [9] [10]**. Due to lack of large labelled financial text data, it is hard to utilize the full potential of deep learning model to improve model performance. **[3]** has attempted to improve fine-tuned BERT model performance by further pre-training BERT on financial corpus, and it shows slight improvement compared with BERT-based model. To the best of our knowledge, no existing research papers have experimented on directly incorporating context information into language model for financial sentiment analysis.

3 Problem Definition and Algorithm

3.1 Task

We are working with Accern, a NYU start-up leveraging machine learning on non-traditional alternative financial data sets, such as public news, blogs, and twitter feeds. Accern currently is using a dictionary-based algorithm to calculate sentiment score for financial documents **[11]**. The limitation of dictionary based algorithm is that it fails to generalize to words not in dictionary and it cannot accurately capture the semantic meaning of sentences. Therefore, the first goal of this project is to build a financial sentiment classifier using state-of-the-arts NLP representation learning model, such as DistilBERT **[2]**.

However, if we want to build a deep learning model, the new problem is that we do not have enough labeled data. In addition, the data provided by Accern suffers from label noise. The sentiment labels are calculated by an algorithm developed by Accern. There are five classes for the labels as shown in table 1. Then the calculated labels are presented to human users via Accern's product. The users may choose to agree or disagree with the calculated label. If a label is confirmed by human users, then we consider this label as correct. Otherwise, we consider the label as incorrect, which is noisy labels in this case. As a result, 15% of labels are disagreed by users. Unfortunately, there is no extra information available for us to correct them. Therefore, the second stage of this project is to come up with some methods to deal with label noise.

Additionally, inspired by the decision making process of financial analysts when they are reading a piece of financial document—they need the context information of the main entity of this document to better judge the sentiment—we propose to incorporate the company information into our language model to further improve the model predictive power. We call this method Company Embedding.

Our project is composed of three stages. Firstly, we implemented a pipeline to fine-tune a Transformer-based NLP model on our dataset. Secondly, we came up with two ways to assimilate noisy data into our training set. Finally, we incorporated the company embedding into our language model by three ways. We experimented three pre-trained language models: BERT-base **[1]**, FinBERT **[3]**, and DistilBERT **[2]**. DistilBERT perform the second best among the three, only slightly worse than FinBERT. In addition, DistilBERT requires significantly less time for training. Therefore, we experimented on the effectiveness of tackling label noise and incorporating company embedding with DistilBERT since it requires less training time.

3.2 Algorithm

3.2.1 Task 1: Deal with Label Noise

As mentioned in Section 3.1, 15% of our data contains label noise. Our mentor and our team have come up with two methods to tackle label noise and incorporate noisy data into our training stage.

- Method 1: Invert the incorrect label to its opposite.
- Method 2: Use non-boundary labels as ground truth

Class	Representation	Score Range
0	strongly negative	-100 to -50
1	weakly negative	-50 to -10
2	neutral	-10 to 10
3	weakly positive	10 to 50
4	strongly positive	50 to 100

Table 1: Sentiment Scores Classification

The first method is to invert the incorrect labels (except for neutral labels). For example, if a sample is incorrectly labeled as strongly negative (class 0), then we treat this sample as if its true label is strongly positive (class 4). Neutral label can be inverted to two directions, either positive or negative, in order to avoid confusion and extra noise, we decided to discard the neutral labels when it's wrong (please refer to Algorithm 1 for detailed implementation). The intuition behind this method is that people are more likely to disagree with a label when the label is the opposite to their opinions.

Algorithm 1: Label Noise - Inverse Method

```
Input  : label ∈ {0, 1, 2, 3, 4}
Input  : indicator if label is correct
Output : denoised label

if label is correct then
    return label
else if label == 2 (neutral) then
    discard label
else
    return (4 - label)
```

Algorithm 2: Label Noise - Non-Boundary Method

```
Input  : label ∈ {0, 1, 2, 3, 4}
Input  : indicator if label is correct
Output : a list of possible labels

if label is correct then
    return label
else if label == 0 (strongly negative) then
    non_boundary_labels = [2, 3, 4]
    return non_boundary_labels
else if label == 1 (weakly negative) then
    non_boundary_labels = [3, 4]
    return non_boundary_labels
else if label == 2 (neutral) then
    non_boundary_labels = [0, 4]
    return non_boundary_labels
else if label == 3 (weakly positive) then
    non_boundary_labels = [0, 1]
    return non_boundary_labels
else if label == 4 (strongly positive) then
    non_boundary_labels = [0, 1, 2]
    return non_boundary_labels

After getting a list of possible_labels
if len(possible_labels)> 1 then
    explode possible_labels to row-like feature to one label data point
```

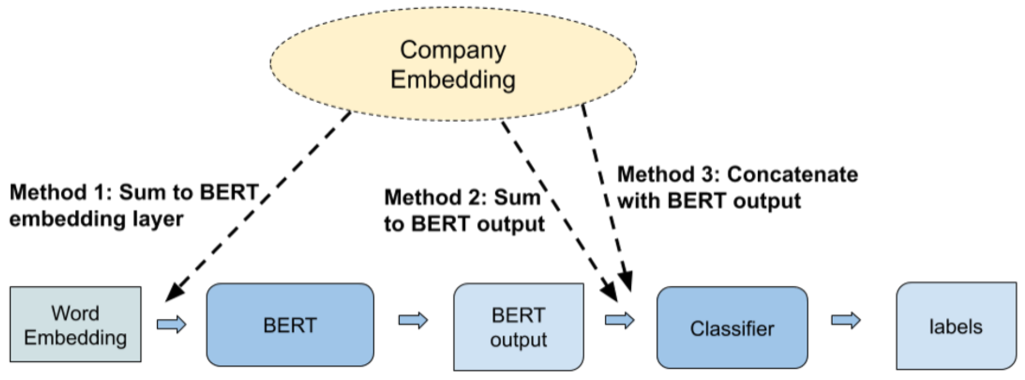
Company Embedding
Method 1: Sum to BERT embedding layer
Method 2: Sum to BERT output
Method 3: Concatenate with BERT output
Word Embedding
BERT
BERT output
Classifier
labels

Figure 1: Three methods to incorporate company information

The second method is called "Non-Boundary Method". We first define boundary labels as the classes adjacent to the calculated label class. For example, if the noisy label is 1 (weakly negative), then its non-boundary labels are 3 (weakly positive) and 4 (positive). Once we have this definition of boundary labels, we treat all non-boundary classes as possibly correct labels. Then we add a copy of the training sample for each possible label. By this way, we give model two chances to learn this sample. Algorithm 2 shows a detailed implementation for each incorrect label.

3.2.2 Task 2: Company Embedding with Language Model

In order to get a good representation of company information, we input a company's description to BERT's embedding layer to calculate the company embedding. The key point is that since company embedding is the output of BERT embedding layer, the company embedding has the same dimension as the BERT representation. In addition, each dimension of the company embedding has the same meaning as BERT representation. Therefore, it makes sense to directly sum company embedding to the hidden states of BERT model. We came up with three methods to incorporate company embedding to the language model as shown in Figure 1.

- Method 1: Sum company embedding vector to BERT embedding layer.
- Method 2: Sum company embedding vector to BERT output.
- Method 3: Concatenate company embedding vector to BERT output.

We show the full results of experimenting these three methods using DistilBERT as the language model in Section 4.2.3. Figure 1 shows the model architecture of incorporating company embedding.

4 Experimental Evaluation

4.1 Data

The data is provided by Accern. The original data set includes 157,710 observations and four features: 'id', 'slop', 'sentiment' and 'sentiment_correct'. 'id' refers to the identity number of financial documents. 'slop' contains the excerpts that summarise the main idea of each financial events, extracted by Accern's internal algorithm. To be noted, 'id' and 'slop' is not one-to-one, since it is likely that one financial document talks about multiple events. 'sentiment' is the sentiment score for each event, numerical values ranging from -100 to 100, which are calculated by Accern's proprietary algorithm. 'sentiment_correct' is the indicator to tell whether the sentiment class is agreed by users or not. If it is 0, it means the label given by Accern is wrong.

The first step of data pre-processing is to label each data point based on the corresponding sentiment score. The score splits data into 5 classes (please see table 1 for detailed classification rule). And this label is listed in the column 'label_from_score'. We also cleaned dataset by applying two label noise

slop	label_from_score	agreed_by_user	y_inverse	possible_non_boundary_labels	y_boundary
Slop 1	1	1	1	1	1
Slop 2	2	0	3	[3,4]	3
Slop 2	2	0	3	[3,4]	4
Slop 3	0	0	4	[2,3,4]	2
Slop 3	0	0	4	[2,3,4]	3
Slop 3	0	0	4	[2,3,4]	4
Slop 4	2	0	discard	[0,4]	0
Slop 4	2	0	discard	[0,4]	1

Table 2: peusdo clean data

methods mentioned in Section 3.2.1 (Algorithm 1 and Algorithm 2). The denoised labels are listed in columns 'y_inverse' and 'y_boundary' correspondingly to Algorithm 1 and Algorithm 2 in Table 2. Consequently, there are three types of data set variations: (1) containing only correctly labeled data, (2) a mixture of correctly labeled data and inverted noisy data (method 1), and (3) a mixture of correctly labeled data and non-boundary noisy data (method 2). For convenience, we call three data variations as Type 1, Type 2 and Type 3 in this work. All three data variations can be used as training set and validation set. Hence, there are 9 combination of training set and validation set. Here is the information of Type1, Type2 and Type 3 data distribution:

- # of Type1 training set: 70283; # of Type1 testing set: 7724
- # of Type2 training set: 78735; # of Type2 testing set: 8736
- # of Type3 training set: 96906; # of Type3 testing set: 9278

4.2 Methodology

4.2.1 Baseline Model

In this task, We take the DistilBERT model trained on only correctly labeled data as the baseline model. In terms of our hyperparameter choices, we use $batch\_size = 16$, $training\_epoch = 6$, $learning\_rate = 2e-5$,and $Bert\_sentence\_embedding\_length = 32$. We also use discriminative training strategy to avoid catastrophic forgetting **[12]**. When evaluating the effectiveness of label noise methods and company embedding, we use the same set of hyperparameters for all other models.

4.2.2 Tackle Label Noise

In order to check whether the two proposed methods for dealing with label noise in Section 3.2.1 are effective, we compare the model performance trained on Type 1 data set with the model performance trained on Type 2 and Type 3 data set.

4.2.3 Incorporate Company Embedding

We scraped text description of all U.S. listed companies from Yahoo Finance. We use BERT-base language model to extract a representation of companies to be the starting point of the embedding layer of our own model. One benefit of using BERT to calculate company embedding is that each dimension of company embedding has the same meaning as the hidden states of the language model, so that it makes sense to sum company embedding vector and language model output. One problem of using BERT is that the descriptions are too long to be fully processed by the the full BERT model. We solve this issue by inputting all tokens in the description to the BERT embedding layer, and we take the average of the BERT embedding output of all tokens as the representation of the company.

Because all the company embedding vectors are clustered in small region in the BERT representation space, we centered all company embeddings such that all companies have 0 mean. After we center the embedding, similar companies have a higher cosine similarity than other more different companies as shown in Figure 2. Once we obtained a good representation of company embedding, we use it as the initial weights of the company embedding layer in our own model. The company embedding layer is updated during training.

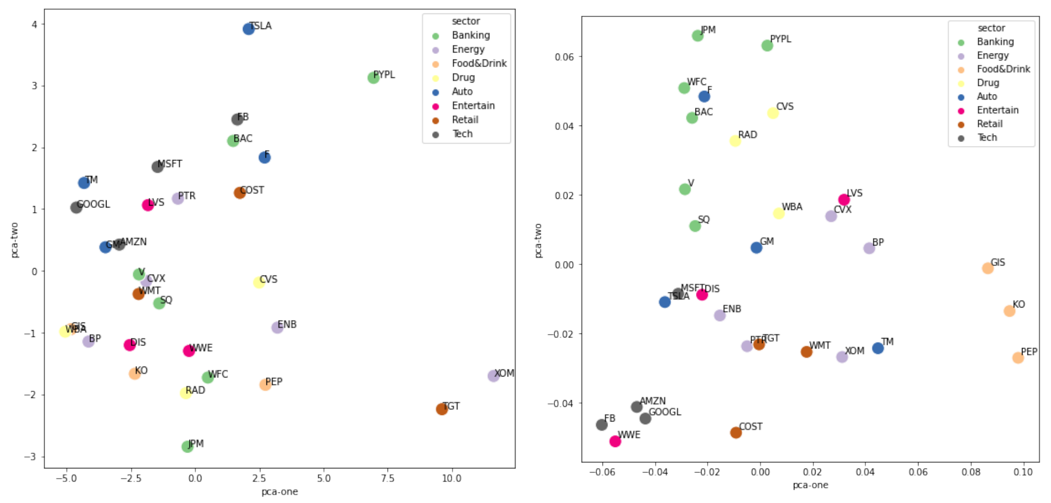

Figure 2: Company Embedding Visualization. Company embedding vector is obtained by taking the average of BERT embedding output of all tokens in the company description. These two images are visualization of company embedding of serval companies using PCA; different color refers to different industry. The image on the left side shows uncentered company embedding vectors, while the image on the right side shows centered company embedding vectors. We can clearly see that centered company embedding is a better representation than uncentered company embedding in terms of differentiate different companies.

Predicted Label	Label from Score	Sentiment Correct	Boundary Accuracy Calculation
3	1	True	Prediction False
3	3	True	Prediction True
3	2	True	Prediction Discard
3	1	False	Prediction True
3	3	False	Prediction False
3	2	False	Prediction Discard

Table 3: Boundary Accuracy Examples

4.2.4 Evaluation

For Type 1 and Type 2 test set, we use accuracy as the primary evaluation criteria, with confusion matrix and AUC on each class as the auxiliary criteria. For the Type 3 test set, we use our self-defined boundary accuracy as the criteria. Boundary accuracy is defined as follows: we throw away the near-correct predictions, which is independent of whether the ground truth is actually true. For instance, if the ground truth is class 2, then predicted class 1 and 3 won't be counted into accuracy calculation no matter class 2 is actually ground truth or not. When class 2 is actually true, prediction of class 0 or 4 is considered as wrong. When class 2 is disagreed by experts, namely noisy label, prediction of either class 0 or 4 is considered as correct (Please see Table 3 for more examples). Based on this definition, we also take the number of dropped test data into consideration. As a result, we want to make the boundary accuracy as high as possible and in the meanwhile maintaining the dropped data amount and percentage as low as possible.

4.3 Results

The results of different label noise methods show an interesting pattern: for one type of testing set, the best model is the model trained on the corresponding type of training set. For example, when tested on Type 1 testing set, the model trained on Type 1 training data yields the best performance. However, when tested on Type 1 and Type 2 testing sets, the performance of the three models are not as large as the difference of the three models tested on Type 3 testing set. However, it is worth noting that the model trained on Type 3 training set gives the best result and the its boundary accuracy is

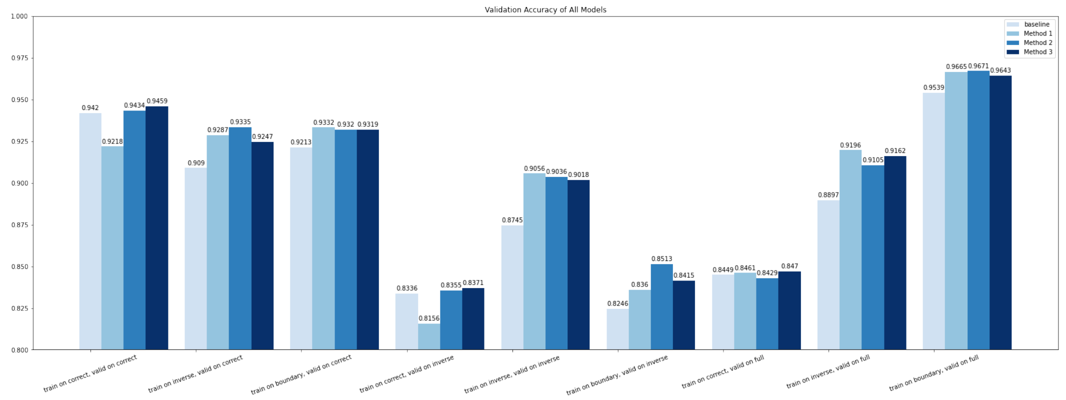

Figure 3: Accuracy of all Model Variations

Model Name	Boundary Accuracy	# of Dropped Samples	% of Dropped Sample
DistillBert_Correct	0.8449	638	6.88%
DistillBert_Correct and Inverted Data	0.8897	656	7.07%
DistillBert_Correct and Non-boundary Data	0.9539	458	4.94%

Table 4: Baseline Performance of DistillBert tested on Full Dataset

much higher than those trained on the other two training sets (please refer to Table 4 and Table 5 for more details).

When it comes to company embedding performance, we evaluated baseline model along with 3 company embedding models on 9 train and validation combinations. Therefore, we tested 36 model variations in total. Their accuracy results are visualized in the Figure 3. In each group, 4 bars are displayed from left to right in the sequence of baseline, company embedding method 1, method 2 and method 3.

By comparing company embedding model with the baseline, company embedding models always obtain higher accuracy than baseline in all train-val groups. For example, in the last group, all three company embedding models perform better than baseline. In the first group, despite that company embedding method 1 doesn't perform well, method 2 and method 3 still outperform the baseline model. It shows that adding extra company information boost the language model's predictive power for financial sentiment analysis task.

Furthermore, the best company embedding method varies from one model to the other. However, Method 3 always performs better than baseline, whereas Method 1 and Method 2 don't work well all the time.

4.4 Discussion

From Section 4.3, we observe that dealing with label noise using either inverting noisy labels or using non-boundary noisy labels do not improve our model performance by much. Specifically, it is a general pattern that the model performance is the best for one type of test set which is trained on

Model Name	Test Type	Accuracy
DistillBert_Correct	Correct Only	0.9420
DistillBert_Correct_Inverted Data	Correct Only	0.9090
DistillBert_Correct_Non-boundary Data	Correct Only	0.9213
DistillBert_Correct	Correct and Inverted	0.8336
DistillBert_Correct_Inverted Data	Correct and Inverted	0.8745
DistillBert_Correct_Non-boundary Data	Correct and Inverted	0.8246

Table 5: Baseline Performance of DistillBert tested on Correct and Mixture Dataset

its corresponding training set. However, we treat the boundary accuracy on our full test set as the most important evaluation metrics because this test set does not throw away any samples. From this perspective, the model trained on noisy label using non-boundary method outperforms the other two models by a decent amount shown in Table 4.

For models with company embedding, they clearly outperform our baseline models. Although there is no clear winner among three methods to incorporate company embedding, it is a clear trend that after adding company embedding information into our language model, at least one method will outperform the baseline result. It exemplifies the effectiveness of adding context information into language model in terms of improving an already high classification accuracy.

5 Conclusions

In our work, we discuss the current development of financial sentiment analysis task. On top of the previous work, we propose two methods to deal with label noise and adding company information into our language model to further improve our model performance. We demonstrate that although two methods to deal with label noise does not improve model performance by much, adding context information, namely adding company embedding into language model enhances model performance. There are still many aspects that we could improve and experiment further. Firstly, given our conclusion that context information improves the model performance, if time permits, it is possible to incorporate more of meaningful text form context information into our language model. Secondly, for now company embedding only contains the simplest description of the company information. We hypothesis that if we can have both the historical description of company and the most up-to-date description of company, we could potentially make a better use of our company embedding. Finally, although company embedding generates promising result, we still lack the theoretical and mathematical foundations to explain why it works and to what extent it can further improve the language model.

6 Lessons learned

Throughout this project, all of us have learned the theories, implementation and application of BERT. In addition, we have taken on the challenges to deal with label noises and invented company embedding to further improve an already high baseline accuracy. Last but not the least, the collaboration, communication and organization skills we learned for this project will pave the way for the future success of our data scientist careers.

References

[1] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

[2] Sanh, Victor, et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter." arXiv preprint arXiv:1910.01108 (2019).

[3] Araci, Dogu. "Finbert: Financial sentiment analysis with pre-trained language models." arXiv preprint arXiv:1908.10063 (2019).

[4] Frénay, Benoît, and Michel Verleysen. "Classification in the presence of label noise: a survey." IEEE transactions on neural networks and learning systems 25.5 (2013): 845-869.

[5] Rolnick, David, et al. "Deep learning is robust to massive label noise." arXiv preprint arXiv:1705.10694 (2017).

[6] Kearney, Colm, and Sha Liu. "Textual sentiment in finance: A survey of methods and models." International Review of Financial Analysis 33 (2014): 171-185.

[7] Day, Min-Yuh, and Chia-Chou Lee. "Deep learning for financial sentiment analysis on finance news providers." 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2016.

[8] Hiew, Joshua Zoen Git, et al. "BERT-based Financial Sentiment Index and LSTM-based Stock Return Predictability." arXiv preprint arXiv:1906.09024 (2019).

[9] Sousa, Matheus Gomes, et al. "BERT for Stock Market Sentiment Analysis." 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2019.

[10] Lee, Chien-Cheng, Zhongjian Gao, and Chun-Li Tsai. "BERT-Based Stock Market Sentiment Analysis." 2020 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan). IEEE, 2020.

[11] Loughran, Tim, and Bill McDonald. "When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks." The Journal of Finance 66.1 (2011): 35-65.

[12] Howard, Jeremy, and Sebastian Ruder. "Universal language model fine-tuning for text classification." arXiv preprint arXiv:1801.06146 (2018).

Student Contributions

Lizhong Wang:
Data Cleaning and Splitting
Built Model Training and Validation Pipeline
Organized language model and company embedding code
Company Embedding Visualization
Label noise and Company Embedding Model Experiments
Writing Report

Hong Gong:
Baseline model implementation
Scrap Company Embedding from Yahoo Finance
Company Embedding Model Experiments
Data distribution visualization
Evaluation Metrics coding
Report writing

Yi Xu:
Data Cleaning and Splitting
Label noise and Company Embedding Model Experiments
Data Visualization
Writing Report