MIG at the NTCIR-15 FinNum-2 Task: Use the Transfer Learning and Feature Engineering for Numeral Attachment Task

Yu-Yu Chen
National Chengchi University, Taiwan
108753144@g.nccu.edu.tw

Chao-Lin Liu
National Chengchi University, Taiwan
chaolin@g.nccu.edu.tw

ABSTRACT

In the FinNum-2 task, the goal is to judge whether the specified numeral is related to the given stock symbol in a financial tweet. We employ a transfer-learning mechanism and the Google BERT embeddings so that we only need to collect and annotate a small amount of data to train the classifiers for the task. In addition, our classifiers consider some intuitive but useful syntactic features, e.g., the positions of words in the tweets. Experimental results indicate that these new features boost the prediction quality, and we achieved better than 68% in the tests in the formal run.

CCS CONCEPTS

• **Information systems → Information extraction**.

KEYWORDS

Numeral attachment, financial social media, transfer learning, feature engineering

TEAM NAME

MIG

SUBTASKS

FinNum-2: Numeral Attachment in Financial Tweets(English)

1 INTRODUCTION

Numerals usually provide crucial information in documents in many domains, especially in financial reports. It is important to clarify the relationships between words and numbers to retrieve useful information. Some researchers focus on mining the numeral information in the textual data in finance. [2] For example, one may judge that "30" in "the stock price increases 30 % in a few days" is related to "the stock price" rather than "a few days". If we can extract informative elements from the texts, e.g., related entities and numbers, we could take advantage of the extracted information for making influential financial decisions.

Chen et al. [1] use the capsule-based model and the auxiliary task to improve the performance. They use two auxiliary tasks to improve the performance of the main task: reason detection (reason-binary) and fine-grained reason type classification (reason-type). [3] At that time the dataset contains 7,984 instances. They use 20 % as test set and 80 % as train set. The macro-F1 score of test data is 73.46. In our research, we use the transfer learning and add syntactic features to enhance our performance. We fine-tune the transfer learning model BERT (Bidirectional Encoder Representations from Transformers) [5] on our relatively smaller dataset. In the FinNum-2 task [4], there are total 10,340 instances in the NumAttach dataset, which is 23 % more than the previous dataset. The new dataset doesn't have the information on the reason detection and fine-grained reason type. We use 20 % as test set, 10 % as development set, and 70 % as train set. We further take apart the train set to the validation set and the train set. Our macro-F1 score of development data is 85.77 and the score of test data is 68.72.

The method we use is different from the original one. First of all, they use the capsule network architecture [7]. It's important to have a lot of data to train the entire network to obtain the embedding of text. They collect more than two million financial tweets from Twitter and learn the document-oriented token embeddings with the skip-gram model. The embedding is the concatenation of token, character, position and magnitude embeddings and it has additional features like reason detection and fine-grained reason type classification. If they only use the capsule network without the two auxiliary tasks, their macro-F1 score on the test data is 67.14.

While we use the pre-trained model and the pre-trained embedding of text. BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia (2,500 million words) and Book Corpus (800 million words). It's a character-based model, so we have a function that is similar to the character embedding. Also, we add the syntactic features like the position of cashtag, the position of number, and the distance between cashtag and number and the average of them. By using transfer learning, we don't need to collect millions of data first. We only need to collect and annotate a small amount of data. At the same time, we can have a similar performance compared to collect many data. So that we can save the collection and the training time. Also, we propose some intuitive but useful syntactic features. The results show that the features further increase the performance.

In Section 2, we will describe our method in detail. It includes how we used the model and how we generate the features. Besides, we mentioned other experiments we made briefly. In Section 3, we present the evaluation results. In Section 4 we discuss all the experiments and make some comparisons. We make a summary in Section 5.

2 METHOD

We fine-tuned the BERT and applied linguistic domain knowledge into new feature. BERT is trained based on a character level but we think the word level shouldn't be neglected. Therefore we need some word-level information and our feature is generated on a word level.

2.1 Fine-tune BERT

We use BERT-large as our pre-trained model because the BERT-large performs better than BERT-base model in our task. When we use the BERT tokenizer, it will mask some character in words. For example, there is a tweet content as following:

"@ElephantLover @osirustwits because he makes me f@cking money are you up $67,673 locked in and another $26,000+ still in play on $ONCS"

The BERT tokenizer will separate this sentence into:

['[CLS] ', '@', 'Elephant', '##L', '##over', '@', 'o', '##sir', '##ust', '##wi', '##ts', 'because', 'he', 'makes', 'me', 'f', '@', 'c', '##king', 'money', 'are', 'you', 'up', '$', '67', ',', '67', '##3', 'locked', 'in', 'and', 'another', '$', '26', ',', '000', '+', 'still', 'in', 'play', 'on', '$', 'ON', '##CS', '[SEP] ']

It doesn't encode the sentence into the complete words. After tokenize the sentence, we treat the output as the input of the pre-trained model. The model will fine-tune the text embeddings of the sentence based on the task. BERT model can fit into several downstream tasks such as sentence pairing. Here we treat our task as sentence pairing and we use features as the second sentence.

We made a hypothesis: The text embeddings we learned can capture the main idea of a tweet. And if the target cashtag and target number are correlated, they will also relates to the meaning of tweet. If the target cashtag and target number are not related, they are not relate to the meaning of tweet either. Hence the features we use is all related to the cashtag and number.

Since there is data unbalance in our data, we set the weight of loss function to response to this situation. In all tweets, about 82% the cashtags and numbers of tweets has a relation. Therefore we try some different weights to have better results. We also try to add a Bi-directional Long Short-Term Memory(BiLSTM) [6] after the layer of BERT. And we tuned the number of epoch to reach a better performance.

2.2 Feature Engineering

The features we use is all related to the cashtag and number. We make an assumption that if a word has an important meaning in a sentence, generally in human eyes we can tell the important word immediately. Because in English, the main idea are usually in the first sentence, which is called topic sentence.

We try to make machine imitate human behavior and human read the sentence on a word level. At first, we did the word segmentation and numbered the word in the order of the position of a tweet. And the numbers we generated are the position features. For example, we encode this sentence: "$NE, last time oil was over $65 you were close to $8." from 0 to 11. The cashtag is $NE so we generate a feature in the number 0. The number is $65 so we generate a feature in the number 6.

In the view of linguistics, if the distance between words are close, it might has higher probability that the words are correlated. There are many certain patterns in linguistic structure. Usually a noun comes with an adjective or a verb. In our case, the cashtag and the number are both noun so the patterns might be n.+v.+n. or n.+adj.+n. or other patterns that the distance isn't too far from each other. As the result, we treat the distance between the cashtag and the number as our feature.

Some of tweets have several sentences. We calculate the length of tweets on a word level. If the length of a tweet is long, it means that the author might used many words to modify his/her main idea. So if the cashtag and number are relate to the meaning of main idea, there might be a lot of adjective words between them. In this case, the distance might be high and the effect will be diluted. Therefore, we need to average them with the length of entire tweet. We create three more features which include average position of cashtag, average position of number and average distance between them. Totally we create six new features.

We also did some experiment as following but it didn't have great improvement. First of all, we use the tweet as the input of BERT model and we extract text embeddings. Then we train a feed-forward neural network with the features we created and get the hidden layer embedding of the architecture, which is the feature embedding. We concatenate the text embedding(768-dimension) and the feature embedding(768-dimension). And we use the new embedding to train another feed-forward network.

Secondly, we try to make features in the form of word. We use the word of cashtag and number themselves as input. And we treat these two words as the sentence two for the sentence pair task in BERT.

Another experiment we made is try to find some financial keyword in tweets. We extracted the data that the label equals to 1. Then we find the top-n terms by calculating term frequency–inverse document frequency(TF-IDF). We expect to find some keywords that can represent the relationship. We try to either use the whole terms or some keywords we picked manually.

Table 1: Formal run results

	Val set	Dev set	Test set
Run1	0.8021	0.8446	0.6827
Run2	0.8473	0.8577	0.6872
Run3	0.7252	0.9069	0.6837

3 RESULTS

In this section, we describe the settings of our systems for formal runs for the FinNum-2 task. We submitted three runs for the test set. We set up three different settings for this purpose. These three runs have better performance for the development set in all the experiments we made. We list the performance of each runs in Table 1. We use the Macro-F1 score to evaluate the performance of the validation set, development set, and test set.

In run 1, we set the weight of loss function to 0.99 and 0.01 corresponds to the data imbalance. It focuses on unrelated data more than related data. And there is a BiLSTM layer after the pre-trained model. We try different epochs in the experiment and the outperformed one is 5 epoch.

In run 2, we set the weight of loss function to 0.8 and 0.2. This ratio is similar to the data distribution. And there is also a BiLSTM layer after the pre-trained model. The number of the epoch is 8.

In run 3, we set the weight of loss function to 0.9 and 0.1. This time we didn't add a BiLSTM layer after the pre-trained model. And the epoch 8 also has a better result.

The result appears that Run 2 has the best performance. The reason might be the weight of the loss function is the most similar to the data distribution. We also list the different epochs in each run in Table 2 to Table 4.

Table 2: Results of Run 1

# of epochs	Val set	Dev set
5	0.8021	0.8446
8	0.8251	0.8354
12	0.8531	0.8178
15	0.8585	0.7786

Table 3: Results of Run 2

# of epochs	Val set	Dev set
5	0.8542	0.7845
8	0.8473	0.8577
12	0.8573	0.8044

Table 4: Results of Run 3

# of epochs	Val set	Dev set
5	0.8573	0.7851
8	0.7252	0.9069

4 DISCUSSION/COMPARISON

In this section, we describe the details of our all experiments. We discuss them and make some comparisons. Here we use the Macro-F1 score of the validation set as our evaluation method.

First of all, because most of our data are labeled as 1, we guess all the data 1 and treat it as the default model. Then we try to use the features that are in the original form of data, which includes the offset of the target number and the number itself. Usually, the important information would be mentioned in the beginning. So if the number means something crucial in that tweet, the position of it might be small. We use these two features to train a multiple layer perceptron(MLP) simple architecture. We set the hidden layer size to 128 dimensions. We observe that the features aren't helpful to recognize the relationship between the number and cashtag. It's even worse than the null model. We think the possible reason for becoming worse might be the characteristic of tweets. Because the content of tweets are often constructed in one or two sentences, it doesn't suitable for the hypothesis that the crucial information would be mentioned at first. In a short text, most of words represent meaningful thought.

As we mentioned previously, we add the position of cashtag, the position of number and the distance features and we calculated these features on a word level. We try to only use the features that we created to train a MLP in order to confirm these features are useful. The results in Table 5 prove our concept.

We use the BERT-base-cased model and it's a great improvement over the null model or others. Next, we hope to mix the effect of the BERT model and the new features we made. As the following, we list several ways to add the new features . One of them is concatenating the embeddings of BERT and the features. During the experiment, we think the new features are important to the task. We think that

Table 5: Null model and numeral features in MLP

	Macro-F1
Null model	0.4872
Original features	0.4811
New features	0.7600

Table 6: BERT and merged embeddings

	Macro-F1
Null model	0.4872
BERT-base-cased	0.8488
New features	0.7600
Merged embeddings	0.6537

Table 7: Evaluation of the experiments

	Macro-F1
Single sentence	0.8488
Words features	0.8457
3 new features	0.8571
6 new features	0.8636
7 new features	0.8618

the information about our three features is as influential as the text meaning. So we amplify the dimension of the features from 3 to 768, which is the same size as the text embedding. The method we generate the 768-dimensions features is by extracting the hidden layer of a MLP network that fitted in our data. Then we concatenate the 768-dimensions of features embedding and the 768-dimensions of text embedding. Finally we get a 1,536-dimensions mixed embedding and use it to train another MLP network. Nonetheless the performance becomes much worse than only use the pre-trained model or only use the numeral features. The performance is listed in Table 6.

We also try to use the word of cashtag and number themselves as input. We treat the two words as sentence two for the sentence pair task in BERT. The performance seems not sensitive to the two words. It's just slightly worse than a single sentence. We refer that the meaning of the words itself isn't related to the relationship between them. Then we add our new features to sentence two and it's better than a single sentence. We further add the average of three features. Therefore totally we have 6 features and the performance increase again. The results are shown in Table 7. The single sentence means it doesn't treat the experiment as sentence pair task. It only uses single sentence (text) to generate the output.

Another feature that we create is called if the keyword exists. We extract the related data and find some keywords by TF-IDF analysis. We remove some meaningless terms manually like numbers and stop words and create a dictionary:

['will', 'up', 'amp', 'shares', 'more', 'today', 'now', 'over', 'down', 'buy', 'next', 'stock', 'out', 'day', 'back', 'just', 'target', 'short', 'bought',

Table 8: BERT-base vs. BERT-large

	Macro-F1
BERT-base	0.8488
BERT-large	0.8823

'price', 'get', 'year', 'last', 'week', 'SPY', 'market', 'million', 'BTC', 'close', 'calls', 'trading', 'AMD', 'sell', 'bullish', 'big']

If the tweets contain any keyword in the dictionary we made, then we annotate this tweet as 1. Otherwise, we annotate as 0. And we save the number as the new feature. In the same way, we treat this feature as sentence two for the sentence pair task in BERT. We add the feature to our best model so far but the performance isn't good as well as only use 6 features. Although the keywords seem to have some meaning in our thought, it doesn't provide a great help. We think the keyword might be only related to the cashtag or the number, not the relationship between them. The result is showed in Table 7.

We also compare the BERT-base-cased and BERT-large-cased model in Table 8. BERT-base has 12 layers (transformer blocks), 12 attention heads, and 110 million parameters. BERT-Large has 24 layers, 16 attention heads and 340 million parameters. Finally, we use BERT-large because it performs better in this task although it cost more time to train.

In Table 1 formal run results, it's interesting that in Run1 and Run2 both have a BiLSTM layer and they perform better than Run3 in the validation set. While in the development set Run3 has better performance than Run1 and Run2. The result shows that in the test set all three runs have a similar Macro-F1 score. So we infer that adding a BiLSTM layer doesn't have a significant effect in this task.

5 CONCLUSIONS

In the research, we find that linguistic concepts can help computers to recognize the relationship between entities. Nowadays we have high-dimensional pre-trained embeddings, so we don't have to spend a lot of time to train the text embeddings whereas we can achieve the basic meaning in a text. We add the distance feature and the feature further increases the performance. It's better than only use the context of the tweet. The result represents that the knowledge of linguistics is important to the relation recognition task. In the future, we can focus more on the relationship of the domain keywords, the cashtag and the number. Because in the field of finance, the financial keywords usually have an important meaning in the text. Also, we can study more on unrelated data and extract some features that can represent them.

ACKNOWLEDGMENTS

This research was support in part by the contract MOST-107-2200-E-004-009-MY3 of the Ministry of Science and Technology of Taiwan. This shared task was partially supported by Ministry of Science and Technology, Taiwan, under grants MOST 109-2218-E-009-014, MOST 109-2634-F-002-040, and MOST 109-2634-F-002-034, and by Academia Sinica, Taiwan, under grant AS-TP-107-M05.

REFERENCES

[1] Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. 2019. Numeral Attachment with Auxiliary Tasks. In *SIGIR '19: The 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[2] Chung-Chi Chen, Hen-Hsen Huang, Yow-Ting Shiue, and Hsin-Hsi Chen. 2018. Numeral Understanding in Financial Tweets for Fine-grained Crowd-based Forecasting. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI).*

[3] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2019. Overview of the NTCIR-14 FinNum Task: Fine-Grained Numeral Understanding in Financial Social Media Data. In *Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies. 19–27.*

[4] Chung-Chi Chen, Hen-Hsen Huang, Hiroya Takamura, and Hsin-Hsi Chen. 2020. Overview of the NTCIR-15 FinNum-2 Task: Numeral Attachment in Financial Tweets. In *Proceedings of the 15th NII Testbeds and Community for Information Access Research (NTCIR-15) Conference.*

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).*

[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. In *Neural computation.*

[7] Min Yang, Wei Zhao, Jianbo Ye, Zeyang Lei, Zhou Zhao, and Soufei Zhang. 2018. Investigating Capsule Networks with Dynamic Routing for Text Classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.*