Postprint: Research on Weibo Sentiment Classification Based on Word Embeddings

Authors: Liu Kan, Yunying Yuan

Date: 2023-08-27T00:00:00+00:00

Abstract

[Purpose/Significance] Weibo has become an important platform for public emotional expression, and sentiment analysis of Weibo plays a significant role in public opinion analysis, user experience, and business opportunity mining. [Method/Process] The proposed sentiment orientation classification algorithm WE_{SDAE} uses word embedding to represent Weibo posts as low-dimensional dense vectors, then optimizes the basic autoencoder algorithm into a stacked denoising autoencoder through the addition of regularization terms and noise, and adds a classifier at the top layer to achieve sentiment orientation classification. Considering the flexible wording of Weibo, models are also trained from both character-level and word-level granularities. [Result/Conclusion] Experimental results demonstrate that the character-level based model outperforms the word-level based model. Furthermore, comparative experiments show that the WE_{SDAE} algorithm is superior to traditional algorithms such as SVM, Naive-Bayes, and XgBoost; the word embedding approach outperforms traditional Vector Space Model representation methods and can achieve promising results in Weibo sentiment analysis.

Full Text

Preamble

Volume 62, Issue 15, August 2018
ChinaXiv Cooperative Journal

Research on Microblog Sentiment Orientation Classification Based on Word Embedding

Liu Kan, Yuan Yunyin
School of Information and Safety Engineering, Zhongnan University of Economics and Law, Wuhan 430074

Abstract

[**Purpose/Significance**] Microblog has become an important platform for public emotional expression, and sentiment analysis of microblogs plays a significant role in public opinion analysis, user experience improvement, and business opportunity mining. [**Method/Process**] The proposed sentiment orientation classification algorithm WE_{SDAE} (Word Embedding_{Stacked} Denoising Auto-Encoder) represents microblogs as low-dimensional dense vectors through word embedding, then optimizes the basic auto-encoder into a deep denoising auto-encoder by adding regularization terms and noise processing, and finally adds a classifier at the top layer to achieve sentiment orientation classification. Considering the flexible language usage in microblogs, the model is trained at both character and word granularities. [**Result/Conclusion**] Experimental results demonstrate that the character-level model outperforms the word-level model. Furthermore, comparative experiments show that the WE_{SDAE} algorithm surpasses traditional algorithms such as SVM, Naive Bayes, and XgBoost, and that word embedding approaches are superior to traditional vector space model representations, achieving favorable results in microblog sentiment analysis.

Classification Number: TP391
Keywords: sentiment analysis, classification, auto-encoder, microblog

Microblogs contain rich emotional information from users. With the widespread adoption of microblogs, an increasing number of users have become accustomed to describing personal life experiences, expressing emotional feelings, or commenting on social events and hot topics. Microblogs often record the daily joys and sorrows of each user. Extracting and studying this emotional information can help governments conduct more timely and effective public opinion guidance, assist enterprises in improving product user experiences, and help entrepreneurs discover tremendous business opportunities. However, microblog texts are short and expressed non-standardly, making traditional methods unable to meet the processing requirements of microblogs. There is an urgent need for new methods that can efficiently extract sentiment orientation from microblogs.

1 Related Research

Sentiment analysis research primarily unfolds from two perspectives: coarse-grained and fine-grained. Coarse-grained analysis mainly focuses on document and sentence levels, concerning the overall positive or negative sentiment attitude toward an entire document or sentence. Fine-grained analysis, conversely, focuses on the word level, examining more specific attitudes beneath the overall sentiment, such as happiness, beauty, praise, and relaxation under positive sentiment. Due to the short length and limited word count of microblogs, it is difficult to conduct in-depth fine-grained research. Moreover, most microblogs express clear and singular attitudes, making them suitable for coarse-grained

sentiment analysis.

Currently, sentiment analysis in the microblog domain mainly includes two categories: methods based on sentiment knowledge and methods based on traditional machine learning. Sentiment knowledge can be specific emotion words, label information, sentiment lexicons, emoticons, etc. P.D. Turney et al. [1] selected "excellent" and "poor" as seed words, then extracted adjectives from sentences and calculated mutual information with these two seed words to determine sentence sentiment orientation. Ren Yuan et al. [2] proposed a more refined classification method for emotion words based on part-of-speech and attempted to introduce topic factors to design a topic-adaptive sentiment classification method. L. Barbosa et al. [3] considered Twitter's structural features and lexical information, utilizing polarity label information to complete sentiment orientation judgment. Pang Lei et al. [4] used emotion words and emoticon images in microblogs to obtain sentiment orientation. Pan Minghui et al. [5] combined emoticon dictionaries with traditional emotion dictionaries and formulated negation grammar rules to identify six emotions expressed in microblogs: joy, sorrow, anger, fear, disgust, and surprise, improving the accuracy of microblog emotion orientation analysis. Liu Quanchao et al. [6] constructed sentiment analysis dictionaries, internet slang dictionaries, and emoticon symbol libraries, and designed microblog sentiment orientation determination algorithms incorporating forwarding and commenting relationships between microblogs.

Machine learning models are also employed to solve sentiment orientation analysis problems. A. Bakliwal et al. [7] integrated semantic features and Twitter-related features, using SVM to classify tweets into positive, negative, and neutral categories. B. Johan et al. [8] utilized psychometric tools to extract six-dimensional emotion vectors for microblog sentiment orientation judgment. C. Tan et al. [9] assumed that users with social relationships are more likely to share similar views, thus proposing to incorporate social relationship information into traditional SVM to construct sentiment orientation models. Liu Zhiming et al. [10] used SVM, Naive Bayes, and N-gram methods for sentiment polarity classification. Zhu Xi et al. [11] optimized the feature selection method and iteration termination conditions of the semi-supervised learning method reserved-self-training, effectively preventing overfitting and improving model accuracy. Sun Jianwang et al. [12] extracted verbs and adjectives from microblogs, completed feature dimensionality reduction based on hierarchical structure, calculated feature polarity values through emoticons, and finally used SVM to classify microblog texts into positive, negative, and neutral categories.

Overall, for knowledge-based methods, constructing sentiment knowledge systems incurs high labor costs, requiring manual annotation for emotion words, labels, or sentiment dictionaries. Moreover, the application scope of knowledge-based methods is limited, as they can only process microblogs containing these sentiment knowledge items. Considering the widespread polysemy in Chinese, the same sentiment knowledge often expresses completely different sentiment orientations in different contexts. For traditional machine learning model-based

methods, existing approaches basically revolve around SVM, focusing on seeking expansion and improvement of features while contributing less to the classification model itself. The auto-encoder algorithm, with its powerful nonlinear learning capability, has achieved good results in many aspects of natural language processing and is a direction worth exploring. Furthermore, most features of machine learning models come from the vector space model. On the one hand, the vector space model cannot describe the semantic information shared between words [13], which affects the accuracy of sentiment orientation determination. On the other hand, microblogs are highly colloquial, with frequent abbreviations and arbitrary collocations. The high-dimensional sparsity characteristic of vectors transformed by the vector space model [14] has long troubled traditional machine learning methods. Word embedding can better capture the semantic features of each word within limited dimensions, which is helpful for solving the above problems.

2 Basic Approach

This paper proposes a microblog sentiment orientation classification algorithm based on word embedding and deep denoising auto-encoders, called WE_{SDAE} (Word Embedding_{Stacked} Denoising Auto-Encoder), without relying on any manually annotated sentiment knowledge. The algorithm mainly includes three core steps: using word embedding to obtain word vectors; improving the auto-encoder to extract low-dimensional abstract features; and supervised global parameter adjustment to complete sentiment orientation classification.

First, preprocessed microblog text is represented as a distributed low-dimensional dense vector through word embedding. Second, these vectors are input into the optimized deep denoising auto-encoder, which transforms them into abstract features through layer-by-layer unsupervised nonlinear learning. The optimization process of the auto-encoder includes two parts: adding L1 regularization terms to the objective function and adding noise during data preprocessing. Finally, a classifier is added at the top layer of the auto-encoder for supervised training with global parameter adjustment to obtain the final sentiment orientation classification algorithm. Considering the large number of abbreviations and shorthand expressions in microblogs and the flexible language usage, the algorithm processes data from two granularities: character and word. The flowchart of the sentiment orientation classification algorithm is shown in Figure 1 [Figure 1: see original paper].

3 Word Vector Acquisition

3.1 Word Embedding

There are many implementation methods for word embedding, with the most popular currently being CBOW (Continuous Bag-of-Words) [15] and Skip-gram

[16]. In semantic-related tasks, Skip-gram performs better than CBOW, so the Skip-gram method is adopted.

When given a set of training words w(t-2), w(t-1), w(t), w(t+1), w(t+2), Skip-gram takes the target word w(t) as input. After processing by the mapping layer, it outputs the words in the context of the target word. Based on the basic assumption that similar words have similar contexts, it attempts to predict contextual information through the current target word. The algorithm flow is shown in Figure 2 [Figure 2: see original paper].

Using the maximum likelihood principle, the objective function of this probabilistic language model is as shown in formula (1):

p(Context(w) | w) = _{u Context(w)} p(u | w) (1)

When solving this language model using the Hierarchical Softmax framework, the input layer is the word vector v_w of the center word w, and the output layer is the Huffman tree corresponding to Context(w), with leaf nodes being words in the context. For any word u in Context(w), there exists a path path_u from the root node to the leaf node corresponding to word u in the Huffman tree. path_u contains l_u nodes, where the encoding of the j-th node is d_{u,j} {0,1}. The Huffman encoding of word u can be represented as [d_{u,2}, d_{u,3}, …, d_{u,l_u-1}] (the root node has no corresponding encoding). The vector corresponding to the j-th non-leaf node in path_u is w_{u,j}. There are l_u - 1 non-leaf nodes on path_u, each leaf node can be seen as a branch corresponding to a binary classification that produces a probability. Multiplying these probabilities yields p(u | w), as shown in formula (2):

p(u | w) = *{j=2}^{l_u}* *p(d*{u,j} | v_w, _{u,j-1}) (2)

where v_w is the word vector of the center word w, which will be randomly initialized and then iteratively optimized through stochastic gradient descent to obtain the final result—a word vector containing the original word's semantic information.

3.2 Microblog Vector

Assume a microblog S consists of words w_1, w_2, w_3, …, w_n. Each word w_i ($1 \leq i \leq n$) can obtain a word vector v_w through the Skip-gram word embedding method. According to T. Mikolov's research results [16, 17], the basic arithmetic operations between word vectors obtained through Skip-gram contain rich potential semantic information. For example, adding the word vector of "Germany" to the word vector of "capital" yields a word vector very similar to that of "Berlin". Another example is that the word vector of "king" minus the word vector of "male" plus the word vector of "female" results in a word vector close to that of "queen". Based on this characteristic of Skip-gram, the average of all word vectors in microblog s is used as the microblog's word vector, as shown in formula (3). The concept is illustrated in Figure 4 [Figure 4: see original paper].

v_s = (1/n) *{i=1}^{n}* v{w_i} (3)

Processing all microblog data using the above method yields a microblog matrix S, as shown in formula (4):

S = [v_{s_1}, v_{s_2}, …, v_{s_M}]^T ^{M · d} (4)

where M represents the number of microblogs in the data, and d represents the dimension of the vector.

3.3 Word Granularity and Character Granularity

First, Chinese word segmentation tools are used to segment microblog text, using the segmented words as the basic composition units of each microblog. That is, the words w_1, w_2, w_3, …, w_n in microblog S refer to the words after segmentation. For example, "中国加油" (China cheer up) will be segmented into "中国 / 加油". This approach is regarded as a word-level word vector acquisition method.

Microblogs are typical short texts with limited context information, frequent abbreviations, and high noise. The segmentation results from word segmentation tools are often inaccurate and contain many ambiguities. For instance, the phrase "高大上网站" (high-end website) using traditional segmentation technology might result in "高大 / 上 / 网站" or "高大 / 上网 / 站", which cannot reflect the correct semantics the microblog intends to express. Additionally, microblogs continuously generate a large number of new words. These may be newly created by netizens, such as "城会玩" (city people know how to play) or "活久见" (live long enough to see anything), or original words that have acquired new meanings on microblogs. For example, in "我来安利一下这个 app" (Let me "anli" this app), "安利" no longer represents a brand but means "strongly recommend". This information may be lost during word segmentation, leading to unsatisfactory segmentation results. Therefore, drawing on X. Zheng et al.'s [18] solution for part-of-speech tagging problems, this paper proposes a character-level word vector acquisition method that directly splits all characters in the microblog, using individual characters as the basic composition units. That is, the words w_1, w_2, w_3, …, w_n in microblog S will represent individual Chinese characters. For example, "高大上网站" will be represented as "高 / 大 / 上 / 网 / 站".

4 Autoencoder Feature Extraction

The autoencoder is an important training model in deep learning that has achieved good results in natural language processing. Next, we will use autoencoders to learn text features, add regularization terms to constrain the algorithm's learning ability, perform noise addition on input data to improve robustness, and stack multiple autoencoders to enhance feature abstraction capability.

4.1 L1 Norm Regularization

Although the powerful nonlinear expression capability of autoencoders helps obtain abstract features, overfitting problems easily occur, where the algorithm also fully learns information unique to individuals. Different microblogs have large differences and inevitably contain many unique features. If the basic autoencoder algorithm is directly used, the extracted feature vectors may not reflect the essential commonalities of the data, and the trained model will have particularly poor generalization ability and cannot be effectively extended. Therefore, it is necessary to constrain the learning ability of the autoencoder.

L1 norm regularization is a commonly used variable selection method widely applied in various algorithm improvements. By using the absolute value function of autoencoder coefficients as a penalty term, coefficient values are compressed, and coefficients with smaller absolute values are directly compressed to 0, thereby ensuring the sparsity of algorithm parameters and avoiding excessive learning of non-significant features in microblogs. The specific calculations are shown in formulas (5) and (6):

L(x, z) = KL(x‖z) + Lasso( ) (5)
Lasso( ) = λ _j | _j| (6)

where the loss function is the Kullback-Leibler divergence, trained using the classic stochastic gradient descent algorithm. λ is the L1 norm parameter; the larger the value, the greater the penalty strength, and the sparser the training result. The specific value needs to be debugged based on actual data to balance the algorithm's fitting ability and generalization ability.

4.2 Noise Addition

Considering the high randomness of microblog input, a large number of netizens use abbreviations and shorthand, even personalized language and symbols when posting microblogs. Due to hurried input, phenomena such as extra input, missing input, and incorrect input of text often occur. This requires microblog sentiment orientation classification algorithms to have strong robustness. To address these issues, a certain amount of noise can be added to the word vectors of microblogs to increase the interference of training data. P. Vincent et al. [19] added noise by randomly selecting a certain proportion of data and forcing it to become 0. In addition to selecting some data to be forced to 0, this paper also selects a certain proportion of data to be forced to become a random number from the standard normal distribution. The former considers the situation of missing data in the input vector, and the trained autoencoder should have the ability to restore these missing values. The latter considers the widespread non-standardization in microblog input, ensuring the algorithm can avoid interference from these personalized or irrelevant inputs.

After the input vector x is input into the encoder, the encoding result y is obtained through linear transformation and activation function processing, as

calculated in formula (7). The encoding result y is then input into the decoder to obtain the reconstructed vector z, as shown in formula (8):

y = f_ (x) = s(Wx + b) (7)
z = g_{ '}(y) = s(W'y + b') (8)

The encoding parameters are = {W, b}, and the decoding parameters are ' = {W', b'}. Here, W is a d' × d weight matrix, W' is the transpose of W (i.e., W' = W^T), and b and b' are corresponding bias vectors. The optimization objective is to make the reconstructed vector z as close as possible to the input vector x, i.e., to minimize the loss from reconstruction and obtain the optimal parameters * and '*, as shown in formula (9):

, ' = argmin L(x, z) = argmin L(x, g_{ '}(f_ (x))) (9)

After adding noise, the input vector x becomes x̃. Using the stochastic gradient descent algorithm, the optimal encoding and decoding parameters * and '* are obtained, as shown in formula (10):

, ' = argmin L(x, z) = argmin L(x, g_{ '}(f_ (x̃))) (10)

4.3 Deep Autoencoder

Stacking multiple denoising autoencoders forms a deep learning network. The more stacked autoencoder layers, the stronger the ability to learn abstract features [20]. During training, the abstract feature vector output by the K-1 layer autoencoder, after adding noise, serves as the input vector for the K-layer autoencoder. The K-layer autoencoder minimizes the loss function to make the reconstructed vector processed by the decoder as close as possible to the original input vector without noise. Parameters are continuously optimized and adjusted. After obtaining the optimal solution, the K-layer autoencoder discards the decoder part and uses the abstract feature vector processed by the encoder (with added noise) as the input vector for layer K+1 to continue the next layer of training. This cycle repeats, layer by layer, forming a deep denoising autoencoder model that ultimately obtains feature vectors. Its structure is shown in Figure 5 [Figure 5: see original paper].

5 Sentiment Orientation Classification

The deep denoising autoencoder described above only extracts abstract features from microblog text through unsupervised methods and cannot yet complete sentiment orientation classification. It is necessary to add a classifier layer behind the last layer of the deep denoising autoencoder as the final output layer.

The input vector of this layer is the output result of the last autoencoder layer, and the output vector is the microblog sentiment orientation vector. Here, the sentiment label set is defined as T = {positive, neutral, negative}, so the dimension of the microblog sentiment orientation vector is 3, in the form v_e

= [a, b, c]^T, where a represents the positive sentiment label, b represents the neutral sentiment label, and c represents the negative sentiment label. A value of 1 indicates the presence of sentiment, and 0 indicates the absence of sentiment. Assuming a microblog expresses only one main sentiment, for example, v_i = [1, 0, 0]^T indicates that microblog i has a positive sentiment orientation.

The processing of this layer includes two steps: linear transformation of the input vector and nonlinear adjustment of the output final vector through an activation function. The softmax function, widely used in multi-classification problems, is selected as the activation function. The calculation is shown in formula (11):

f(x) = softmax(Wx + b) (11)

where W ^{h×d} (h is the number of sentiment categories, here 3, and d is the dimension of the input vector) represents the weight matrix, and b ^h represents the bias term.

This layer still uses KL divergence as the loss function, measuring the similarity between the distribution of the algorithm's output vector and the target vector. The calculation formula is shown in (12):

L(X, f(X)) = KL(x‖f(X)) (12)

Thus, the training process of the entire microblog sentiment orientation classification algorithm WE_{SDAE} can be summarized in two steps. First, the single-layer denoising autoencoders in the algorithm will sequentially perform unsupervised learning, continuously abstracting and iterating to gradually extract the essential features of the data from the original input vectors as the basis for subsequent training. In this step, the training process of each layer is relatively independent. Then, with the help of already annotated sentiment orientation vectors, global supervised learning is conducted. The training method still uses standard gradient descent. This optimization process not only adjusts the parameters in the top-level classifier but also fine-tunes all previously trained denoising autoencoder parameters, ensuring the entire process has optimal learning capability, as shown in Figure 6 [Figure 6: see original paper].

6 Experimental Results and Analysis

6.1 Data Source Selection and Data Preprocessing

The experimental data comes from the COAE (Chinese Opinion Analysis Evaluation) microblog evaluation dataset from the 2014 NLPCC conference. This dataset contains 5,000 annotated microblog data items, including 2,656 with positive sentiment orientation and 2,344 with negative sentiment orientation. Additionally, 2,500 microblogs with neutral sentiment orientation were crawled and manually annotated, resulting in a final total of 7,500 experimental data items, as detailed in Table 1 . Sample data is shown in Figure 7 [Figure 7: see

original paper]. The data was randomly split into two parts at a 4:1 ratio for training and test sets.

Table 1 Experimental Data
| Positive | Neutral | Negative | Total | |———-|———|———-|——-| | 2656 | 2500 | 2344 | 7500 |

The microblog dataset in COAE2014 has undergone certain cleaning and organization, removing emoticons and system-generated "forwarded microblog" information, and saved in text format. Based on this, the following four preprocessing steps were performed:

1. **Splitting multiply forwarded microblogs.** For example, a microblog in the dataset "看看相片，看看孩子可怜的身体，愤怒!! ／／@张蜀梅：是真的吗? ／／@阿子：他是从罗马尼亚流窜过来的么?? " actually contains multiple microblogs from multiple users and needs to be split and restored.

2. **Removing irrelevant text.** For instance, in "回复@ｆｙｘ璇：哈哈，我可是鲁能泰山的忠实球迷，今年中超南京客场我可找你啊," the part "回复@ｆｙｘ璇：" is not the content posted by the user and is irrelevant to sentiment orientation analysis.

3. **Removing link addresses but retaining the keyword "http".** Links in microblogs are short URLs automatically generated by the system and do not contain much information themselves. However, considering that the act of adding links may indicate the user's firm stance, information indicating this behavior needs to be retained. For example, "质量很垃圾!! 我在：http://t.cn/zjOsElS".

4. **Removing @ names but retaining the keyword "@".** The @ behavior generally occurs when the author has positive or negative sentiment orientation, but the name content is irrelevant to the sentiment, so it can be directly removed. For example, "雷克萨斯不错，不过日本车，哎！还是喜欢奥迪！@Ｎｉｃｋｏｌｅ星@夏日料理王@ｓｓ６２８@ Gulu Gulu Gulu Blue @潘炜晨".

Additionally, during preprocessing, punctuation marks and topic tag information were deliberately retained. This is because punctuation marks are important carriers of user emotion, and different punctuation marks often carry different sentiment information. For example, "?" indicates questioning and is often used in negative sentiment orientation [21]. Topic tags are concise summaries of microblog discussion content and often carry specific emotional information, helping to better determine sentiment orientation, such as "# 信用卡里传来了绝望的呜咽 #" (The desperate whimper from the credit card).

6.2 Evaluation Metrics

The average F-value F_{avg} is used to screen parameters, while the actual effect is analyzed by examining the F_i value for each sentiment orientation

category C_i [22-23]. The F_{avg} value is the average of the F_i values for all sentiment orientation categories, calculated as shown in formula (13):

F_{avg} = (1/C) _{i=1}^{C} F_i (13)

where C is the number of different sentiment orientation categories, which takes the value 3 in this paper.

The F_i value comprehensively considers the precision and recall of sentiment orientation category C_i. A larger value indicates better performance of sentiment orientation classification on this category. The calculation formulas are shown in (14) and (15):

F_i = (2 ·precision_i ·recall_i) / (precision_i + recall_i) (14)
precision_i = m_{right}i / (m_{right}i + m_{wrong}i), recall_i = m_{right}i / m_{all}i (15)

where m_{right}i is the number of microblogs correctly classified into sentiment orientation category C_i, m_{wrong}i is the number of microblogs incorrectly classified into sentiment orientation category C_i, and m_{all}i is the actual number of microblogs contained in sentiment orientation category C_i.

6.3 Algorithm Implementation

The sentiment orientation classification algorithm WE_{SDAE} proposed in this paper can be implemented in two steps in practice: obtaining microblog vectors and sentiment classification, while considering both character and word granularities.

Obtaining Microblog Vectors. Preprocessed microblogs are first segmented, then each character or word is converted into a vector through word embedding, and finally the final vector is calculated through formulas. When processing microblog data at the character granularity, the processing is relatively simple: the cleaned microblogs are directly split character by character. For example, "喜欢这款产品！" (Like this product!) becomes "喜 / 欢 / 这 / 款 / 产 / 品 / ！" after segmentation. When the granularity requirement is word-level, the NLPIR Chinese word segmentation system, most commonly used in Chinese natural language processing, is used to segment the microblog data. The segmentation result of "喜欢这款产品！" will become "喜欢 / 这 / 款 / 产品 / ！". During processing, it should be noted that each punctuation mark is also retained as a separate character or word in the segmented data.

The Skip-gram method for training word vectors has been efficiently implemented in the open-source tool word2vec, which is directly used in experiments to obtain word vectors for microblog data. According to V. Mikolov et al. [24], the larger the amount of data used in training, the better the trained word vectors can describe the semantic features of words. Therefore, the full 40,000 microblog data items from COAE2014 will be used to train each word vector. In the character-level word vector training process, the basic composition unit of

microblogs is a single character, and word2vec will train a corresponding word vector for each character. In the word-level word vector training process, the basic composition unit becomes the word after segmentation, and word2vec will train a corresponding word vector for each segmented word. To ensure stronger comparability between the character-level model and the word-level model, the same set of parameters is used when training word vectors, as detailed in Table 2 .

Table 2 Word2vec Parameter List
| Parameter | Value | |————|——-| | Word vector dimension (size) | 300 | | Context window (window) | 8 | | Minimum frequency (min_{count}) | 5 | | Sampling threshold (sample) | 1e-3 | | Number of iterations (iter) | 5 |

Based on research experience in Chinese word vectors, the word vector length is set to 300, meaning each word is mapped into a continuous space with a dimension of 300. The context window is set to 8, meaning contextual information consists of the 8 words before and after the target word. The minimum frequency of 5 ensures that only words appearing more than 5 times in the dataset are added to the word vector library; those not added are randomly initialized. The sampling threshold of 1e-3 determines whether a word will be sampled during training. If a word's frequency exceeds this threshold, it will be sampled to save training time. The number of iterations for word vector training is set to 5.

After the above steps, both character-level and word-level data have obtained corresponding word vector libraries. Next, microblog vectors are processed sequentially. For processed character-level data, each character in a microblog corresponds to a word vector. These vectors are aggregated, and the microblog's vector is calculated according to formula (3). For processed word-level data, the aggregated vectors come from the word-level word vector library, and the final result is similarly obtained using formula (3).

Implementing the Classification Algorithm. The WE_{SDAE} algorithm is implemented using the deep learning library Theano. Key parameters in the training process include the number of hidden layers, number of nodes per layer, regularization penalty coefficient, and noise addition ratio. Since the input vectors include both character-level and word-level types, parameter selection is conducted separately for training. Candidate parameter values are shown in Table 3 .

Table 3 Candidate Parameter Values
| Parameter | Candidate Values | |————|————————| | Number of hidden layers | 2/3/4/5/6 | | Number of nodes per layer | 100/300/500/700 | | Regularization penalty coefficient | 1e-1/1e-2/1e-3/1e-4/1e-5 | | Noise addition ratio | 0.1/0.2/0.3/0.4/0.5 |

As shown in Figure 8 [Figure 8: see original paper], the overall performance on the training set improves as the number of hidden layers increases, indicating that more hidden layers enable the extraction of more information-rich

abstract features from the training set, but may also lead to overfitting. From the perspective of test set performance, the effect shows a single-peak pattern. The character-level model reaches optimal performance at 4 layers, while the word-level model reaches optimal performance at 3 layers.

As shown in Figure 9 [Figure 9: see original paper], as the number of nodes per layer increases, performance on the training set gradually improves, then stabilizes, and finally declines slightly. This aligns with Y. Bengio's previous research results [25], where autoencoders can obtain better feature extraction effects when the output layer dimension is larger. The algorithm's performance on the test set still shows a single-peak pattern, and both character-level and word-level models achieve the best performance when the number of nodes per layer is 500.

As shown in Figure 10 [Figure 10: see original paper], the larger the regularization penalty coefficient, the more parameters are set to 0, and the stronger the constraint on learning ability. In the experiment, as the regularization penalty coefficient continuously decreases, the performance on both training and test sets first improves and then deteriorates overall. The difference between training and test set performance also reaches its minimum at the same value. Both character-level and word-level models achieve the best regularization penalty coefficient at 1e-3.

As shown in Figure 11 [Figure 11: see original paper], as the noise addition ratio increases, the performance difference between training and test sets becomes smaller, and performance on the test set improves, indicating that for short texts like microblogs with flexible language usage, adding noise indeed enhances the algorithm's anti-interference capability. Overall, the word-level model fluctuates more significantly than the character-level model, indicating that the word segmentation process introduces some deviations and contains more noise information. The optimal noise addition ratio for the word-level model is 0.5, while for the character-level model it is 0.4. Noise is added in two ways: some values are set to 0, and some are set to random numbers. In this experiment, the ratio is set to 4:1.

Based on the above analysis, the character-level model achieves optimal performance with 4 hidden layers, 500 nodes per layer, a regularization penalty coefficient of 1e-3, and a noise addition ratio of 40%. The word-level model achieves optimal performance with 3 hidden layers, 500 nodes per layer, a regularization penalty coefficient of 1e-3, and a noise addition ratio of 50%. At this point, the algorithm's performance on each sentiment orientation category is shown in Table 4 .

Table 4 F-value Comparison of Character-level and Word-level Models

Model	Positive	Neutral	Negative	Average
Character-level	0.871	0.862	0.865	0.866
Word-level	0.858	0.858	0.866	0.861

This shows that the character-level model has stronger recognition ability than

the word-level model across all sentiment orientation categories, indicating that for microblog corpora, traditional word segmentation tools are indeed not accurate enough in segmentation work, resulting in some information loss. The character-level model can better preserve information in microblogs and obtain more comprehensive and effective features.

Figure 12 [Figure 12: see original paper] shows detailed experimental results for the character-level model. From the figure, the precision for positive sentiment is the highest, indicating that positive sentiment is easier to identify, but the recall is lower, showing that some positive sentiment is missed. Neutral sentiment shows the opposite pattern, indicating that many positive sentiments are classified as neutral. Negative sentiment falls in between, indicating its features are not prominent enough. This is also a characteristic of this algorithm—it does not rely on annotated sentiment systems or sentiment lexicons, yet all indicators achieve relatively ideal results.

6.4 Comparative Analysis

Two groups of comparative experiments were designed to verify the rationality and effectiveness of the WE_{SDAE} algorithm. Based on the previous experimental analysis, the comprehensive performance of the character-level model is better than that of the word-level model. Therefore, the WE_{SDAE} algorithm used in the following comparative experiments is implemented based on the character-level model.

(1) Comparative Analysis of WE_{SDAE} and WE_{SVM}. Currently, the most commonly used algorithm for sentiment orientation classification is SVM. Therefore, the WE_{SVM} algorithm is trained on the same dataset and its results are compared with WE_{SDAE}.

The LibSVM toolkit is used for experiments. The data preprocessing method is exactly the same as the character-level WE_{SDAE} algorithm. Each microblog is ultimately represented as a vector in continuous space. After debugging and optimization, the performance of WE_{SVM} on the test dataset is compared with WE_{SDAE}, as shown in Table 5 .

Table 5 F-value Comparison of WE_{SDAE} and WE_{SVM} Algorithms
| Algorithm | Positive | Neutral | Negative | Average | |————|————-|————|————-|————| | WE_{SDAE} | 0.871 | 0.862 | 0.865 | 0.866 | | WE_{SVM} | 0.829 | 0.812 | 0.831 | 0.824 |

This shows that the WE_{SDAE} algorithm is indeed superior to the traditional WE_{SVM}. The deep denoising autoencoder can better extract abstract features from microblogs, helping the classifier obtain more accurate sentiment orientation determination results.

(2) Comparative Analysis of WE_{SDAE} and VSM_{SDAE}. The WE_{SDAE} algorithm uses word embedding to obtain each microblog vector. Currently, in Chinese natural language processing, the more common approach

is to use the Vector Space Model (VSM) to process microblog data. Therefore, the experimental data is processed into VSM form for comparison with the WE approach.

Characters appearing fewer than 5 times are removed, resulting in 5,236 different characters, making the corresponding microblog vector length also 5,236. To ensure that data processed by the VSM method and the WE method can be used in the same classifier, dimensionality reduction is required for microblog vectors processed by the VSM method. The PCA algorithm in Scikit-Learn is used to reduce the vectors to 300 dimensions before inputting them into the classifier for training. The results are shown in Table 6 , which also includes comparison results with SVM, Naive Bayes (N-Bayes), and ensemble classification (XgBoost) algorithms.

Table 6 F-value Comparison of WE_{SDAE} and VSM_{SDAE} Algorithms | Algorithm | Positive | Neutral | Negative | Average | |————|———-|———|———-|———| | WE_{SDAE} | 0.871 | 0.862 | 0.865 | 0.866 | | VSM_{SDAE} | 0.805 | 0.787 | 0.805 | 0.799 | | XgBoost | 0.711 | 0.695 | 0.731 | 0.712 | | N-Bayes | 0.756 | 0.738 | 0.762 | 0.752 | | SVM | 0.767 | 0.759 | 0.743 | 0.756 |

This shows that the word embedding vector acquisition method is indeed superior to the traditional vector space model. On the one hand, word embedding solves the dimensionality disaster problem to a certain extent, avoiding dimensionality reduction operations. On the other hand, it can better mine the semantic and contextual information of text, helping to solve sentiment orientation classification problems.

This paper proposes new ideas for microblog sentiment orientation classification from three aspects: 1) Considering that word vectors obtained through traditional vector space models ignore the semantic correlation between words and lack important contextual information in semantic analysis, word embedding is used to map each word in microblogs into a vector in continuous space, maximizing the retention of semantic information in microblog text itself; 2) The algorithm no longer relies on any manually annotated sentiment knowledge system or traditional machine learning models, but improves the autoencoder in deep learning, leveraging its powerful unsupervised nonlinear learning capability to complete microblog feature extraction and sentiment orientation prediction; 3) As a casual social media platform, microblog users often use flexible language with numerous abbreviations. Traditional word segmentation tools may not accurately identify this information, so experiments segment microblog text at both character and word granularities to minimize unnecessary information loss.

The algorithm proposed in this paper is affected by the corpus. A larger or more specialized corpus will make character/word encoding more accurate, which is beneficial for improving algorithm precision. At the same time, due to the lack of assistance from sentiment lexicons, the algorithm's grasp of sentiment words is insufficient. Future work could consider adding enhanced processing of

sentiment words. Additionally, further research includes incorporating syntactic analysis and contextual semantic relationships.

References

[1] TURNEY P D. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews[C]//Proceedings of annual meeting of the Association for Computational Linguistics. Stroudsburg PA: The Association for Computer Linguistics, 2002: 417-424.

[2] Ren Yuan, Chao Wenhan, Zhou Qing, et al. Topic-adaptive Chinese microblog sentiment analysis[J]. Computer Science, 2013, 40(11): 231-235.

[3] BARBOSA L, FENG J. Robust sentiment detection on Twitter from biased and noisy data[C]//Proceedings of 23rd international conference on computational linguistics. Cambridge: MIT Press, 2010: 36-44.

[4] Pang Lei, Li Shoushan, Zhou Guodong. Chinese microblog sentiment classification method based on emotion knowledge[J]. Computer Engineering, 2012, 38(13): 156-158.

[5] Pan Minghui, Niu Yun. Microblog emotion recognition based on multi-cue mixed dictionary[J]. Computer Technology and Development, 2014(9): 28-32.

[6] Liu Quanchao, Huang Heyan, Feng Chong. Research on multi-feature microblog topic sentiment orientation determination algorithm[J]. Journal of Chinese Information Processing, 2014, 28(4): 123-131.

[7] BAKLIWAL A, FOSTER J, VANDERPUIL J, et al. Sentiment analysis of political Tweets: toward an accurate classifier[C]//Proceedings of NAACL Workshop on language analysis in social media. Stroudsburg PA: The Association for Computer Linguistics, 2013: 49-58.

[8] JOHAN B, ALBERTO P, HUINA M. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena[C]//Proceedings of 5th AAAI international conference on Weblogs and social media. Menlo Park, California: The AAAI Press, 2011: 450-453.

[9] TAN C, LEE L, TANG J, et al. User-level sentiment analysis incorporating social networks[C]//Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining. New York: ACM, 2011: 1397-1405.

[10] Liu Zhiming, Liu Lu. Empirical research on Chinese microblog sentiment classification based on machine learning[J]. Computer Engineering and Applications, 2012, 48(1): 1-4.

[11] Zhu Xi, Dong Xishuang, Guan Yi, et al. Microblog sentiment orientation analysis based on semi-supervised learning[J]. Journal of Shandong University: Natural Science Edition, 2014, 49(11): 37-42.

[12] Sun Jianwang, Lü Xueqiang, Zhang Leihan. Research on Chinese microblog sentiment analysis based on dictionary and machine learning[J]. Computer Applications and Software, 2014, 31(7): 177-181.

[13] LIU N, ZHANG B, YAN J, et al. Text representation: from vector to tensor[C]//Proceedings of IEEE international conference on data mining. New Jersey: IEEE Press, 2005: 725-728.

[14] KIM K, LEE J. Sentiment visualization and classification via semi-supervised nonlinear dimensionality reduction[J]. Pattern recognition, 2014, 47(2): 758-768.

[15] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[C]//Proceedings of international conference on learning representations. New York: ACM, 2013: 1301-1309.

[16] MILOLOV T, SUTSKEVER I, CHEN K, et al. Distributed representations of words and phrases and their compositionality[C]//Proceedings of the 27th annual conference on neural information processing systems. Cambridge: MIT Press, 2013: 3111-3119.

[17] MILOLOV T, YIH W, ZWEIG G. Linguistic regularities in continuous space word representations[C]//Proceedings of the 2013 conference of the North American Chapter of the Association for Computational Linguistics: human language technologies. Stroudsburg: The Association for Computer Linguistics, 2013: 746-751.

[18] ZHENG X, CHEN H, XU T. Deep learning for Chinese word segmentation and POS tagging[C]//Proceedings of the 2013 conference on Empirical methods in natural language processing. Stroudsburg: The Association for Computer Linguistics, 2013: 647-657.

[19] VINCENT P, LAROCHELLE H, BENGIO Y, et al. Extracting and composing robust features with denoising autoencoders[C]//Proceedings of the 25th international conference on machine learning. New York: ACM, 2008: 1096-1103.

[20] VINCENT P, LAROCHELLE H, LAJOIE I, et al. Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion[J]. Journal of machine learning research, 2010, 11(6): 3371-3408.

[21] Gui Bin, Yang Xiaoping, Zhu Jianlin, et al. Research on Chinese microblog sentiment orientation analysis based on sense group division[J]. Journal of Chinese Information Processing, 2015, 29(3): 100-105.

[22] HASSAN S, HE Y, HARITH A. Semantic sentiment analysis of Twitter[C]//Proceedings of the 11th international conference on the semantic Web. Berlin: Springer, 2012: 508-524.

[23] PAK A, PAROUBEK P. Twitter as a corpus for sentiment analysis and opinion mining[C]//Seventh conference on international language resources &

evaluation. Paris: European Language Resources Association, 2010: 1320-1326.

[24] SVETRIK V, LIAW A, TONG C, et al. Random forest: a classification and regression tool for compound classification and QSAR modeling[J]. Journal of chemical information & computer sciences, 2003, 43(6): 1947-1958.

[25] BENGIO Y, LAMBLIN P, POPOVICI D, et al. Greedy layer-wise training of deep networks[C]//Proceedings of the 21st annual conference on neural information processing systems. Cambridge: MIT Press, 2007: 153-160.

Author Contributions:
Liu Kan: Model design, architecture design, paper revision and finalization.
Yuan Yunyin: Data collection, experiments, and initial draft.

Abstract: [Purpose/significance] Microblog has become an important platform for public emotional expression. Microblog's sentiment analysis plays an important role in public opinion analysis, user experience, and business opportunity mining. [Method/process] The sentiment orientation classification algorithm WE_{SDAE} proposed in this paper uses word embedding to transform a microblog into a low-dimensional dense vector, then optimizes the basic auto-encoder algorithm into a deep denoising auto-encoder by adding regularization terms and noise processing, and adds a classifier at the top level to achieve sentiment orientation classification. Considering the flexible language usage in microblogs, the model is trained at both character and word granularities. [Result/conclusion] Experimental results show that the character-level model outperforms the word-level model. In addition, comparative experiments demonstrate that the WE_{SDAE} algorithm is superior to traditional SVM, Naive Bayes, XgBoost, and other related algorithms; word embedding approaches are better than traditional vector space model representations and can achieve good results in microblog sentiment analysis.

Keywords: sentiment analysis, classification, auto-encoder, microblog

Note: Figure translations are in progress. See original paper for figures.