Constructing Financial Sentimental Factors in Chinese Market Using Techniques of Natural Language Processing

Junfeng Jiang, Jiahao Li

Abstract

In this paper we develop an integrated algorithm to evaluate the sentimental factors of Chinese markets. By focusing on news from several influential financial websites, we use techniques of NLP(Natural Language Processing) under Chinese context to compute a sentimental factor. This sentimental factor we compute proved to have significant correlationship with Chinese market, which makes it a useful factor when making investment decisions.

1 Introduction

Natural language processing, as one of the most promising fields of machine learning, has achieved great development recently and has been used in lots of aspects in the society. Many researches also implemented the technique of NLP in financial market. The difficulty of using natural language is that it is not structural data. Finding a way to process such kinds of non-structural data is the main focus of NLP. A lot of models have been demonstrated to do well in turning natural language data into numerical data. With the implementation of these models, it becomes possible and easier to make use of natural language data.

Some models are based on the idea of naive Bayesian. The logic behind these models is: words that show the same kind of sentiment will appear simultaneously more frequently. Researches choose several words as the label words. By analyzing the words used in a large amount of texts and researching on the relationship between the frequency of these label words and the frequency of other words, it becomes possible to cluster words. For any given texts, it is able to use these words to evaluate the sentiment behind. Researches have proved this kind of methods can successfully evaluate the sentiment of texts like twitter or news. By taking advantage of the sentiment, investors can make appropriate investment decisions.

However, this kind of methods have their own limitations. The main limitation is they focus on too limited words. Some new words showing similar sentiment but do not appear frequently will not be considered. Sometimes, these words are also signifiant when analyzing the sentiment of a text. The lost of information can have great damage to the accuracy of the evaluation.

This study is aiming to put eyes on more words to analyze sentiment. The specific purposes of this study are as followed: (1) Download news from several influential financial websites automatically. (2) Make a pretreatment on the news we crawl from the Internet. (3) Find a way or some algorithms to analyse this per-treated text data, and finally compute a sentimental factor of each day with the news at that day. (4) Choose a proper criterion to analyze the correlationship between the sentimental factor and the market trend, and judge whether our factor is useful in financial investment.

The remainder of this paper is organized as follows. Section 2 describes the research background and related works of Jieba, Word2vec and WordNet. Section 3 shows the methodology and the data for analysis. Section 4 contains the experimental result and discussion. Finally, in Section 5, we proposed our conclusions.

2 Related Works

2.1 Jieba

The tokenization of Chinese is much more complicated than English. To tokenize English words, we just need to split words in sentence by blank or punctuation. Chinese doesn't have blank between words. An additional step of tokenizing is, therefore, needed.

Jieba(Chinese for "to shutter") Chinese text tokenization is a Chinese word tokenization module. The algorithm of Jieba is probability language modeling. It generates a trie tree based on a dictionary transcendentally and also calculate the frequency of words in the dictionary. When dealing with the sentence that is needed

to be tokenized, it generates a DAG(Directed Acyclic Graph) to record every possible tokenization.A DAG is a dictionary, where the keys are the starting position of a word in the sentence and the values are lists of possible ending position.

For every possibe words in the DAG, Jieba calculates their probability based on the transcendental dictionary. Then it find the paht with the largest probability from the right side of the sentence to the left side. This largest probability path gives us the most possible tokenization.

In the case where the sentence includes words that are not in the dictionary, Jieba uses HMM(Hidden Markov Model) and Viterbi algorithm to tokenize. Every character has four conditions based on its possitlbe position in a word: B(Begin), M(Middle), E(End) and S(Single). The process of tokenizing words not in the dictionary is based on their conditions mainly. With three probability tables from the training of a large amount of texts, Jiaba then applies Viterbi algorithm to calculate the most posbbile condition of a word and uses the conditions chain to tokenize.

2.2 Word2vec

In 2013, Google published a powerful tool named word2vec. It contains two models, one is the Skip-gram, another is Continuous bag of words(CBOW). With the word2vec model, we can turn a specified word into a calculable numeric vector. To speak of, moreover, it can well express the degree of similarity and analogy between two different words.

Since word2vec have been published, it is widely applied in Natural Language Processing, and its original models and training methods also enlighten many word embedding models and algorithms on the following days.

2.2.1 Skip-gram

In Skip-gram, we focus on one word, and use it to predict which words will appear around it.

For example, the boy adores that girl, we can achieve five background words like the boy adores that girl easily because we have blanks between every two words. Let adores be the center word, and set window size equals to 2, then, in Skip-gram, what we are interested in is the conditional probabilities of each background word under the given center word, where the background words is apart from the center word in two words. That is the mainly idea of Skip-gram. Lets describe the Skip-gram model in a strict mathematical language.

Assume that size of the set of dictionary index D is $\big|\mathrm{D}\big|$, and denoted as D=$\{1,2,...,\big|\mathrm{D}\big|\}$. Given a text sequence with the length of T, and the t^{th} word denoted as $w^{(t)}$.When window size equals to m, Skip-gram requires that we should maximize the total of all conditional probabilities of each background word that is apart from the center word in m words under arbitrary center word.

$$\prod_{t=1}^{T} \prod_{-m\leq j\leq m, j\neq 0, 1\leq t+j\leq \big|T\big|} P(w^{(t+j)}\big|w^{(t)})$$

So, the likelihood function is,

$$\sum_{t=1}^{T} \sum_{-m\leq j\leq m, j\neq 0, 1\leq t+j\leq \big|T\big|} logP(w^{(t+j)}\big|w^{(t)})$$

Maximizing the likelihood function above is equal to minimize the following loss function,

$$-\frac{1}{T}\sum_{t=1}^{T} \sum_{-m\leq j\leq m, j\neq 0, 1\leq t+j\leq \big|T\big|} logP(w^{(t+j)}\big|w^{(t)})$$

Denote the vectors of center words and background words with $\mathbf{v}$ and $\mathbf{u}$, that is, as for a word with index i, $\mathbf{v}_i$ and $\mathbf{u}_i$ are the vectors when it is as center word and background word. And the parameters of model we want to train are the two kinds of vectors of every words. In order to implantate the model parameters into loss function, we should express the conditional probabilities of background word under given center word with model parameters. Assume that generating each background words is independent mutually when center word is given, then as for the center word w_c and the background word w_b, b,c are the indexes of them in the dictionary. Such that, the probability of generating background word w_b under the given center word w_c can be defined by softmax function, as

$$P(w_b\big|w_c) = \frac{exp(\mathbf{u}_b^T\mathbf{v}_c)}{\sum_{i\in D} exp(\mathbf{u}_i^T\mathbf{v}_c)}$$

With derivation, we achieve the gradience of the conditional probability above,

$$\frac{\partial logP(w_b\big|w_c)}{\partial \mathbf{v}_c} = \mathbf{u}_b - \sum_{j\in D}\frac{exp(\mathbf{u}_j^T\mathbf{v}_c)}{\sum_{i\in D} exp(\mathbf{u}_i^T\mathbf{v}_c)}\mathbf{u}_j$$

namely,

$$\frac{\partial logP(w_b|w_c)}{\partial \mathbf{v}_c} = \mathbf{u}_b - \sum_{j\in D} P(w_j|w_c)\mathbf{u}_j$$

Then, we can solve this by Gradient Descent or Stochastic Gradient Descent iteratively, and finally achieve the word vectors v_i and u_i $i = 1, 2, ..., |D|$ of every single words when it is as center word and background word, when the loss function reaches to minimum.

If the length of text sequence T is too long, we can sample a rather short subsequence randomly to calculate the loss about this subsequence in each epoch, in order to find out an approximate solution.

In general, we will use the central word vector of Skip-gram as the word vector of each word in natural language processing application.

2.2.2 Continuous Bag of Words

CBOW is familiar with Skip-gram, this model predicts the central word with the background words around it in a text sqeuence. For example, the boy adores that girl, we can achieve five background words like the boy adores that girl. Let adores be the central word again, and set window size equals to 2, then, in CBOW, what we are interested in is the conditional probabilities of generating the given central word under all the background words which are apart from the central word in two words. That is the mainly idea of CBOW. Assume that size of the set of dictionary index D is $|D|$, and denoted as D={1,2,...,|D|}. Given a text sequence with the length of T, and the t^{th} word denoted as $w^{(t)}$.When window size equals to m, CBOW requires that we should maximize the total of all conditional probabilities of generating the arbitrarily given central word under all the background words which are apart from the central word in m words.

$$\prod_{t=1}^{T} P(w^{(t)}|w^{(t-m)}, ..., w^{(t-1)}, w^{(t+1)}, ..., w^{(t+m)})$$

where m is the window size, and we should insure that (t-m+j)∈[1,|T|],j∈[0,2m]. Therefore, the likelihood function is,

$$\sum_{t=1}^{T} logP(w^{(t)}|w^{(t-m)}, ..., w^{(t-1)}, w^{(t+1)}, ..., w^{(t+m)})$$

Maximizing the likelihood function above is equal to minimize the following loss function,

$$-\sum_{t=1}^{T} logP(w^{(t)}|w^{(t-m)}, ..., w^{(t-1)}, w^{(t+1)}, ..., w^{(t+m)})$$

We still use the notation when we discuss Skip-gram model. Now, as for the central word v_c and its background words w_{b0},w_{b1},...,$w_{b\cdot 2m}$, such that, the probability of generating the given central word w_b under all the background words w_{b1},w_{b2},...,$w_{b\cdot 2m}$ can be defined by softmax function as,

$$P(w_c|w_{b0}, w_{b1}, ..., w_{b\cdot 2m}) = \frac{exp(\frac{\mathbf{v}_c^T(\mathbf{u}_{b0}+\mathbf{u}_{b1}+...+\mathbf{u}_{b\cdot 2m})}{2m})}{\sum_{i\in D} exp(\frac{\mathbf{v}_i^T(\mathbf{u}_{b0}+\mathbf{u}_{b1}+...+\mathbf{u}_{b\cdot 2m})}{2m})}$$

With derivation, we achieve the gradience of the conditional probability above,

$$\frac{\partial logP(w_c|w_{b0}, w_{b1}, ..., w_{b\cdot 2m})}{\partial \mathbf{u}_{bi}} = \frac{1}{2m}(\mathbf{v}_c - \sum_{j\in D} \frac{exp(\frac{\mathbf{v}_c^T(\mathbf{u}_{b0}+\mathbf{u}_{b1}+...+\mathbf{u}_{b\cdot 2m})}{2m})}{\sum_{i\in D} exp(\frac{\mathbf{v}_i^T(\mathbf{u}_{b0}+\mathbf{u}_{b1}+...+\mathbf{u}_{b\cdot 2m})}{2m})} \cdot \mathbf{v}_j)$$

namely,

$$\frac{\partial logP(w_c|w_{b0}, w_{b1}, ..., w_{b\cdot 2m})}{\partial \mathbf{u}_{bi}} = \frac{1}{2m}(\mathbf{v}_c - \sum_{j\in D} P(w_c|w_{b0}, w_{b1}, ..., w_{b\cdot 2m}) \cdot \mathbf{v}_j)$$

As the same of Skip-gram, we can also solve this by Gradient Descent or Stochastic Gradient Descent iteratively, and finally achieve the word vectors v_i and u_i (i=1,2,...,|D|) of every single words when it is as center word and background word, when the loss function reaches to minimum.

If the length of text sequence T is too long, we can sample a rather short subsequence randomly to calculate the loss about this subsequence in each epoch, in order to find out an approximate solution.

In general, we will use the background word vector of CBOW as the word vector of each word in natural language processing application.

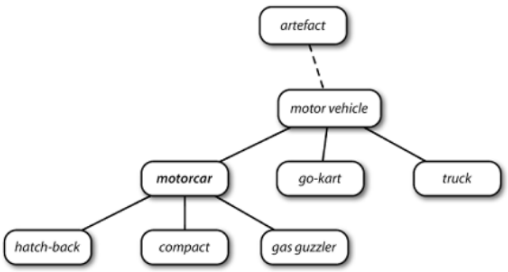

Figure 1: An Example of the Structure of WordNet

2.3 WordNet

WordNet is a large lexical database of English. In WordNet, synsets are interlinked by means of conceptual-semantic and lexical relations. The main relation between words in WordNet is synonym. By using a network to show the relation between words, WordNet helps us find synonym of words and also shows how two words are similar with each other in the perspective of meanings.

Having the properties above makes WordNet more reliable to analyze the sentiment. Chinese Open Wordnet is a large scale, freely available, semantic dictionary of Mandarin Chinese inspired by WordNet. It has the same structure and principles as WordNet but based on Chinese. It contains 42,315 synsets, 79,812 senses and 61,536 unique words and the construction is still ongoing. Our research mainly uses COW to compute the Senti-score of each single words.

3 Methodology

3.1 Pretreatment

For any given news or texts, we use Jieba to tokenize them first. In addition to turn the whole article into a bag of words, we also needs to eliminate some words and symbols that are meaningless when doing our analysis. Such words and symbols are called stop words. They often appear very frequently in almost every news or texts but they themselves alone make no sense at all, which will interfere our evaluation. With the process of removing the stop words, we are able to ignore words that are not important and only focus on words that influence the sentiment of the article most.

3.2 Text Analysis

3.2.1 Morphological Similarity

At the beginning, we should train our word2vec model achieving more word vectors as much as possible (Theoretically we can achieve word vectors of all words as long as we train word2vec with a large enough corpus, but in fact, we cannot always have such a corpus). For the generality, we choose zh-wiki as our training corpus.

It should be noted that we cannot remove stop words on training corpus because stop words can be significant describing a specified central word.

Set an appropriate large length of word vector in order to make words linearly separable in the hyperplane as far as possible. And we can set a proper threshold of frequence to omit some unfamiliar words, saving memory without losing any vital information.

After training, we achieve a word2vec model, meanwhile we get to know the word vectors of all words. We should just look up a word vector of a word in the model(like a dictionary).

Now, we can calculate the morphological similarity of two words with their cosine distance. For example, let's consider two words w_1 and w_2, and normalize as $w_1^{'}$ and $w_2^{'}$, so

$$distance = \frac{w_1 \cdot w_2}{\|w_1\|\|w_2\|} = w_1^{'} \cdot w_2^{'}$$

which indicates the morphological similarity of two words w_1 and w_2.

Note: Though now we can estimate the similarity of two words with word2vec, what we achieve above is only the similarity morphologically. That is to say, we can just find out which words are similar with a specified word morphologically, but do not know their meanings. Let's take an easy example, consider two words 'increase' and 'decrease', we can often read such a sentence 'The .DJI increases by 5 percent today' on the financial news, and you will find it certainly possible to exchange 'increase' into 'decrease' in this sentence without any difficulty. As known in **2.2**, the word vectors of this two words will approximately be the same, which may make us confused in determine their respective Senti-scores. But, word2vec does help us to find out some words familiar with a given word in a manner.

3.2.2 Semantics Similarity

As mentioned in 2.3, WordNet uses trees to record words. The structure of trees defines distances naturally. When

computing the semanticc similarity of words using WordNet, we use the shortest path linking the two nodes representing the words to compute. Take the reciprocal of the shortest path of the two nodes as the similarity. The similarity between a word and itself is 1. We also define the similarity between two words is 0 if there is no path linking them. With definitions above, the semantics similarity we compute will be a value between 0 and 1. The larger the value is, the more similar the two words are semantically.

3.3 Senti-score Computation

3.3.1 Label Words

We use a label word set to evaluate the sentiment of a single word. The idea is that the similarity of a word and another word we set as a criterion represent its sentiment. We choose 100 words that appear in financial news frequently and also have specific sentiment as our label words. In order to make the computation more fair, we choose 50 words that have positive attitude toward the market and the other 50 have the opposite sentiment. Before computing the Senti-score of a word, we first compute both the morphological similarity and the semantics similarity with the label words.

3.3.2 Senti-score of Words

With the computation above, we will have a vector of 200 dimensions for every word. The first 100 dimensions are the Word2vec similarity with the 100 words in the label word set. They show the morphological similarity with the label words. The last 100 dimensions, on the other hand, are the WordNet similarity with the 100 words in the label word set, which represent the semantic similarity with the label words. Since the words in the label word set show attitudes toward the market, based on the similarity above, we can evaluate the attitude of a word, which is the sentiment. To represent the sentiment quantitatively, we define a value called Senti-score.

In our research, we use the similarity vector to compute the Senti-score of every word. In specific, the process includes the following steps:(1)Use the Word2vec similarity and the WordNet similarity respectively. Then use collaborative filtering to classify it as positive word or negative word. (2)Use Word2vec similarity to compute the Senti-score. The specific process and the reasons behind are as followed.

For a word, when we consider its similarity with words in the label word set, there are two kinds of similarity we should think about. The first is the morphological similarity. The second is the semantic similarity. Collaborative filtering can help us find the several most similar words morphologically and semantically. First of all, we use the first 100 dimensions, which are the Word2vec similarity, to find the 9 most similar words. We find them by picking up the 9 words that have tha largest value in the first 100 dimensions. The words we find in this way will be the 9 words that are the most morphologically similar to the target word we need to compute. Then we compare the WordNet similarity of these 9 words, which are in the last 100 dimensions. From these 9 words, we pick up 3 that have the largest WordNet similarity. These 3 words will be the 3 most similar to the target word both morphologically and semantically.

With the 3 words we can judge whether the target word is positive or negative. However, it is not enough if we just label it as +1 or -1. The reason is that the degree of positive or negative can be different for the same kind of words. A positive word can be more positive than another positive word. Therefore, we need to compute a score for the word. We use the absolute value of the Word2vec similarity of the 3 words. For the positive ones time one and for the negative ones time negative one. Add them up and we got the Senti-score of target word.

A problem we have here is that the words included in the COW is far less than words in the Word2vec model we trained. In some cases, the last 100 dimensions of the word cannot be computed. In these cases, we use the Word2vec similarity only to evaluate. For these words, in the second step of collaborative filtering, we use the Word2vec similarity instead. For the same reason, finally the Senti-score of every word is computed by Word2vec similarity.

Up to now, we can compute the Senti-score of a single word.

3.3.3 Senti-score of Articles and Sentimental Factor

Use the Senti-scores of words above, we can compute the Senti-score of every piece of news. By adding up all the Senti-score of news in a day, we will get the sentimental factor we need. Details are as follows.

After pretreating the news, we will get a word bag. It is traight-forward to compute the similarity vector of each word in the word bag and use these vectors to compute the Senti-scores. However, it is not efficient and not necessary. Too many words will be computed for many times. The fact is, if we compute the Senti-socres of the most commonly used words in advance, we can get the

Senti-score of an article much quicker and much more efficient.

We use over 50,000 common words in *The Common Vocabulary of Modern Chinese* publiced by Commercial Press and pick up words that are not in these over 50,000 words in 3,000 news. Collect these words and we get a common word set of around 100,000 words. We compute the Senti-scores of words in common word set in advance. When we compute the Senti-score of an article, we will just need to look up the common word set. Some words may be ignored if we use this method, but our experiments show that words not in the common word set will only be around 5% of words in an article. Ignoring the 5% will not influence our accuracy that much.

Use the method above we can compute Senti-score of news. We compute the average Senti-score of a day and use it as the sentimental factor that day. The sentimental factor computed in this way can show the sentiment of the market effectively.

3.4 Correlation Analysis

We need a criterion to evaluate the efficiency of the sentimental factor. The main criterion is to do a linear regression using the sentimental factor and the market trend. Assume that the market sentiment has positive correlation with the market trend, then the significance of the regression result will be able to show the efficiency of the sentimental factor.

4 Experimental Results and Discussion

We calculate the Senti-score of 181 days, and make a correlation analysis about Senti-score and some market indexes.

Linear regression is carried out with them, and we can see that all of the coefficients successfully pass significance test.

And the Pearson correlation coefficient is xxx, which states they are moderately correlative.

5 Conclusion

In this paper we develop an algorithm to compute a sentimental factor of Chinese markets and demonstrate that this factor has significant correlationship with Chinese market. This factor provides us with a new way to make investment decisions.

The method to compute a sentimental factor is the main contribution of this paper. It will help us even more if we are able to compute sentimental factors for every financial product. Also, a combination of this sentimental factor and tradictional financial factors may help us to make even better investment decisions. Looking forward to seeing related research.

6 References

[1]George A. Miller (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

[2]Christiane Fellbaum (1998, ed.) WordNet: An Electronic Lexical Database. Cambridge, MA: MIT Press.

[3]Building the Chinese Wordnet (COW): Starting from Core Synsets. In Proceedings of the 11th Workshop on Asian Language Resources: ALR-2013 a Workshop of The 6th International Joint Conference on Natural Language Processing (IJCNLP-6). Nagoya. pp.10-18.

[4]Theoretical and Practical Issues in Creating Chinese Open Wordnet (COW). Paper presented at The 7th International Conference on Contemporary Chinese Grammar (ICCCG-7), Nanyang Technological University, Singapore.

[5]Linking and extending an open multilingual wordnet. In 51st Annual Meeting of the Association for Computational Linguistics: ACL-2013. Sofia. 1352C1362

[6]Rao T, Srivastava S. Analyzing Stock Market Movements Using Twitter Sentiment Analysis[C]// International Conference on Advances in Social Networks Analysis and Mining. IEEE Computer Society, 2012:119-123.

[7]Kim Y. Convolutional Neural Networks for Sentence Classification[J]. Eprint Arxiv, 2014.

[8]Day M Y, Lee C C. Deep learning for financial sentiment analysis on finance news providers[C]// Ieee/acm International Conference on Advances in Social Networks Analysis and Mining. IEEE, 2016:1127-1134.

[9]Zhang W, Skiena S. Trading Strategies to Exploit Blog and News Sentiment[C]// International Conference on Weblogs and Social Media, Icwsm 2010, Washington, Dc, Usa, May. DBLP, 2010.s

[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[11] Alec Go, Richa Bhayani, Lei Huang, 2009 [R]. Twitter Sentiment Classification Using Distant Supervision.

7 Appendix

7.1 Figure

Correlation between Senti-score and market indexes

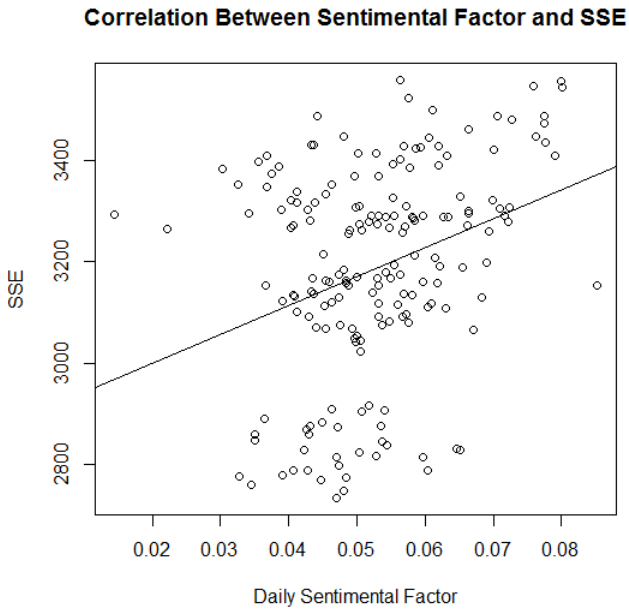

Figure 2: fig1

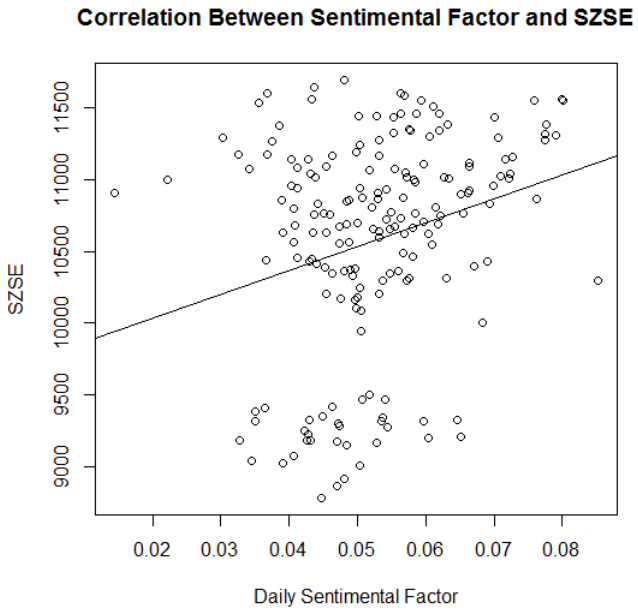

Figure 3: fig2

Time series of Senti-score and market indexes

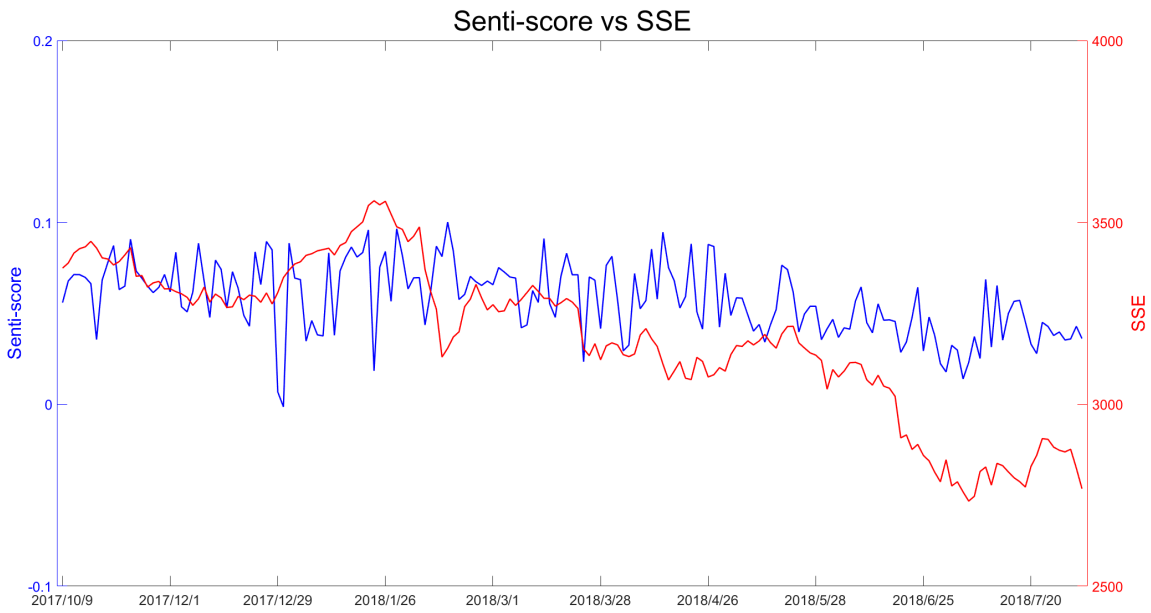

Figure 4: Time Series of Senti-score and SSE

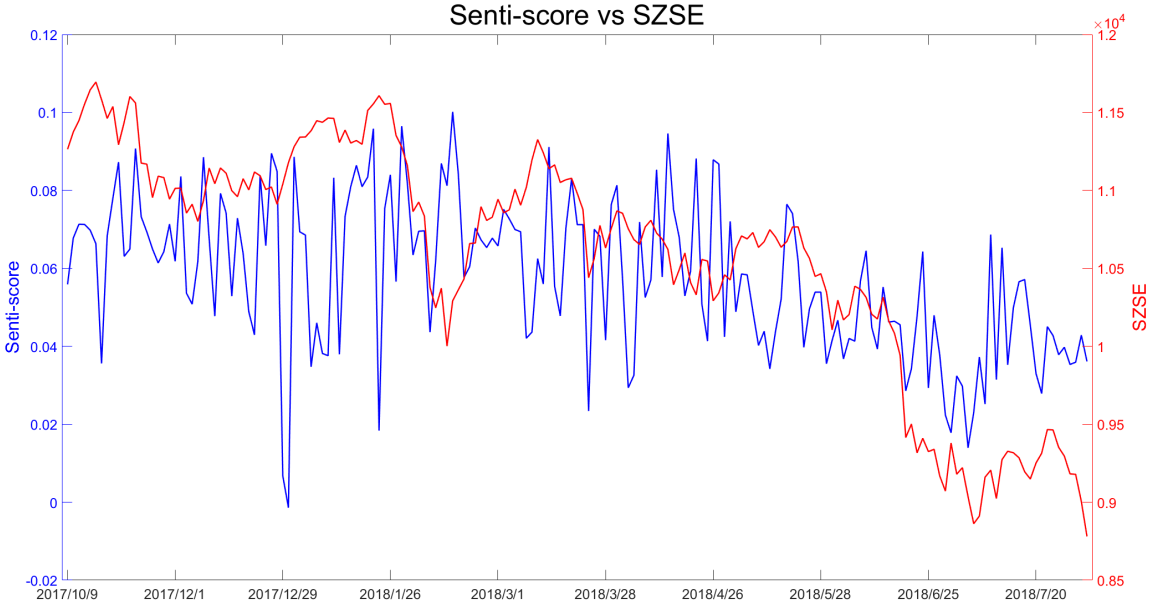

Figure 5: Time Series of Senti-score and SZSE