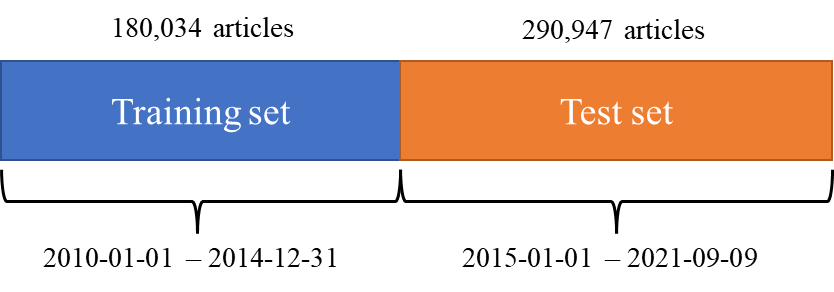
**Replication of ‘Predicting Returns with Text Data’**

Name: YANG Jiaqi Student ID: 20811352

1. **Introduction**

This paper introduces a new text-mining methodology that extracts information from news articles to predict asset returns. Unlike the traditional dictionary-based methods, this method mentioned in paper is a supervised learning process and also is fit for the problem of return prediction. There are three main steps in this method(for convenience, we name this algorithm SESTM as following): 1) isolating a list of sentiment-charged words via predictive screening, 2) assigning prediction weights to these words via topic modeling, 3) aggregating terms into an article-level predictive score via penalized likelihood.

To replicate this algorithm in our Chinses stock market, we choose the final cleaned dataset which include 472,420 financial articles and its corresponding security codes ,dates and returns. The reason why the dataset is so big is that sometimes there are lots of articles of one stock in the same trading day. Besides, two extra labels are constructed in order to consider the delayed effect of financial news. In our experiment, because the first several years’ data is relatively sparse, so our training data begins from 2010-01-01 to 2014-12-31 and testing data is from 2015-01-01 to data’s last date.



Furthermore, we try different model’s hyper-parameters to compare with dictionary-based method and do lots of algorithm extension of SESTM to fully explore best model, including applying popular word2vec and machine learning method into our model and then make a comparison with previous algorithms.

Lastly, according to predicted sentiment scores of different algorithms, 11 portfolios are constructed to verify the algorithms’ validity and check the best model.

1. **Data pre-processing**

We need to do data pre-processing before we apply these algorithms.

The steps are as follows:

1. Removing the null and duplicates values and merging the two raw dataset.
2. Screening Chinese words by regular expression where numbers, punctuation, blank space are removed.
3. Using ‘jiaba’ package for tokenization and ‘Ha gong da stopping words’ to remove stop words. Also, words are tagged with POS(Parts of Speech)

* <https://github.com/fxsjy/jieba>
* <https://github.com/goto456/stopwords>

1. Screening words by 3 kind of rules:

* method one: only select adj. and v. words (notation e.g.: av100 means 100 words tagged with a and v )
* method two: delete different POS of words for different word length (notation e.g.: ex100)
* method three: select all the words versus select the words whose word length >=2 (notation e.g.: ex100\_2p)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Length 1 | |  | Length 2 | |  | Length 3 | |  | Length 4 or above | |
| POS | explanation |  | POS | explanation |  | POS | explanation |  | POS | explanation |
| w | 标点符号 |  | nw | 作品名 |  | s | 处所名词 |  | s | 处所名词 |
| r | 代词 |  | u | 助词 |  | r | 代词 |  | LOC | 地名 |
| ns | 地名 |  | m | 数量词 |  | LOC | 地名 |  | f | 方位名词 |
| f | 方位名词 |  | t | 时间 |  | nt | 机构名 |  | ORG | 机构名 |
| p | 介词 |  | TIME | 时间 |  | ORG | 机构名 |  | n | 普通名词 |
| c | 连词 |  | nr | 人名 |  | f | 方位名词 |  | nz | 其他专名 |
| q | 量词 |  | PER | 人名 |  | n | 普通名词 |  | PER | 人名 |
| n | 普通名词 |  | nz | 其他专名 |  | nz | 其他专名 |  | nr | 人名 |
| nr | 人名 |  | n | 普通名词 |  | nr | 人名 |  | t | 时间 |
| m | 数量词 |  | q | 量词 |  | PER | 人名 |  | TIME | 时间 |
| u | 助词 |  | ORG | 机构名 |  | t | 时间 |  | m | 数量词 |
|  |  |  | f | 方位名词 |  | TIME | 时间 |  | nw | 作品名 |
|  |  |  | LOC | 地名 |  | m | 数量词 |  |  |  |
|  |  |  | r | 代词 |  | nw | 作品名 |  |  |  |
|  |  |  | s | 处所名词 |  |  |  |  |  |  |

1. constructing 2 extra return labels:

* specret[t:t+1] : means that the sum of ‘specret’ over time t and t+1.
* specret[t+2:t+6] : means that the sum of ‘specret’ over time t+2 and t+6.

1. **Benchmarks**

We use dictionary-based method as benchmarks and 2 dictionary are applied in our experiment.

* <https://github.com/dictionaries2020/SentimentDictionarie>
* <https://github.com/MengLingchao/Chinese_financial_sentiment_dictionary>

Besides, 12 scores are defined to describe articles’ sentiment degree.

* Score 1= # positive words/# of words excluding stop words and symbols etc.
* Score 2=(-1)\* # negative words/# of words excluding stop words and symbols etc.
* Score 3= Score 1+score 2.
* Score 4-6, change the denominator to (#positive+#negative words) in score 1-3.
* Score 7-9: use # of sentences instead of # words in Score 1-3.
* Score 10-12: use # of sentences instead of # words in Score 4-6

By calculating the pearson and spearman correlation coefficients to test model’s performance.

Covariance:

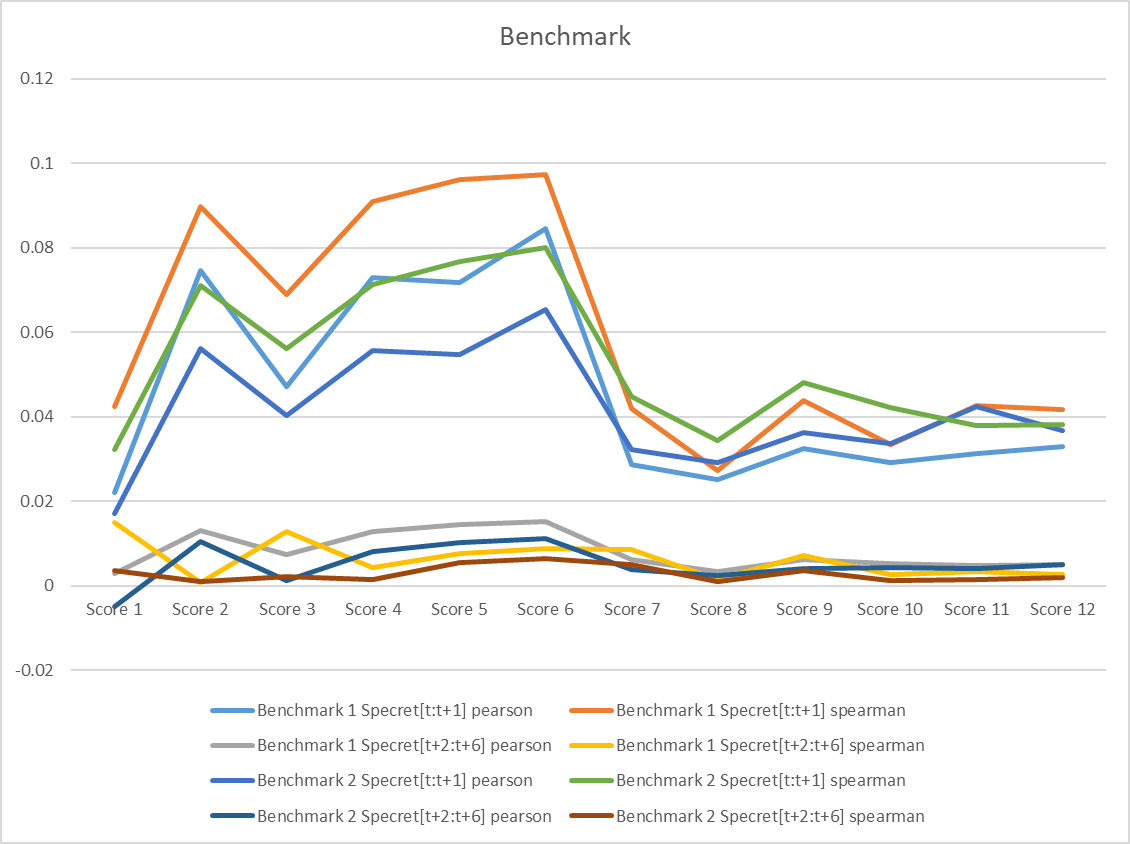
Pearson correlation coefficient:

spearman correlation coefficient:

where is the rank of in .

The results are below:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Benchmark 1 | | | | Benchmark 2 | | | |
| Specret[t:t+1] | | Specret[t+2:t+6] | | Specret[t:t+1] | | Specret[t+2:t+6] | |
| pearson | spearman | pearson | spearman | pearson | spearman | pearson | spearman |
| Score 1 | 0.0221 | 0.0423 | 0.0028 | 0.0150 | 0.0171 | 0.0322 | -0.0050 | 0.0036 |
| Score 2 | 0.0745 | 0.0896 | 0.0130 | 0.0008 | 0.0562 | 0.0711 | 0.0103 | 0.0010 |
| Score 3 | 0.0471 | 0.0690 | 0.0073 | 0.0129 | 0.0402 | 0.0562 | 0.0011 | 0.0022 |
| Score 4 | 0.0728 | 0.0908 | 0.0128 | 0.0043 | 0.0557 | 0.0712 | 0.0080 | 0.0014 |
| Score 5 | 0.0717 | 0.0961 | 0.0145 | 0.0076 | 0.0546 | 0.0768 | 0.0102 | 0.0054 |
| Score 6 | 0.0845 | 0.0973 | 0.0152 | 0.0087 | 0.0653 | 0.0799 | 0.0112 | 0.0064 |
| Score 7 | 0.0287 | 0.0420 | 0.0061 | 0.0085 | 0.0321 | 0.0447 | 0.0039 | 0.0050 |
| Score 8 | 0.0250 | 0.0272 | 0.0032 | 0.0015 | 0.0290 | 0.0344 | 0.0024 | 0.0009 |
| Score 9 | 0.0325 | 0.0439 | 0.0062 | 0.0071 | 0.0361 | 0.0481 | 0.0040 | 0.0036 |
| Score 10 | 0.0291 | 0.0334 | 0.0053 | 0.0025 | 0.0336 | 0.0422 | 0.0042 | 0.0011 |
| Score 11 | 0.0312 | 0.0427 | 0.0048 | 0.0033 | 0.0423 | 0.0379 | 0.0040 | 0.0015 |
| Score 12 | 0.0328 | 0.0416 | 0.0050 | 0.0027 | 0.0368 | 0.0381 | 0.0049 | 0.0018 |



From the results, we can see that score 6 is relatively higher than other scores. Thus, score 6 is chose as benchmark for later experiment. What’s more, the scores of specret[t+2:t+6] are generally lower than specret[t:t+1], so the delayed effect of news is not very long.

1. **SESTM**
2. We first introduce our SESTM algorithm detailly.

Step 1: Screening for Sentiment-Charged Words

For each word , let

For a proper threshold , and to be determined,

Where is the total count of articles in which word appears.

Step2: Learning Sentiment Topics

Sort the returns in ascending order, let

So we get the positive and negative topic .

Step3: Scoring New Articles

Assuming the sentiment-charged words follow the multinomial distribution:

Then we using MLE(Maximum Likelihood Estimation) method to get with penalty parameter:

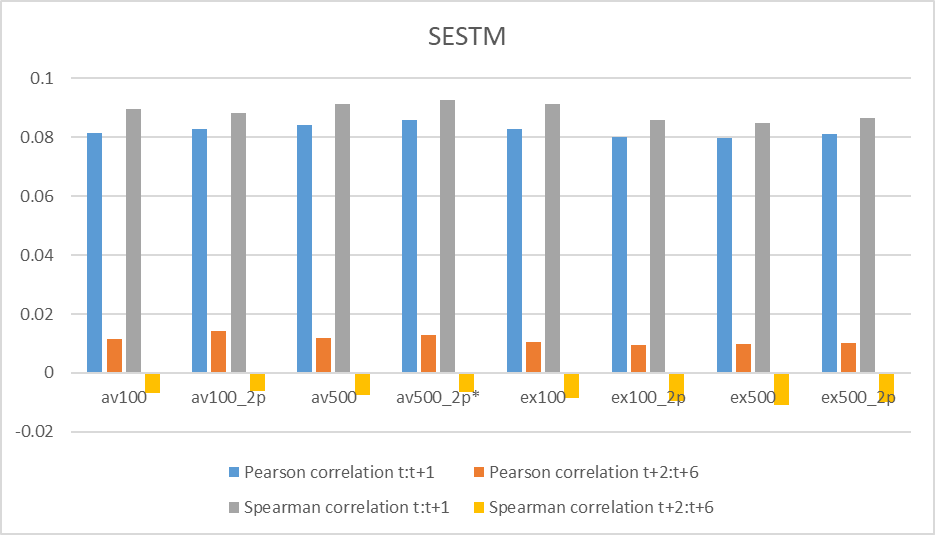
1. There are lots of hyper-parameters in SESTM, but both the results of parameters’ difference is not much significant. So we choose

* : set the k value to choose high frequency words, 88%-94% quantile. The higher , the more frequent words.
* : set the number of words in each topic. We choose 100 or 500.

According to above model’s parameters, we mainly do following variations:

* different number of sentiment charged words, 100 or 500.
* words with different POS tag(a+v or different word length table)
* different return labels(specret[t:t+1] or specret[t+2:t+6]).

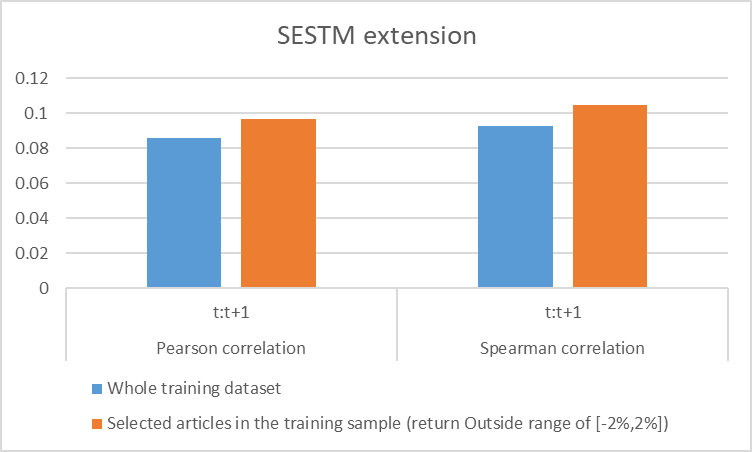
Then, we get the results:

****

As we can see, av500\_2p model performs best and we use av500\_2p for later comparison. Also, the spearman correlation coefficients of label ‘specret[t+2:t+6]’ even get negative, which verifies this label is not suitable for modeling again. What’s more, we can find that the correlation coefficient values become stable and generally higher than our benchmark, which shows that SESTM algorithm is better than dictionary-based method.

1. SESTM extension

Considering the label’s noise, which means that the small values of return label could be noise but can also make an effect on model’s training, we only choose the extreme returns which is outside the range of [-2%,2%] as our training data. Then we compare with the original method and the results is below:



From the above graph, we can conclude that there are noise in our return labels because the pearson and spearman correlation coefficients of training data which is outside[-2%,2%] is higher than original model.

Besides, we also do other extension such as applying popular word2vec into this experiment. The detail is below.

1. **Word2vec**

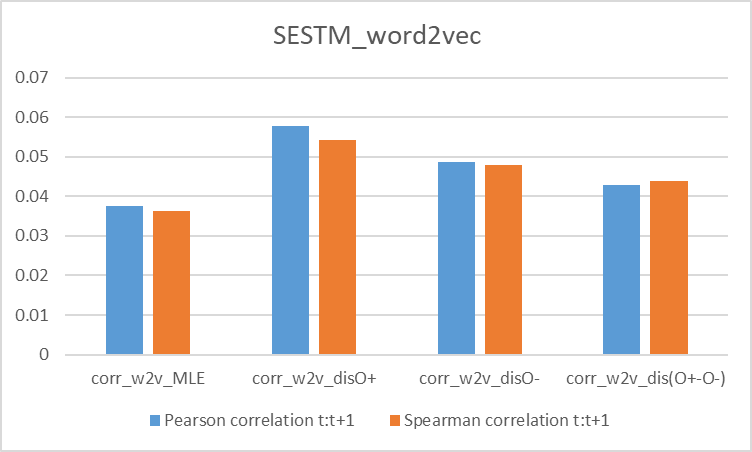
Word2vec is a set of related models used to generate word vectors. These models are shallow, two-layer neural networks trained to reconstruct linguistic word texts. The network is represented by words, and the input words in adjacent positions are guessed. After the training, the word2vec model can be used to map each word to a vector to represent the relationship between words, which is the hidden layer of the neural network.

In our experiment, we use pre-trained Chinese word vector in github website (<https://github.com/Embedding/Chinese-Word-Vectors>). Because every sentiment-charged word has a 300 dimension vector, the article vectors are calculated by averaging the word vectors.

Both SESTM and machine learning method could use word2vec.

For SESTM:

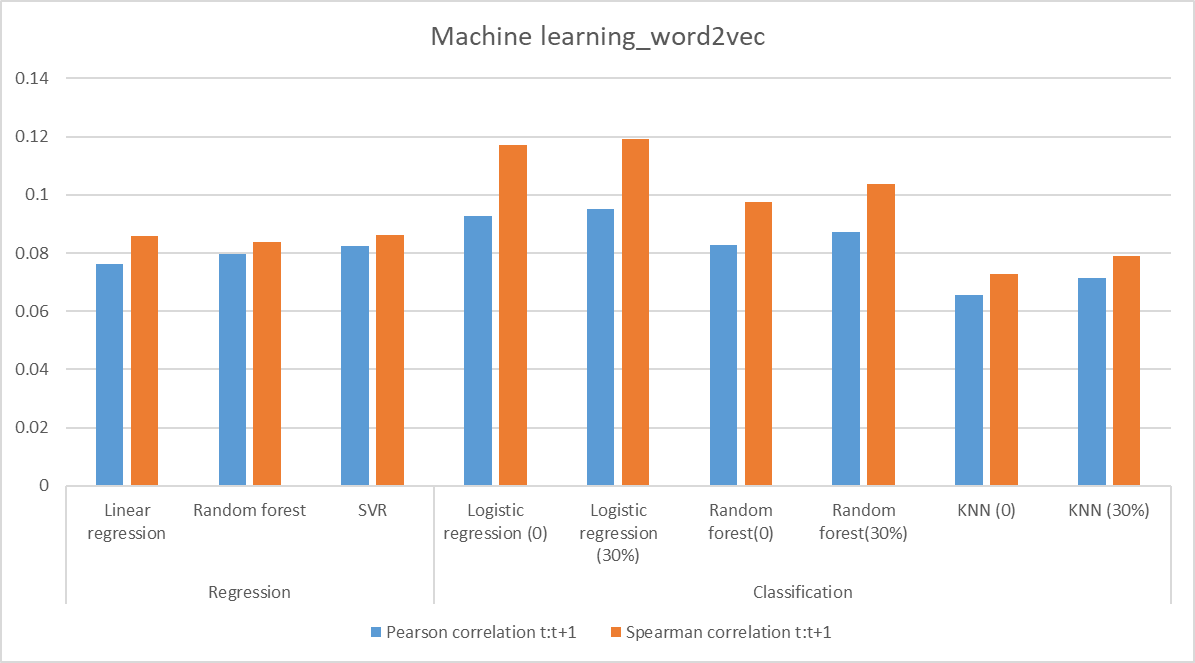
* Instead of using bag of words, we let the article vectors to apply SESTM and then still derive the predicted sentiment scores from MLE method.
* Another method to get sentiment scores is to calculate the distance between article vectors and the (positive topic) , (negative topic) and the . Then finally we get three values.



According to the result, although corr\_w2v\_disO+ performs best among the four models, its values(0.05783) still very low compared with the previous results. So we don’t use this method for later portfolio construction.

For supervised machine learning method:

* The 300 dimension article vectors are the feature inputs and the return labels are the labels.
* SVR(support vector regression), Random forest and linear regression are used as regression algorithms.
* KNN, Logistic and Random forest are applied as classification algorithms.
* For classification problem, we also separate the top 30% returns as label 1 and the bottom 30% returns as label 0 to test if there are noise in the labels.



According to the above results, logistic regression method for classification performs best. And the idea of choosing top 30% and bottom 30% as training data indeed improve the models’ performance.

1. **Portfolio construction**

Our main idea of constructing portfolio is to use previous 30, 60 and 90 days data to evaluate all stocks by our algorithms ,then divide them into 10 groups and finally determine following 30 days investment behavior.

Firstly, we should verify if the previous 30, 60 ,90 days’ performance could make an effect on following months. The results are as following:

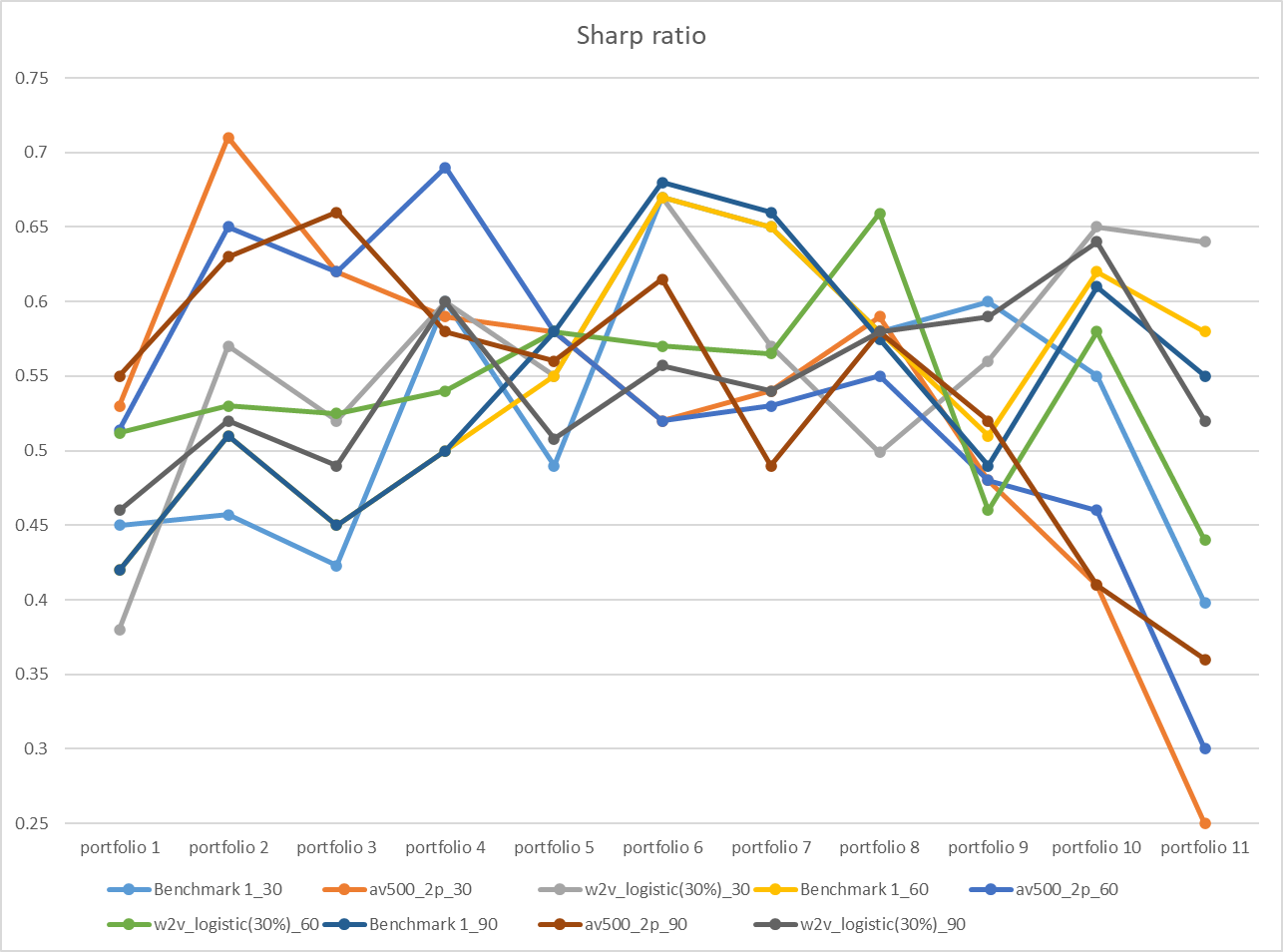
|  |  |  |  |
| --- | --- | --- | --- |
| Pearson/Spearman | Pre 30 days | Pre 60 days | Pre 90 days |
| Benchmark 1 | 0.0119/0.0187 | 0.0123/0.0117 | 0.0109/0.0112 |
| Benchmark 2 | 0.0067/0.0087 | 0.0066/0.0087 | 0.0045/0.0052 |
| av500\_2p | 0.0087/0.0067 | 0.0097/0.0076 | 0.0079/0.0081 |
| w2v\_logistic(30%) | 0.0145/0.0154 | 0.0163/0.0174 | 0.0169/0.0175 |

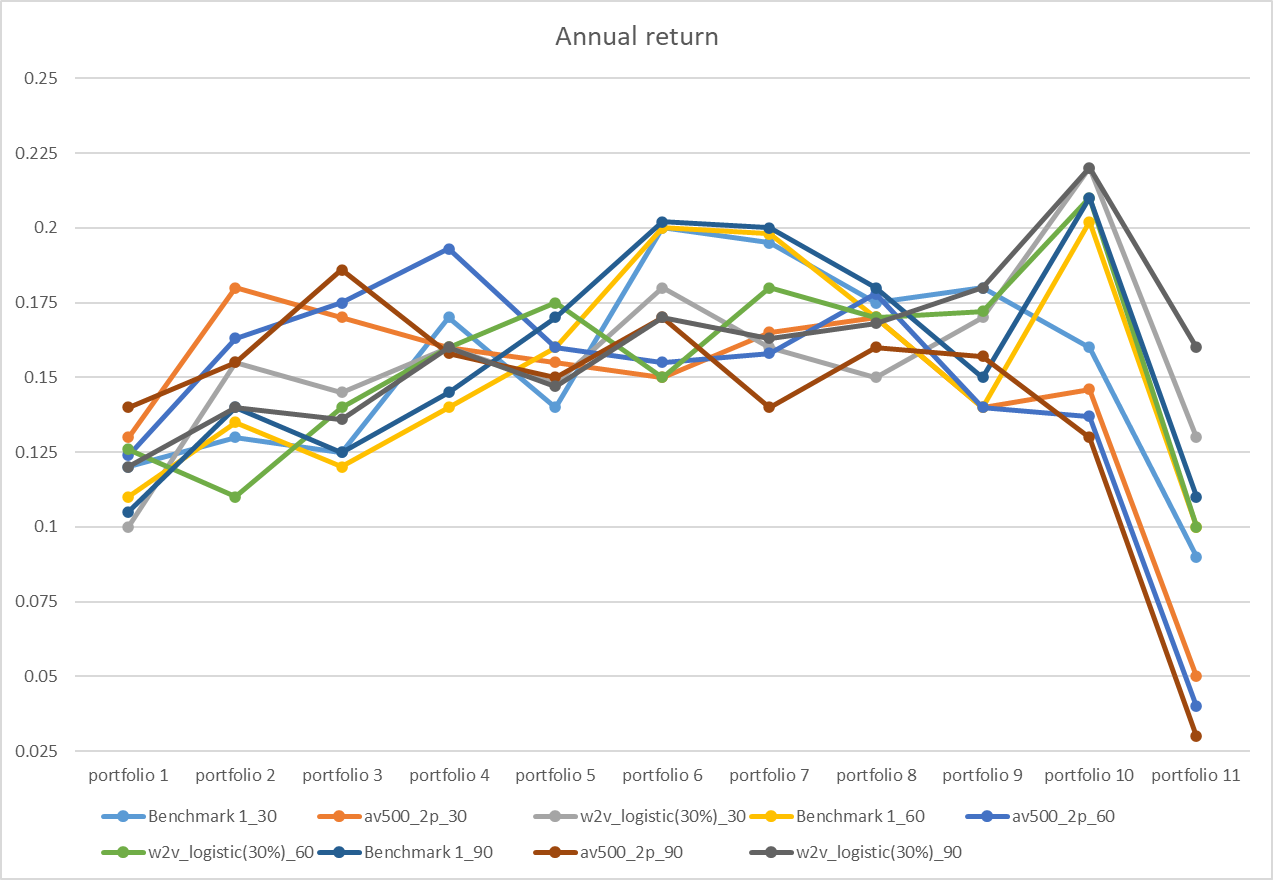
From above results, we can conclude that the previous 30, 60, 90 days indeed have more or less influence on later 30 days. And we can mention that the logistic regression using word2vec performs best among these 4 methods.

Then the detail of portfolio construction as following steps:

* Using the previous trained models to predict the each stock’s pre 30, 60, 90 days return.
* Sorting these stocks into 10 groups. The group 1 means the bottom 10% stocks and the group 10 means the top 10% stocks. Besides, we add group 11 which buys the group 10 and sells the group 1.
* According to the return of 11 groups’ following 30 days, we derive the following 30 days’ portfolio return.
* Rolling the window and finally get the sharp ratio and annual return of 11 portfolios.

In our experiment, we use Benchmark 1, av500\_2p and w2v\_logistic(30%) as our 4 different models. The results as below:





From above results, we can find that despite different models, the annual return has an ascending trend among the portfolio 1-10 and suddenly drops at portfolio 11. For sharp ratio, the first 10 portfolio fluctuate and then drop at portfolio again. The reason why this phenomenon needs to be explored further. And the w2v\_logistic(30%) performs relatively better than other models.

1. **Summary**

There are five steps in our experiment including data pre-processing, benchmark, SESTM, word2vec, portfolio construction. In each part, we can get lots of conclusions by comparison. Therefore, every conclusion is showed as below:

Data pre-processing:

* Get the cleaned data.
* Tokenization and screen words via different rules.
* Construct two return labels.

Benchmark:

* Score 6 performs best among 12 scores.
* Return label ‘specret[t+2:t+6]’ has a weak impact on prediction, so the delayed effect of financial news is not very long.

SESTM:

* Model av500\_2p surpass other models.
* SESTM is better than dictionary-based methods.
* There are noise in return labels because the result of return label outside [-2%,2%] is better than original.

Word2vec:

* Although corr\_w2v\_disO+ performs best among the four distance models, its values(0.05783) still very low compared with the previous results.
* After using machine learning method, logistic regression method for classification performs best. And the idea of choosing top 30% and bottom 30% as training data indeed improve the models’ performance.

Portfolio construction:

* Previous 30, 60, 90 days indeed have more or less influence on later 30 days.
* According to sharp ratio and annual return, the w2v\_logistic(30%) model performs relatively better than other models.