Risk Analysis and Default Prediction for Taiwan Companies

Milestone II Report

Team Member: Yuan-Hsi Lai, Pei-Ling Tsai

**Current Status**

In milestone I (The green bubbles in ***Figure1.***), we finished retrieving data from news, Twitter and financial statement (Please refer to Appendix I for screenshots of source codes).

A close up of a logo

Description automatically generated

Figure 1

**Accounting Data**

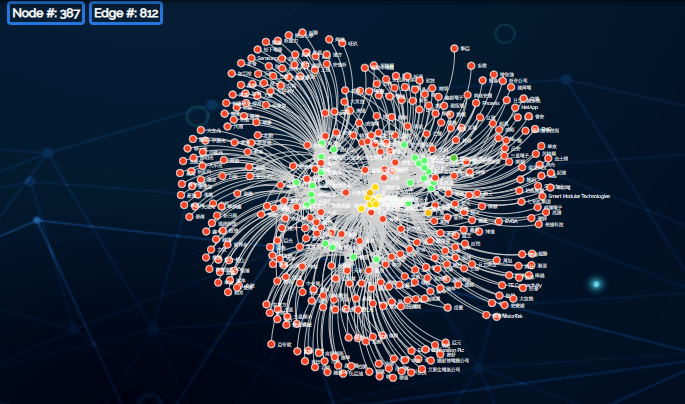
We obtained several numbers and ratios that are relevant to the credit analysis: (TBD) total asset value, book-to-market value, long-term debts/ total invested capital, debt/equity, operating income/ received capitals, net income before tax/ received capitals, gross profit margin, earnings per share, EBIT interest coverage, quick ration and Merton’s distance to default.

**News and Twitter Posts**

We put the Web crawlers onto GCP (Google Cloud Platform) to run, obtaining such huge amount of data. For news, we currently collected 14 days of news from Google News, which aggregates comprehensive news sources from all the world. For Twitter posts, we collected data in one year and take the first 50 items.

**Relationship Graph of Companies**

We used Ardi to build the graph database of companies (see image below). In this graph, we can obtain the “distance” between two companies. This information will be used to count the score of default prediction for one company, which doesn’t get much news coverage, by using the score of another company that is close to it and multiply a weight vector.



**Sentiment Analysis of news**

We trained a Naïve Bayes classifier using SnowNLP for sentiment analysis. This is a traditional method of doing sentiment analysis. Since we are doing Chinese sentiment analysis, the first step of handling Chinese articles is word cutting. We use Jieba for word cutting on a Traditional Chinese dictionary and also convert all the training data into Traditional Chinese since the news we crawled are mostly Traditional Chinese. The next part is word counting. The Naïve Bayes classifier classifies an article using the probability equation shown below, with sentiment of an article calculated by the words in positive divided by all the word appeared in the article.



We obtained a result of 85.8% on the dataset, which includes hotel reviews, online shopping reviews, and article opinions.

**Challenges & Solutions**

**Improvement on Sentiment Analysis**

We fit some article headlines into the model and tested the accuracy of sentiment; however, we found that the accuracy is only around 50%. We think this result from several reasons:

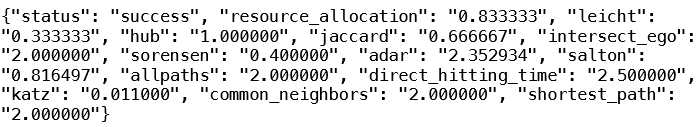
1. Word context: Naïve Bayes model consider each word to be independent, however in reality words have synonyms and polysemes. For example, bad is related to worse, good is related to excellent. These synonyms should be considered as same words of sentiment so that the probability of these words is accurate. On the other hand, “not bad” is actually a positive word, and word context of these polysemes should also be considered.
2. Training data relevance: The data we used to train includes hotel reviews, online shopping reviews, and article opinion, which are mainly irrelevant to our use case, which is financial article analysis. As a result, there are many financial vocabularies that are neutral, such as, “earnings”, “investment”, “deal”, will be incorrectly classified since these words rarely appeared in our training dataset.
3. Data cleanness: Some of the news and tweet aren’t even relative to the company. These data might not affect the accuracy of sentiment analysis but will affect the risk score prediction therefore must be removed.

Therefore, we are considering three ways to solve this problem.

1. Keep using the Naïve Bayes model: We can continue using the Naïve Bayes model by handling special cases. For example, transform synonym, polysemes cases such as “not bad”, “not good” into “good”, “bad” in advance.
2. Use LSTM models: LSTM models have proven to have great result on considering contextual information. Specifically, we might try first do word2vec, and then take the word embedding sequence into two-layer LSTM networks for sentiment prediction. By using word2vec, we solve the problem of synonyms and by using LSTM, we can solve the polysemes problem.
3. Use state-of-the-art language model – BERT: While researching methods of sentiment analysis, we found this state-of-the-art model. BERT stands for Bidirectional Encoder Representations from Transformers, which is a technique for NLP pre-training developed by Google in 2019. We discover that BERT might be utilized to do sentiment analysis using the pretrained Chinese model on Wikipedia. If we are able to get the embeddings of a sentence or an article, we might be able to do transfer learning on our sentiment analysis task.

For the dataset part, we might try finding dataset of news articles for fine tuning the news sentiment model, and use different dataset for social media posts. For the data cleanness, we might try some methods such as word bags, tf-idf to filter out unrelated news.

**Making Use of Relationship Graph**

Ardi has several analytics that can be used to evaluate closeness between two nodes. The below image is the sample of metrics that can be used. We can either get the closeness between companies and products. 

The challenge part is how we take these metrics into account when getting final risk scores. One thought is to predict a first version of risk score and then according to the weights between companies, e.g. A = 0.8\*B, we iterate over the scores until the score matches the weights. Another thought is to modify the features of a company before risk score prediction model. As the final model takes sentiment scores, financial ratios as feature, we could add features from related companies or products.

**Combining Financial Ratios with Public Opinions**

In convention, banks use Altman Z-score\*1, to do the default risk analysis. The formula is static:

*Z* = 1.2*X*1 + 1.4*X*2 + 3.3*X*3 + 0.6*X*4 + 1.0*X*5.

Since we need to take public opinions into consideration, the formula with fixed dimensions is insufficient for us to build the NLP model. Therefore, we decided to adopt the method in, “Credit Rating Change Modeling Using News and Financial Ratios”\*2, to put financial data and public opinion variables into a vector, (), running the NLP model and obtain a weight vector, W(). If the value is large (e.g., total asset), then we will take “log” to prevent the value in the weight vector from being too small.

**Evaluating the Result**

The rating report of companies is hard to get (need to pay a certain amount of money). Thus, we are going to track the trend of stock prices instead. The mindset is: If the stock price is rising in this year, that means investors are confident with the company and are willing to invest. This will generate more equity in the company and makes it less likely to default.

**Next Steps**

Training model (TBD)

**Other Enhancements (on-going)**

* Adding timestamps to sentiment scores: We believe the time when news or social media posts are posted should have different impacts on the risk. That is, older news should have less effect on the risk. Therefore, we will add timestamps on the sentiment scores and treat the scores as time series. We will probably use time series models to extract features before feeding into the final prediction model.
* Extend the collecting period of news: Since the financial statements are revealed each quarter. It’s impractical to collect news only in 14 days. Thus, we are going to collect news in each quarter and randomly choose 100 articles.

References

\*1. Altman Z-score: <https://en.wikipedia.org/wiki/Altman_Z-score>

\*2. “Credit Rating Change Modeling Using News and Financial Ratios”: <https://www.researchgate.net/publication/262323524_Credit_Rating_Change_Modeling_Using_News_and_Financial_Ratios>

Appendix I. Coding Part