CYUT at the NTCIR-16 FinNum-3 Task: Data Resampling and Data Augmentation by Generation

Xie-Sheng Hong
s11027602@gm.cyut.edu.tw
Department of CSIE
Chaoyang University of Technology
Taichung, Taiwan

Jia-Jun Lee
s11027603@gm.cyut.edu.tw
Department of CSIE
Chaoyang University of Technology
Taichung, Taiwan

Shih-Hung Wu*
shwu@cyut.edu.tw
Department of CSIE
Chaoyang University of Technology
Taichung, Taiwan

Mike Tian-Jian Jiang
tmjiang@gmail.com
Zeals Co, Ltd
Tokyo, Japan

ABSTRACT

This paper presents a description for our submission to the NTCIR-16-FinNum-3 shared task in fine-grained claim detection for finacial documents. We submitted 3 runs in both the Analyst's Report (Chinese data) and Earnings Conference call (English data) sections in the final test. The CYUT-1 uses MacBERT (for Analyst's Report) and Roberta (for Earnings Conference call) with the classical classifier BiLSTM as the baseline of this study. In CYUT-2, we change the classifier to AWD-LSTM for comparison. Furthermore, considering the the problem of unbalanced training data when training the model, we use data resampling technique in both CYUT-1 and CYUT-2. And we propose an attempt to extend the data using GPT2 in CYUT-3. In the Macro-F1 validation, the 3 Runs obtained 88.80%, 86.76%, and 88.20% in the Analyst's Report, and 85.53%, 87.49%, and 87.88% in the Earnings Conference call, respectively. Also, after the formal runs were submitted, we did several additional runs to test the validity of our original proposed methods by conducting more oriented attempts. Furthermore, we found that data augumentation has improved the prediction of Analyst's Report in our experiment. Marco-F1 can improve about 2% in additional runs, maximum to 90.24%.

CCS CONCEPTS

• **Information systems** → **Information extraction**.

KEYWORDS

Financial documents, Fine-grained claim detection, MacBERT, RoBERTa, BiLSTM, Data Resampling, Data augmentation

TEAM NAME

CYUT

SUBTASKS

FinNum-3:Investor's and Manager's Fine-grained Claim Detection

*Contact author

1 INTRODUCTION

FinNum is a series of shared tasks focused on the understanding of financial numbers. The aim is to demonstrate the importance of understanding numbers in finance-related descriptions and to find their possible applications. In the previous FinNum-1 and FinNum-2, the focus was mainly on data from financial social media[2]. In the NTCIR-16 FinNum-3, the target of analysis is shifted to two official documents, professional analyst's report and earnings conference call. The organizers propose a task called fine-grained claim detection. The objective of the task is to detect whether the numbers described by investors and professional managers in the 2 official documents are "claims" or only objective figures[1].

In particular, the task is to predict whether a given number is a "claim" in a given description. Therefore, this task can be defined as a binary classification task. In the task, we proposed a total of 3 methods, all based on MacBERT or RoBERTa, and tried data augmentation methods to solve the problems that may be caused by imbalance in training data. We also tried several other methods in subsequent additional runs to verify the effectiveness of our attempts in various aspects.

In the following section of this paper, we describe our proposed method in detail, show the relevant settings of our experiments and the results of our experiments, and analyze the experimental results. In the last paragraph, we present our conclusions and future work.

2 METHOD

The method of our formal runs are designed as follows: the first is the combination of MacBERT[3] (for Analyst's Report task) and RoBERTa[9](for Earnings Conference call task) with the traditional BiLSTM[7] classifier which is also used for our Baseline. For comparison, we replaced the BiLSTM with AWD-LSTM[11] in the second Runs to test the difference between the two. Also, considering the problem of unbalanced training data, we use data resampling technique in both Runs to balance the data set. In the third Runs, we use the first attempt as the basis, but do not use data resampling. Instead, we use GPT-2[16] to generate data to populate the training data in order to try to improve the imbalance of the training data in a different way.

Table 1: Number of Data and Distribution Ratio

Data	In-claim	Out-claim	Distribution
Analyst's Report (Chinese)	999	3220	1:0.3
Earnings Conference call (English)	1039	7298	1:0.14

2.1 BERT, RoBERTa, and MacBERT

The 3 proposed Runs are based on two pre-training models, RoBERTa (for Earnings Conference call task) and MacBERT (for Analyst's Report task) , which are both variants of BERT. The full name of BERT is Bidirectional Encoder Representations from Transformers, which is a two-way unsupervised, transformer-based language representation model proposed by Google. It is mainly pre-trained using Masked Language Model (MLM) and Next Sentence Prediction (NSP). Different from word2vector[13] and GloVe[15] that do not use context, BERT is able to use the context of the text for inference, which makes its performance in various tasks superior[5].

RoBERTa (Robustly optimized BERT approach), an enhanced optimization variant of the original BERT, was jointly proposed by Facebook Inc. and University of Washington. RoBERTa uses more model parameters, larger batch size for training, and increases the size of the training data. Also, dynamic masks are used in the training, it allows the model to generate a new mask pattern for each input sequence, making it possible to gradually adapt to different mask patterns as data is input[9].

MacBERT (MLM as correction BERT) is an improved variant of BERT jointly proposed by iFLYTEK Research and SCIR for the Chinese language domain. It improves the MLM pre-training task used in the original BERT by overwriting the original single words with similar words to perform the MLM task, thus reducing the difference between pre-training and fine-tuning. In addition, some modifications were made to the original BERT training tasks, including the use of Whole Word Masking(WWM) and N-gram masking. Hence, the performance of the pre-training model in the Chinese language domain has been improved to a certain extent compared with the original BERT[3, 4].

In this study, our system is mainly based on the base versions of MacBERT and RoBERTa, using a machine learning framework called pytorch[14] and calling Huggingface[19] for model construction and training.

2.2 BiLSTM and AWD-LSTM

In our model architecture, different kinds of LSTMs are used as the classifier after the output of BERT. They are BiLSTM (Bi-directional Long Short-Term Memory) and AWD-LSTM (ASGD Weight-Dropped LSTM).

LSTM (Long Short-Term Memory) is an improved version of RNN (Recurrent Neural Network), which solves the problem of long-term dependency and possible gradient disappearance or explosion of RNN. It learns and forgets information through the Gate architecture, making it more capable of processing long sequence data[6]. Moreover, our system uses Bidirectional LSTM, which allows the LSTM to take into account the order of the inputs.

The AWD-LSTM (ASGD Weight-Dropped LSTM) is a variant of the LSTM, which, as the name suggests, is a weight-dropped LSTM. Its authors used the DropConnect method[18], based on the original LSTM, to drop part of the data of the weight matrix between the hidden states in the LSTM. This prevents the overfitting problem of traditional LSTM and minimizes the effect on the training speed. Moreover, the authors used NT-ASGD, a variant of Average -SGD, as an optimizer with a fixed learning rate parameter, so that no manual adjustment is required and its performance is better than SGD[11, 12]. Therefore, after referring to the various improvements and effectiveness of LSTM in the authors' paper, we decided to apply it to the system as an another attempt of classifier, hoping that it can outperform the traditional LSTM.

2.3 Data Augmentation Techniques

From Table1, we can see the number and distribution ratio of the two labels, 0 (out claim) and 1 (in claim), in the current data set. Considering the imbalance in the data given to the model for training. We believe that if the model is trained with these data directly, the trained model may be biased toward the answer with more data when making predictions. Therefore, we have tried to solve this problem by data augmentation. We have used two approaches to try this, namely, data resampling or using GPT-2 to generate additional data.

2.3.1 Random Resampler. The first is random resampler. In CYUT-1 and CYUT-2, we use random resampling to try to solve the data imbalance problem. The first is random resampler. In CYUT-1 and CYUT-2, we use random resampling to try to solve the data imbalance problem. The random resampler is to select the data that have already been selected again when constructing the training data set. In this way, a smaller number of data are re-sampled in order to achieve the goal of having a similar number of training data for the two predicted targets in model training. In this work, we conducted several tests and experiments using the sampling ratio as the adjustment variable. The model is trained by sampling a specific ratio of data with labels 0 and 1 (out claim and in claim) and repeating the sampling after a single training round. We try to find the best sampling ratio to train the model. After testing and experimenting, we found that resampling all data is the best in the experimental results. That is, we use all data with label 0 (out claim) and resample data with label 1 (in claim) until the ratio of the two data is 1:1.

2.3.2 GPT-2 Data Generate. In CYUT-3, we use GPT-2 for data augmentation. GPT-2 (Generative Pre-trained Transformer 2) is a huge pre-trained language model derived from the Decoder in Transformer. It can predict the next most likely word based on all words in the given text. It can predict the next most likely word based on all words in the given text. Since this model uses a large amount of cross-domain data in training, it can be used for a wide

range of tasks. The main one is used in the generation of the text that we are trying this time[16].

For the GPT-2 model selection, we used the Chinese pre-training model, CLUECorpussmall[21] for the Analyst's Report data, which is a GPT-2 Chinese pre-training model trained from the CLUECorpus2020 Chinese corpus[20]. For the Earnings Conference call data, we used the original GPT-2 model[16] without additional training.

The following is a simple example of our text generation. We use a very intuitive way to generate text with GPT-2: we input a fixed text in Chinese or English with a random number into GPT-2 and let it generate a random text data automatically. We start the Chinese text with " 我們推測會上升 X" (We speculate that it will rise X), and fix the final generated length to 100. The English text starts with " We anticipate a X increase" and the fixed length is 50. The X in both texts represents a random number, in the example below it is 98.02 and 149.

Input:
我們推測會上升 98.02
Output:
我們推測會上升 98.02%，明天早晨大跌...
Input:
We anticipate a 149 increase
Output:
We anticipate a 149 increase in the number of cases with...

In this way, the lable of the text generated by GPT-2 must be in-claim.

3 EXPERIMENTS

The overall experimental flow of our system in this study is shown in Figure1. After the data is input into the system, it is firstly preprocessed and augmented. Then, we create a dataloader for train, valiation and test for follow-up model training. Then we build a neural network model based on MacBERT/RoBERTa and train it. After training, the test data are passed to the model for prediction and exported into a specified format.

In this section and the next sections, we describe in detail the parameters of our system and how the test experiment was designed. The experimental and validation results of our system are presented, and we analyze the results to identify errors and possible improvements.

3.1 Parameters and Setting

In our formal runs, most of the hyperparameter settings are the same for both subtasks. The detailed numerical settings can be found in Table 2. First, in order to use the information provided by the training text as much as possible, we set the maximum sequence length to the maximum 512 that BERT/RoBERTa can use. Considering the large amount of text used for training, CUDA out of memory problem may occur during model training. We therefore set the batch size to a smaller degree of 4. For the choice of model optimizer, we use AdamW[10], a variant of the classic optimizer Adam[8], it improves the problem of Adam in the weight decay. In addition, with reference to the description in [17], we choose a low learning rate to train the model, so the learning rate is set to "2e-5". Finally, it is worth noting that we originally used the linear format for the learning rate schedule. However, the training was not effective. After several different experiments, we found that switching to a cosine schedule was effective in improving the results of our model. Therefore, we use the cosine schedule[1] to dynamically warm up and adjust the learning rate.

Table 2: Parameters Value of Model

Parameters	Values
BERT Model	macbert-base or roberta-base
Batch size	4 or 8
Max length	512
Optimizer	AdamW
Learning rate	2e-5

3.2 Formal Run

3.2.1 CYUT-1: BiLSTM. CYUT-1 is our baseline for this task, which uses MacBERT or RoBERTa with the classical classifier BiLSTM to perform classification prediction. The main structure of the model is to add the LSTM layer after the output of BERT layer, and then to stack the model layers twice in the order of linear layer, activation layer and Dropout layer(with p=0.2), and finally to output in a linear layer, in which the activation layer uses leaky relu(with negative slope=0.01).

3.2.2 CYUT-2: AWD-LSTM. As a comparison, we used a model architecture similar to CYUT-1 in CYUT-2, with changes in the LSTM layer only. In CYUT-2, we replaced the original BiLSTM with AWD-LSTM to test whether the theoretically more advanced approach is also effective in this task. Other settings such as the number of layers of the model are the same as CYUT-1.

3.2.3 CYUT-3: Data Generation Extensions. The CYUT-3 system is our biggest attempt in this task. In this CYUT-3, we do not use the data resampling method that we describe in the previous paragraph. Instead, we use our another attempt at data augmentation, data expansion. For details of the data generation method, please refer to Section 2.3.2. In the CYUT-3 experiment, a total of 2200 data were generated in Analyst's Report data (Chinese), and the ratio of label 0 to 1 was increased from 1:0.3 to about 1:1. 4000 data were generated in Earnings Conference call data (English), and the ratio of label 0 to 1 was increased from 1:0.14 to about 1:0.7.

3.3 Additional Run

After completing the formal runs submission, in order to complement the attempts that were not completed in time, and to verify the validity of the data augmentation method that we believe it to be. We have done several more additional runs with different orientations based on this method. This section describes our experiments in detail.

3.3.1 Experiment1: More Seeds. First, in the first experiment, we adjusted the text starters that were input to GPT-2 at the beginning. In the original CYUT-3, only one fixed opening was used for both subtasks. In this experiment, we increased the number of text

[1] https://huggingface.co/docs/transformers/main_classes/optimizer_schedules

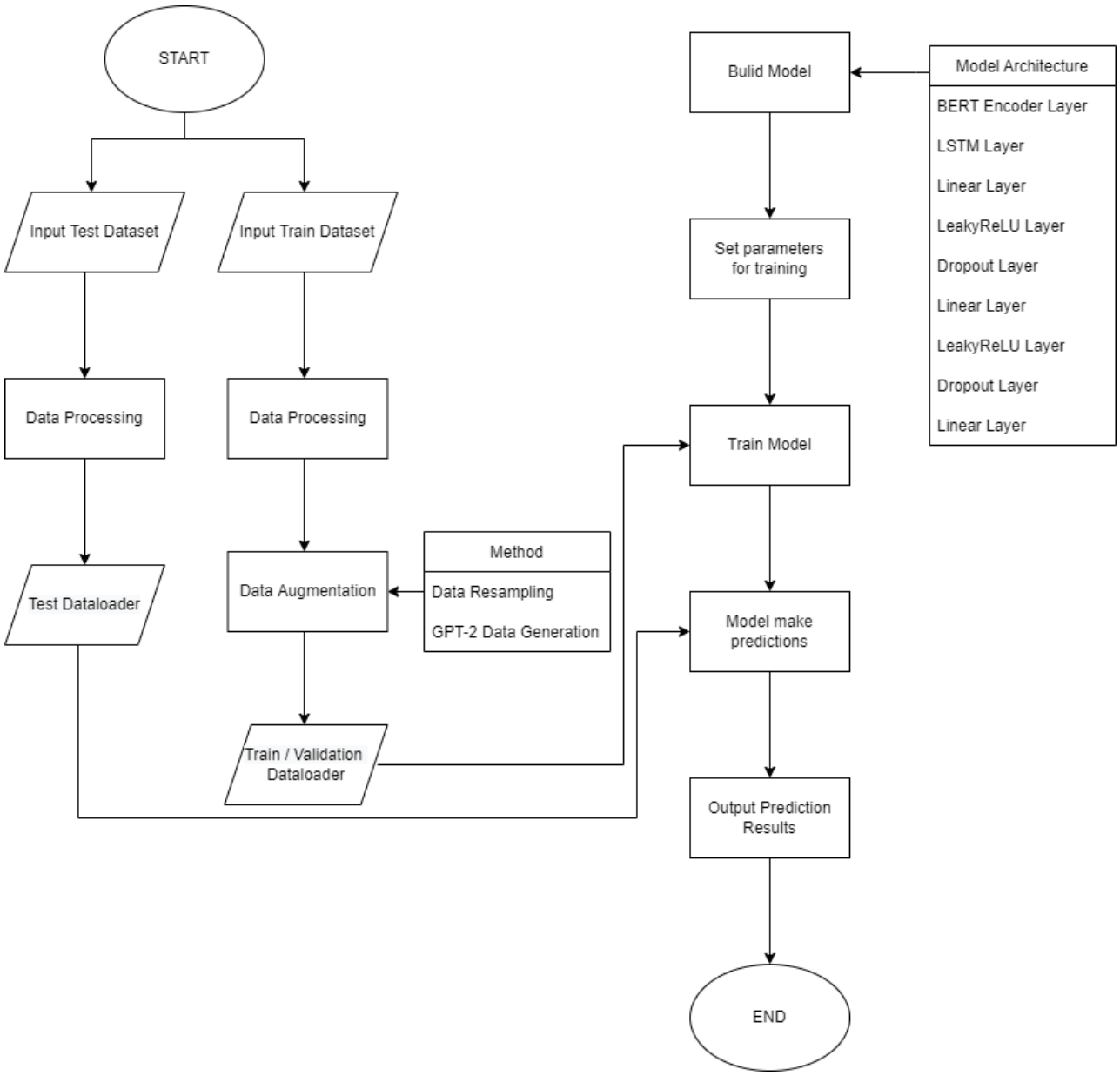

Figure 1: The flow chart of our study

starters to 6 types in Analyst's Report data (Chinese) and 5 types in Earnings Conference call data (English), trying to increase the diversity of text generation. The examples of text starters are shown below.

Analyst's Report (Chinese) :
我們推測會上升 *X (We estimate it will go up X)*
我們估計會下降 *X (We presume it will go down X)*
他懷疑會增加 *X (He suspects it will increase X)*
我猜測會回升 *X (I guess it will go back up X)*
他們預估會下滑 *X (They predict it will go down X)*
專家預測會減少 *X (The experts predict a decrease of X)*

Earnings Conference call (English) :
We anticipate growth of X
They estimate increase of X
He expects a rise of X
She predicts to reduce by X
Expert forecasts a decline of X

3.3.2 Experiment2: More Data. In the second experiment, we wanted to see if the predictive effect of the model could be improved if

more data were used for training. Therefore, this experiment was based on CYUT-3, and the same text start and settings were used to generate additional data. A total of 5,500 Analyst's Report (Chinese) and 13,000 Earnings Conference call (English) data were generated. The ratio of label 0 and 1 was increased from 1:0.3 to about 1:2 and from 1:0.14 to 1:1.9 respectively.

3.3.3 Experiment3: More Data and Seeds. This experiment is a combination of the previous 2 experiments. Using the text start from the Experiment1 and generating more additional data. This ensures the diversity of the data while increasing the amount of data.

3.3.4 Experiment4: 1000 Data. For comparison, in Experiment 4, we only add the first 1000 data generated in CYUT-3 as additional data to the dataset. This is used to test the effect of the amount of additional data on the model training.

3.3.5 Experiment5: No Change. Finally, Experiment 5 is a basic comparison. In Experiment 5, we use only the most basic MacBERT or RoBERTa combined with BiLSTM as our system. Do not use any other data enhancement techniques used in the formal runs and additional runs. The purpose is to test how all our proposed attempts actually have impacts in this task.

4 RESULT AND DISCUSSION

4.1 Result

Table 3 and Table 4 show the performance of our system on the development set in 2 domains respectively. The Analyst's Report is Chinese data and the Earnings Conference Call is English data. It can be seen that CYUT-1, which uses a more traditional approach, performs best on the Analyst's Report data. However, in the Earnings Conference Call data, CYUT-3 with additional data has a certain degree of lead, showing that this method seems to have some effectiveness in the development set.

Table 3: Development Result on Analyst's Report

Run	Macro-F1
CYUT-1	85.23%
CYUT-2	83.49%
CYUT-3	83.00%

Table 4: Development Result on Earnings Conference Call

Run	Macro-F1
CYUT-1	86.59%
CYUT-2	87.05%
CYUT-3	90.76%

Tables 5 and table 6 show the performance of our system in the formal test set in the 2 domains, respectively, along with the teams with the highest correct rates in this task. Similar to the results of the development set, although the overall correctness rate decreased. However, in the Test set, CYUT-1 and CYUT-3 are still the best performers in our system for Analyst's Report and Earnings Conference Call data respectively.

Table 5: Test Result on Analyst's Report

Run	Micro-F1	Macro-F1
CYUT-1	92.11%	88.80%
CYUT-2	91.73%	86.76%
CYUT-3	92.16%	88.20%
IMNTPU-1[1]	95.31%	93.18%

Table 6: Test Result on Earnings Conference Call

Run	Micro-F1	Macro-F1
CYUT-1	94.67%	85.53%
CYUT-2	95.64%	87.49%
CYUT-3	96.43%	87.88%
JRIRD-3[1]	97.27%	91.03%

The table 7 and the table 8 show the performance and ranking of all our additional runs and formal runs on the test set, from top to bottom, the lower the position, the better the performance in Macro-F1. It can be found that although the 2 systems do the similarly classification task, they show completely different experimental results with different target languages.

4.2 Discussion

Looking at Tables 7 and 8, we find that the experimental results seem to be different from what we originally thought.

First of all, in the Analyst's Report data experiment, although the "More Data" system had the highest Macro-F1 as we thought, it should be noted that the system with only 1000 additional data also had nearly 90% results. The Earnings Conference call data section is out of our expection, as all the experiments we did to adjust the data did not perform better than the CYUT-3. This further illustrates that the amount of data does not seem to be the most important in this task.

Moreover, the performance of the experiments with more text starters (More Seeds, More Data and Seeds) was not as good as we expected at the beginning. Although there is an improvement over CYUT-3 for the Analyst's Report data, it is still lower than the experiment with only one text as the beginning (More Data). This phenomenon is even more obvious for the Earnings Conference call data, where both experiments performed lower than the experiment using only one text as the beginning.

Also, it is worth noting that the system without data augmentation (No Change) outperformed the 3 formal runs we proposed for both subtasks. On Earnings Conference call data, the "No Change" system is even the best system in Macro-F1 among all the experiments.

Taking these points together, and considering the way we generate the additional data is straightforward. We conjecture that the amount of additional data is not a decisive factor in this task, but it does improve the training of the model to some extent. However,

Table 7: Additional Result on Analyst's Report

Run	Macro-F1	Micro-F1	Recall
CYUT-2	86.76%	91.73%	90.32%
CYUT-3	88.20%	92.16%	88.76%
CYUT-1	88.80%	92.11%	87.34%
No Change	88.75%	92.52%	89.30%
More Data and Seeds	89.23%	92.86%	89.92%
More Seeds	89.30%	93.14%	91.66%
1000 Data	89.97%	93.16%	89.52%
More Data	90.24%	93.43%	90.31%

Table 8: Additional Result on Earnings Conference Call

Run	Macro-F1	Micro-F1	Recall
CYUT-1	85.53%	94.67%	79.82%
More Seeds	85.93%	95.00%	80.74%
More Data	86.73%	95.93%	84.78%
More Data and Seeds	87.17%	95.76%	83.25%
1000 Data	87.28%	95.76%	83.15%
CYUT-2	87.49%	95.64%	82.39%
CYUT-3	87.88%	96.43%	87.25%
No Change	88.15%	96.22%	85.03%

the quality of the additional data added seems to be more important to improve the accuracy of the system. If the quality of the added data is high enough and closer to the domain of the task, even if only a small amount of additional data is added, the improvement can be achieved to some extent. On the contrary, if the model is trained with poor quality additional data, the prediction performance of the model will be worse than the original one.

Figure 2 - figure 5 shows the error rate of all experiments in each category for official runs and additional runs on this task. The lower the Y value, the lower the prediction error rate of the system in that category.

After dividing the categories of errors, we found that most of the systems in the experiment had particularly good or poor predictions in specific categories. Even the system with the best overall performance in Macro-F1 has this condition. For example, Figure 2 shows that although CYUT-3 does not perform as well as CYUT-1 and CYUT-2 in the Money category, it has a considerable advantage in the Product Number category. In Figure 3, although "More Data and Seeds" does not perform well overall, it has an overwhelming advantage in the Money category.

Similarly, in the Earnings Conference call data section, each system has its own advantages. In Figure 4, we can see that CYUT-3 has a significantly higher error rate in the Product number category, but can predict the Date category with nearly 0% errors. In Figure 5, "More Data", which does not perform well in the Product number category, is able to predict the change category with nearly 0% errors.

In summary, although we do not yet have a system that can predict all categories perfectly. However, we found that there is a significant difference in prediction accuracy between different systems for different categories. Therefore, we believe that if we can select different models to make predictions according to different categories, and build a large system with multiple models to predict data in different categories, then it can effectively improve the overall accuracy. For this task, if the models from " More Data and Seeds", "1000 Data", "CYUT-3" and "More Data" systems can be applied to the Analyst's Report data at the same time. Theoretically the best model can be made in our experiment. It even gets close to 0% error rate in 3 categories. Therefore, we think this will be an idea worth further investigation.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we describe our proposed approach on the FinNum-3 shared task about fine-grained claim detection in NTCIR-16. In this shared task, we submit a total of 3 different approaches. In addition to the classical deep learning methods, we also try to build our system using some newer and theoretically improved techniques. We also tried two data augmentation techniques, data resampling and data expansion, to try to solve the problem of data imbalance when training the model, which we think may be caused by it. The experimental results show that data augmentation has improved the prediction of Analyst's Report (Chinese) to a certain extent, up to 90% of Macro-F1. However, it is important to notice that these methods do not Tlhave much success on Earnings Conference call

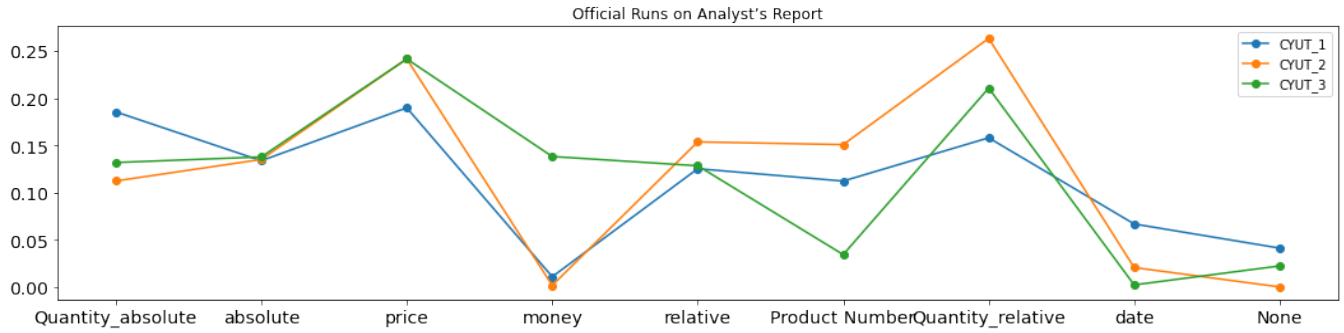

Figure 2: Error Rates for Various Categories on Formal runs on Analyst's Report

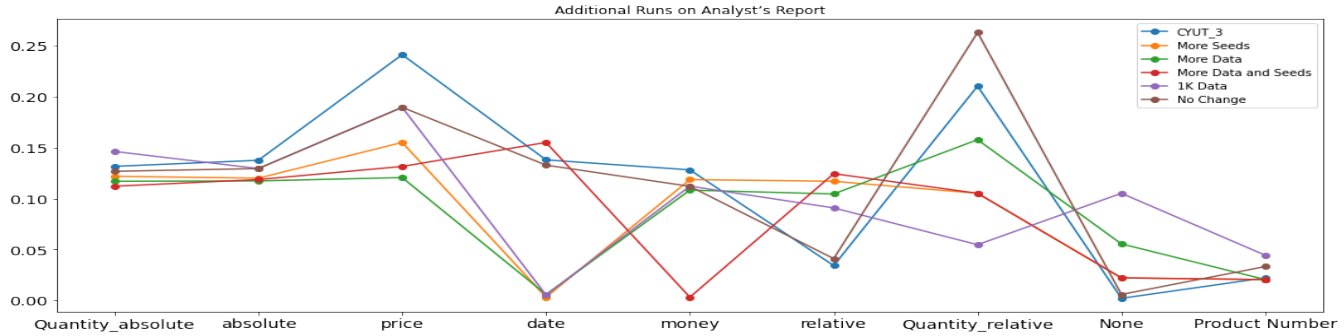

Figure 3: Error Rates for Various Categories on Additional Runs on Analyst's Report

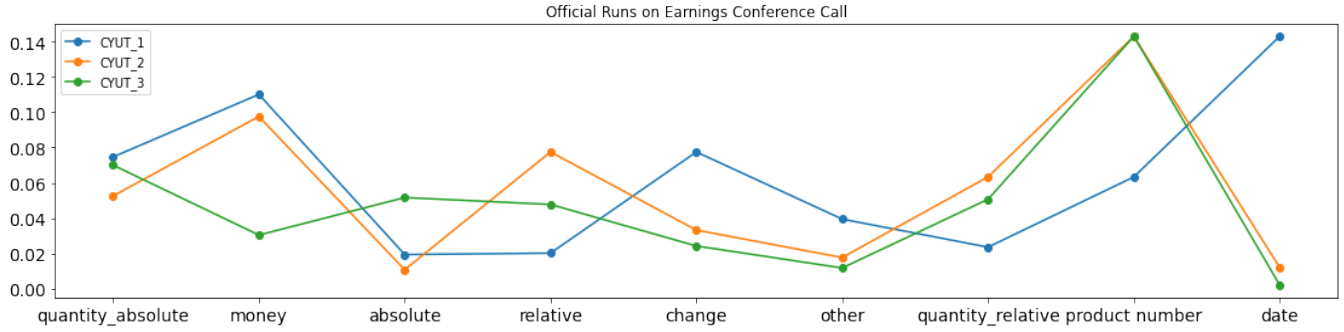

Figure 4: Error Rates for Various Categories on Formal Runs on Earnings Conference Call

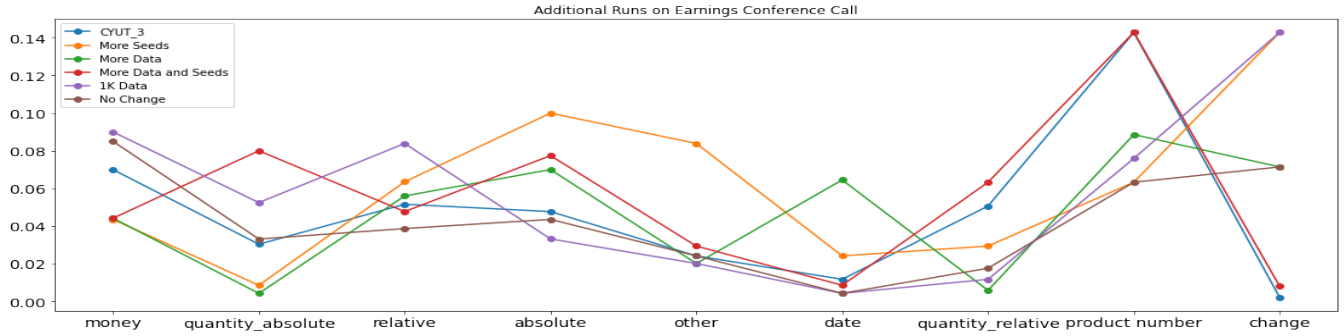

Figure 5: Error Rates for Various Categories on Additional Runs on Earnings Conference Call

(English). It is possible that there is still much room for improvement in the data generation methods. For example, the length of the generated data can be increased, or the GPT-2 model can be pre-trained with data from a similar domain as the task training data before generating additional data, so that GPT-2 can generate more task-appropriate data. These are all possible further attempts.

In addition, since the " No Change" system predicts performance on Earnings Conference call data better than all of our systems in the formal runs, we find that data resampling does not seem to be helpful in this task and that other data augmentation methods would be a better choice. We found that data resampling does not seem to be helpful in this task, and other data enhancement methods would be a better choice. Alternatively is it that we have missed something in the adjustment of data resampling. Or whether excessive resampling has in turn caused overfitting to occur. These are all questions that can be further explored.

Finally, for future work, we can try to implement the large multi-model system mentioned in the previous section. By making full use of the prediction categories that each model is good at, the overall prediction accuracy of the system may be improved.

REFERENCES

[1] Chung-Chi Chen, Hen-Hsen Huang, Yu-Lieh Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2022. Overview of the NTCIR-16 FinNum-3 Task: Investor' s and Manager' s Fine-grained Claim Detection. (2022).

[2] Chung-Chi Chen, Hiroya Takamura, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Overview of the NTCIR-15 FinNum-2 Task: Numeral Attachment in Financial Tweets. (2020).

[3] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting Pre-Trained Models for Chinese Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. Association for Computational Linguistics, Online, 657–668. https://www.aclweb.org/anthology/2020.findings-emnlp.58

[4] Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, and Ziqing Yang. 2021. Pre-Training with Whole Word Masking for Chinese BERT. https://doi.org/10.1109/TASLP.2021.3124365

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

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80. https://doi.org/10.1162/neco.1997.9.8.1735

[7] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1508.01991 [cs.CL]

[8] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

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

[10] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. arXiv:1711.05101 [cs.LG]

[11] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2017. Regularizing and Optimizing LSTM Language Models. arXiv:1708.02182 [cs.CL]

[12] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. An Analysis of Neural Language Modeling at Multiple Scales. *arXiv preprint arXiv:1803.08240* (2018).

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

[14] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG]

[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[16] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[17] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to Fine-Tune BERT for Text Classification? arXiv:1905.05583 [cs.CL]

[18] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of Neural Networks using DropConnect. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 1058–1066. https://proceedings.mlr.press/v28/wan13.html

[19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv:1910.03771 [cs.CL]

[20] Liang Xu, Xuanwei Zhang, and Qianqian Dong. 2020. CLUECorpus2020: A Large-scale Chinese Corpus for Pre-training Language Model. *ArXiv* abs/2003.01355 (2020).

[21] Zhe Zhao, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du. 2019. UER: An Open-Source Toolkit for Pre-training Models. *EMNLP-IJCNLP 2019* (2019), 241.