Chinese Financial News Sentiment Analysis

WANG Yicheng
The University of Hong Kong
u3008632@connect.hku.hk

Abstract

Natural Language Processing (NLP) lets machines understand and interact with humans. In the past research, machine learning techniques such as deep neural networks outperformed traditional models in some NLP tasks. We wonder how useful it is to apply it to a sentiment analysis task in the financial market. We apply from rule-based approach to BERT to predict negative financial news. Unlike most practical works done in English, we particularly focus on Chinese natural language texts. We show that deep learning methods can do similarly well but no better on Chinese sentiment analysis, but it is still worth trying in the industry.

1 Introduction

NLP is an inseparable part of artificial intelligence that machines can interpret human languages. It went through several eras: Symbolic NLP from the 1950s to 1990s, Statistical NLP from 1990s to 2010s, and Neural NLP since then. In the symbolic NLP stage, machines were asked to perform tasks by applying the given collections of rules. In the statistical NLP stage, machine learning (ML) techniques were used in NLP. Neural network (NN) algorithms can be dated from the 80s but are applied in NLP starting in a recent decade. People use NN to replace feature engineering and try to feed everything into the network. With the increasing computational power and available data resources, more advanced deep learning pre-trained models are developed, such as BERT. These advanced methods have shown state-of-the-art results in many NLP tasks, e.g., text classification and language modeling.

NLP and ML have been widely used in quantitative finance, my field of study. P-quant is a group of quantitative finance people who extract information from past data to predict future market movements. They often use machine learning methods on some unique datasets to gain such exclusive information. One known strategy is to look for buy/sell signals from positive/negative financial news. Many have done on English news, but a few worked on Chinese news. The reason behind this is that the publicly available news datasets from Chinese news sources are often limited and Chinese languages are harder to process. Therefore, in this project, we apply several ML methods to do Chinese financial news sentiment analysis. We find NN is useful in Chinese NLP tasks and we can use the output as trading signals on Chinese public companies.

The project will proceed as follows: first, it will preprocess data, which are some Chinese financial news texts; later, several methods will be applied to predict the sentiments of these texts, including a rule-based approach using Lexicon, traditional ML approach using Lexicon or Tfidf, deep learning approach using Recurrent Neural Network (RNN) or LSTM architecture, and more advanced NLP pre-trained models BERT; lastly, the performances of these methods will be compared and discussed.

2 Data

Our data consists of 10k Chinese financial news related to lending services from a data contest [1]. It comes with an excel file with information already parsed from the news sources. Each data sample includes the title, content, the entities and the negative entities. 5k of them are labeled, which indicates whether they are negative news (1 means negative, 0 means positive). We are only going to use these labeled data, of which 80% for training and 20% for testing. We find out the labels are

[1] Source: https://www.datafountain.cn/competitions/353/datasets

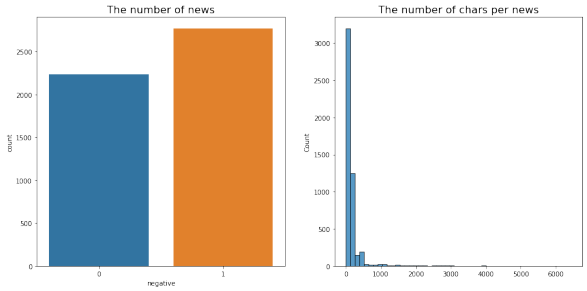


Figure 1: Exploratory data analysis.

Figure 2: Wordcloud.

Field Info	Type	Description
id	String	Data ID
title	String	Text title
text	String	Text content
entity	String	Given entities
negative	Boolean	Does the text contain negative information
key_entity	String	Key entity that is negative in the news

Table 1: Data column descriptions.

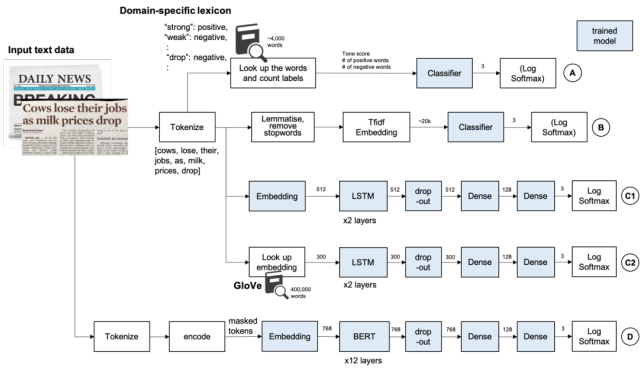


Figure 3: Model framework.

balanced as shown in Figure 1. And the length of each news varies. Data descriptions are shown in Table 1.

2.1 Preprocessing

Since there are two types of characters in the text: English and Chinese, we encode characters by 'utf-8'. We will ignore punctuation, unmeaningful English phrases, Chinese stopwords [2]. Compared with English, there are no spaces between Chinese words and there are no lowercase and tenses of Chinese characters. Hence, there is no need to normalize and stem Chinese characters, and we will use a specialized tokenizer for Chinese.

"Jieba" [3] is a Python Chinese text segmentation tool used to cut Chinese sentences into words. We use it to tokenize each sample text and end up with 27950 tokens in total. In this process, we also passed in a list of entity words from our data to the dictionary of the tokenizer, since these words are specific to our task and need to be identifiable. We display the occurrences of all unique Chinese words in a Wordcloud in Figure 2.

[2]Source: https://github.com/stopwords-iso/stopwords-zh

[3]Python Library: https://github.com/fxsjy/jieba

3 Modeling

After preprocessing the data, we start modeling. In this section, we present various models to predict Chinese news sentiments.

3.1 Methods

Different approaches will be required for different tasks and in most cases are the combination of multiple NLP techniques (Takahashi, 2020). In the symbolic NLP era, a rule-based approach can handle the grammar in language, while ML approach in the statistical NLP era could understand semantics better. In the Neural NLP era, some neural networks are shown to deal with contexts better. In 2018-2019, NLP has entered the era of pretrained models.

The task for this project is to predict whether the news contains negative information. We will borrow from a previous framework done for English texts [4] to our task in Chinese, shown in Figure 3. Many models from the simplest rule-based to the more complicated BERT are implemented and compared according their complexities and performances.

[4]Source: https://github.com/yuki678/financial-phrase-bert

Features	Description
x_1	the number of entities
x_2	the number of negative entities
x_3	total number of words in a text
x_4	the number of positive words according to LM dictionary
x_5	the number of negative words according to LM dictionary
x_6	the number of positive words according to NTUSD dictionary
x_7	the number of negative words according to NTUSD dictionary
x_8	$LMTone1 = (x_4 - x_5)/x_3$
x_9	$LMTone2 = (x_4 - x_5)/(x_4 + x_5)$
x_{10}	$NTUSDTone = (x_6 - x_7)/(x_6 + x_7)$

Table 2: Feature descriptions.

3.2 Rule-based approach

In rule-based classification, we count the number of negative words versus the number of positive words in a piece of news and if we have a greater proportion of negative words, we output its sentiment negative.

Based on the research of 曾庆生;周波;张程;陈信元; (2018), the list of financial emotional English words provided by LOUGHRAN and MCDONALD (2011) (LM) were translated to Chinese according to Youdao Dictionary and Kingsoft Iciba. The final word list includes 2080 negative words and 1076 positive words. As a result, we calculate two-tone scores: "LMTone1" and "LMTone2". Referring to the research of 王华杰;王克敏; (2018), using the National Taiwan University Semantic Dictionary (NTUSD) produced by Taiwan University (ku2), we calculate "NTUSDTone". Definitions of these tone scores are given in Table 2. The larger the value is, the more positive the tone of the text message is. To predict by tone scores, we take the average of these three scores x_8, x_9, x_{10} and predict the tone to be negative if the average score is smaller than or equal to 0. We can see using this simple counting rule, the accuracy can achieve about 74%.

3.3 ML approach

The rule-based method can be very accurate, but rules remain hard to define. We can also try some statistical methods to do predictions. In the training data, we are given texts and its respective labels. So we are able to use supervised learning to train on these data.

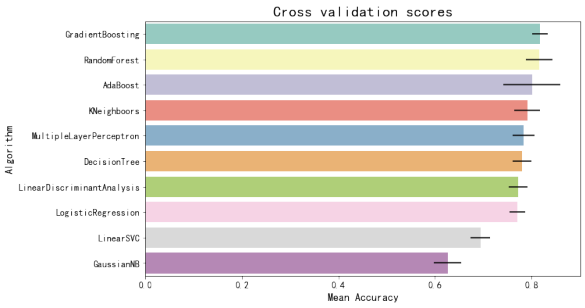


Figure 4: Machine Learning models performance.

3.3.1 ML using Lexicon-based features

We can design features to feed into classifiers for prediction. Our features are based on word counts described in Table 2.

There are many types of classifiers. The simplest model is Naive Bayes, which runs fast, requires low storage, robust to irrelevant features, but it works less well as it requires independence between features, which is not the case in our designed features. More advanced classifiers perform better with our sample size and designed features. Logistic regression is a useful baseline model, as its output is naturally 0 or 1 for binary classification that fits our labeled data. And this method enables more freedom in designing features and is more robust to correlated features. For example, x_8 is actually a function of x_3, x_4, x_5.

We also work on a number of other classifiers, including Support Vector Classifier, LDA, Decision Tree, Ada Boost, Random Forest, Gradient Boosting, Multiple Layer Perceptron, and K-nearest Neighboors. We use 10-fold cross-validations for training and validation to reduce bias. The hyperparameters are by default using the sklearn package. The relative performance of different ML classifiers is shown in Figure 4. We see Gradient Boosting gives the best performance with an accuracy score of 0.8247 and an f1 score of 0.8232, while Gaussian Naive Bayes is the worst due to correlated features. Figure 5 shows the most important feature in our classifier is variable x_8 "LMTone1".

3.3.2 ML using Tfidf

There is a large degree of freedom in designing features. We can also retrieve information directly from tokens in texts.

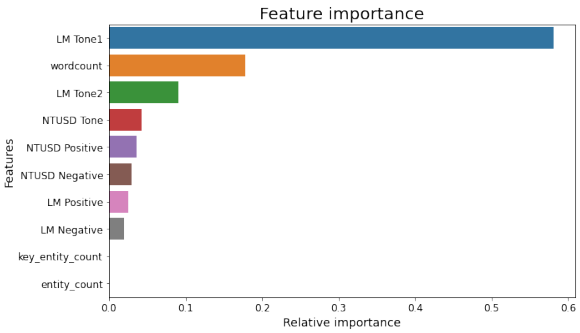


Figure 5: Features ranking.

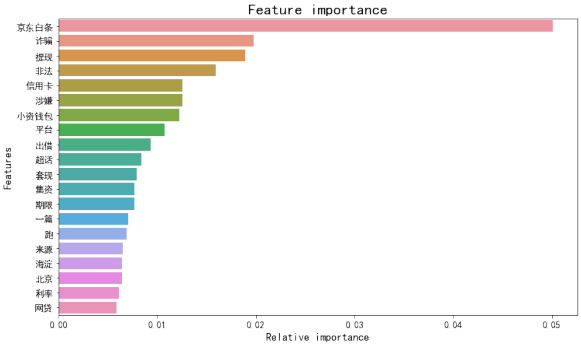


Figure 6: Words Ranking.

We see from Figure 2 ”公司” is a common word in all news, however, this word represents an entity that does not give any emotional tendency, making it less useful in our prediction. As a result, we want to assign a higher score to terms that appear more frequently in a text and less frequently across all texts, so we will use tf-idf weights to obtain such a score. It is calculated as $w_{d,t} = (1 + log(tf_{d,t})) * log(N/df_t)$. Since we only build the weight matrix on the train set, each w_d represents a score vector with a dimension equals to the number of words in the train set. Any word that appears in a valid set but not a train set is ignored. But this matrix is sparse as the dimension is still large.

We can use this sparse matrix as features in some ML models, such as logistic regression and trees. We achieve a high model accuracy of about 95% using either of these two ML models. Figure 6 shows the most important tokens/features are ”京东白条”, ”诈骗”, etc. For example, ”诈骗” meaning ”fraud” explicitly implies the news is negative.

3.4 RNN/LSTM

Compared to traditional ML methods, Recurrent Neural Network (RNN) architecture, which uses the previous information from the input sequence and handles time-series data, performed better in capturing and remembering the context (Takahashi, 2020). Rather than Feedforward NN, RNN can handle inputs of variable length like texts. One of the typical architectures Long Short-Term Memory (LSTM) overcomes the vanishing gradient problem of RNN. They are proven to be highly effective approaches to the text classification task.

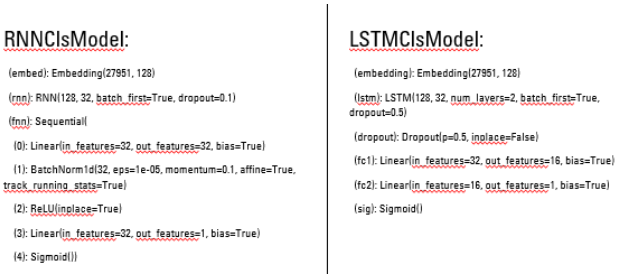


Figure 7: RNN/LSTM model architecture.

We need to do an additional preprocess of the data before we feed them into RNN. We first create a Vocab to Int mapping dictionary. From Figure 1, we see texts come from different lengths. In order to use batching, we would like to make the input length the same, where we pad sentences of shorter length with zero and truncate longer length ones. Each text length is 1000, where the ”pad” token fills the empty spaces and the ”unk” token replaces tokens that are not in the Vocab dictionary. We then encode the tokens. That is, the index 1-27950 are mapped 27950 tokens respectively, and the zero index represents ”pad” token and the last index 27951 represents ”unk” token. After encoding, each text becomes an Int sequence of the same length, which is then passed into RNN.

Two designed NN architectures are shown in Figure 7. We initialize the feature embeddings and learn them from the data progressively. In RNN, one RNN layer is followed by two linear layers. We also normalize them between two linear layers. ReLU is used as an activation function between linear layers because it trains quickly and performs well due to good gradient backflow. We add dropout to each RNN and LSTM layer to prevent overfitting. In LSTM, we use two LSTM layers, which is usually better than one layer. We train data in batches. Adam optimizer is a good one to start with, giving differential per-parameter learning rates. The initial learning rate is set to 0.001. From Figure 8, RNN is able to achieve 89% accuracy, and LSTM can get higher 93%. They even perform worse than ML using Tfidf.

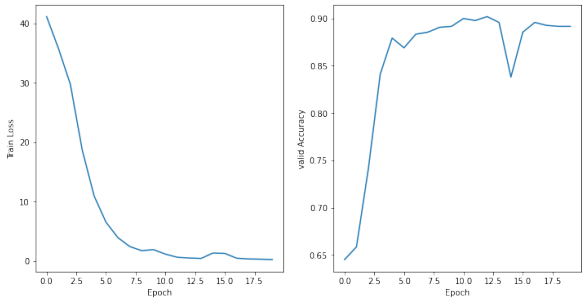


Figure 8: RNN/LSTM model performance.

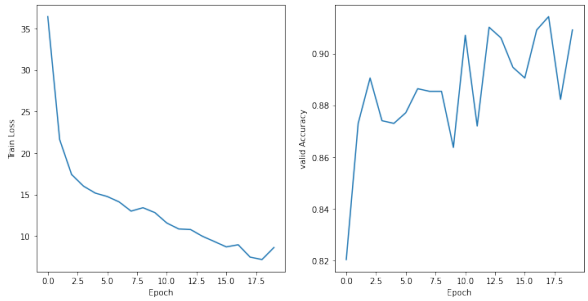


Figure 9: RNN/LSTM with pretrained embedding performance.

3.4.1 Pretrained Chinese Word Embeddings

Embeddings in the previous subsection 3.4 are subject to change, so it might be better to use fixed embeddings for particular words. And Tfidf matrix in subsection 3.3.2 is sparse and does not contain context information. People often want to use more condensed vectors to represent words under the distributional hypothesis. Popular methods are word2vec (Mikolov et al., 2013), Glove (Pennington et al., 2014), FastText (Bojanowski et al., 2017), etc. However, many pre-trained embeddings using these methods are for English. We find Tencent AI Lab embedding corpus for Chinese words and phrases by Song et al. (2018). More suitable for the Chinese language, the method they use is directional Skip-Gram, based on word co-occurrence and the directions of word pairs. We use the embedding corpus of vocab size 2 million and dimension 100. Vectors initially learned with on large corpus of Chinese texts can be used in a neural network for sentiment classification. We do extrinsic evaluation by plugging these word embeddings into our NLP system and seeing whether this improves performance.

In our procedure, we first convert tokens in texts into word embeddings. Regular Chinese tokens are found in the Chinese embedding corpus. If a token is not found, we use randomly generated embeddings with a mean of zero and a standard deviation of 0.6 for this token. We use zero embedding for "pad" token and the average of all word embeddings for "unk" token. A weight matrix consists of all word embeddings is loaded in NN. Then we train our RNN/LSTM as before. From Figure 9, RNN with pre-trained embeddings improves 1% accuracy, but LSTM does not seem to improve.

3.5 BERT

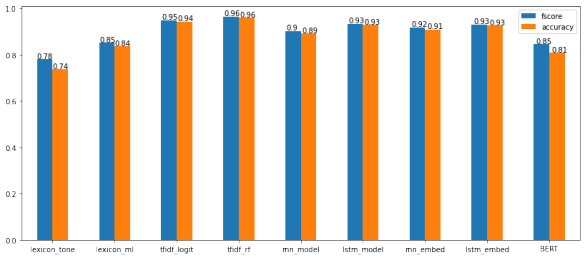
To overcome the information bottleneck of RNN, Vaswani et al. (2017) proposes a transformer architecture, which is an encoder-decoder stack based on the attention mechanism. Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) is a masked state-of-art language model with multiple encoder stacks by Google in 2008. It pre-trains deep bidirectional representations from unlabeled texts by jointly conditioning on both left and right contexts in all layers. As a result, it can be fine-tuned with just one additional output layer for various applications, without substantial task-specific architecture modifications.

Huggingface provides open-source NLP implementation and pre-trained models. For our task, we use Chinese RoBERTa-Base Models for Sequence Classification. RoBERTa (Liu et al., 2019) is a BERT model pre-trained on a large corpus of data in a self-supervised fashion. It was pre-trained on the raw texts only with no human labelling. In order to directly predict sentiments of Chinese texts, we use a task-specific finetuned RoBERTa "liam168/c2-roberta-base-finetuned-dianping-chinese" [5]. Inputs are raw texts, and outputs are predicted labels. We see an accuracy score of 81%, even lower than the best ML model using Lexicon-based features.

4 Discussion

We want to correctly predict negative news given a collection of news. We calculate precision, recall, f1, and accuracy score for all methods used and compare. The performance summary is shown in Figure 10. Although BERT achieved the highest precision 97%, it has a particularly low recall. It means that using this finetuned model, almost all negative predictions are actually negative while a large proportion of negative news is not identified.

[5] Finetuned model: https://huggingface.co/liam168/c2-roberta-base-finetuned-dianping-chinese



	lexicon_tone	lexicon_ml	tfidf_logit	tfidf_rf	rnn_model	lstm_model	rnn_embed	lstm_embed	BERT
precision	0.843038	0.873840	0.964945	0.968635	0.918095	0.939048	0.942857	0.920000	0.969474
recall	0.726621	0.830688	0.932264	0.958029	0.886029	0.930189	0.895118	0.941520	0.748955
fscore	0.780512	0.851718	0.948323	0.963303	0.901777	0.934597	0.918367	0.930636	0.845066
support	3208.000000	567.000000	561.000000	548.000000	544.000000	530.000000	553.000000	513.000000	3350.000000
accuracy	0.737748	0.836000	0.942714	0.959799	0.891641	0.928793	0.909185	0.925697	0.808053

Figure 10: Performance summary.

The reason might be that BERT model finetuned for reviews may not be suitable for financial news, a different type of corpus. The best-performed model of all metrics is ML using Tfidf. It seems that simply extracting information from individual tokens without context information can achieve good performance in our task. But there is a lot of room for us to tune the hyperparameters of these models to get better performances.

5 Conclusion

This project has used various methods from a rule-based approach to machine learning techniques for Chinese financial news sentiment analysis. State-of-art deep learning method such as BERT does not seem to outperform traditional ML models in this task, because we have not finetuned the model using our own data and it is hard to find finetuned BERT for the same task. The result is not deterministic based on a rather simple manual hyperparameter tuning in our proposed methods. BERT might also be better suited for a more refined prediction. We can measure sentiment as a degree of how good or how bad, rather than a binary "good" or "bad", . This framework is applicable to other tasks related to sentiment analysis in Chinese, and the results could be different. For example, in quantitative trading, people also often look at financial reports and their sentiments as trading signals. We could finetune RoBERTa model introduced previously by financial reports data. There are many ways we can research and develop machine learning models on NLP for production use in trading systems.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

TIM LOUGHRAN and BILL MCDONALD. 2011. When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

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)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Yan Song, Shuming Shi, Jing Li, and Haisong Zhang. 2018. Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 175–180, New Orleans, Louisiana. Association for Computational Linguistics.

Yuki Takahashi. 2020. Nlp in the financial market — sentiment analysis. [Accessed: 05/12/2022].

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

曾庆生;周波;张程;陈信元;. 2018. 年报语调与内部人交易. 管理世界, 34(09):143–160.

王华杰;王克敏;. 2018. 应计操纵与年报文本信息语气操纵研究. 会计研究, (04):45–51.

A Supplemental Material

Python code please refer to: https://github.com/wangy8989/Chinese-Financial-News-Sentiment-Analysis