Deep Learning-Based Sentiment Classification of Product Reviews (Postprint)

Authors: Li Wenjiang, Chen Shiqin

Date: 2023-10-08T00:00:00+00:00

Abstract

[Purpose / Significance] By combining existing text representation and classification algorithms, this study identifies a combination scheme with low computational complexity and reduced training time to construct an optimized model for sentiment classification of product review texts. [Method / Process] Employing the Keras API as the implementation environment, Word2vec word embeddings are fed into the Embedding layer. Based on sentence word index sequences, three text representations for product reviews are realized by controlling the trainable parameter. Different text representations are then paired with distinct classification algorithms, analyzing disparities in classification performance to establish an optimal algorithmic combination. [Results / Conclusion] The text representation approach wherein Word2vec embeddings are input into the Embedding layer for continued training, combined with a classification model trained via the TextCNN algorithm, demonstrates favorable classification performance on the product review test set, achieving accuracy and ROC-AUC values of 94.02% and 0.9827, respectively. Empirical results indicate that the classification model can effectively accomplish sentiment classification of product reviews and exhibits favorable generalization capability.

Full Text

Preamble

ChinaXiv Partner Journal [Academic Exploration]
Research on Sentiment Classification of Commodity Reviews Based on Deep Learning
Li Wenjiang[1] Chen Shiqin[2]

Abstract

[**Purpose/Significance**] This study combines existing text representation and classification algorithms to identify a low-complexity, minimal-training-time configuration for constructing an optimized sentiment classification model for commodity review texts. [**Method/Process**] Using the Keras API as the application environment, Word2vec word vectors are input into an Embedding layer. Based on sentence word index sequences, three distinct text representation methods for commodity reviews are implemented by controlling the trainable parameter. Different text representations are then matched with various classification algorithms, and classification performance differences are analyzed to establish an optimal algorithm combination. [**Result/Conclusion**] The text representation method that continues training Word2vec word vectors through an Embedding layer, combined with the TextCNN algorithm for classification model training, demonstrates superior performance on the commodity review test set, achieving an accuracy of 94.02% and an ROC curve AUC value of 0.9827. Application results show that this classification model effectively realizes sentiment classification of commodity reviews and exhibits strong generalization capability.

Keywords: deep learning; sentiment classification; Word2vec word vector; Embedding layer; TextCNN
Classification Number: TP391
Citation Format: Li Wenjiang, Chen Shiqin. Research on Sentiment Classification of Commodity Reviews Based on Deep Learning [J/OL]. Knowledge Management Forum, 2018, 3(6): 353-363 [citation date]. http://www.kmf.ac.cn/p/154/.

Text sentiment analysis represents a critical challenge in intelligent multimedia content understanding and forms the foundation for enabling machines to learn, reason, and cognize. Text sentiment classification constitutes the core of sentiment analysis. Current research faces difficulties in obtaining standard texts, uneven distribution of emotional resources, and data sparsity issues. Moreover, diverse text classification technologies exhibit different advantages and disadvantages, necessitating that specific sentiment classification techniques be applied to concrete domains to discover their optimal effectiveness [1]. E-commerce websites provide abundant commodity review data with distinct emotional features, making sentiment analysis of commodity review texts both practically grounded and promising for application [2].

As improvements to text representation methods and classification algorithms for commodity review sentiment classification increase in complexity, classification accuracy improves somewhat, but model training time lengthens and algorithm reproducibility becomes more difficult. Building upon current mainstream text representation and classification algorithms, this paper adopts JD.com product reviews as corpus and selects different combinations of text representation methods and classification algorithms to conduct applied research on

text sentiment classification model construction, training, and evaluation from the perspectives of reducing model complexity, decreasing training time, and lowering implementation difficulty.

1 Related Research on Commodity Review Sentiment Classification

Existing research on commodity review sentiment classification reveals three primary categories of sentiment analysis techniques: (1) rule-based sentiment classification, which establishes semantic rules combined with corpora and sentiment dictionaries to extract emotional information and calculate sentiment orientation. For instance, Wang Zhitao et al. performed sentiment analysis on Weibo by developing syntactic analysis rules, inter-sentence relationship rules, and word n-gram analysis rules [3]. The key to this approach lies in rule formulation, which depends on sentiment dictionaries and classification objectives, requiring substantial time and manual effort. Consequently, this method is now rarely employed. (2) machine learning-based sentiment classification, which utilizes machine learning methods to establish sentiment classification models from known text features for classifying unknown texts. Key aspects include feature and algorithm selection, particularly feature selection. Commonly applied machine learning algorithms for commodity review sentiment analysis include Naive Bayes (NB) [4-5], Stochastic Gradient Descent (SGD) [5-6], Support Vector Machine (SVM) [7], K-Nearest Neighbors (KNN) [8], and Random Forest (RF). While feature extraction methods vary, all rely on sentiment dictionaries.

(3) deep learning-based sentiment classification. With deep learning development, numerous methods have been applied to sentiment classification. Examples include Hu Chaoju et al.'s fusion of sentiment tags to improve Word2vec vectors for text representation [9], Jin Zhigang et al.'s use of Bi-LSTM to extract abstract features from social media user evaluations [10], Liu Quan et al.'s RCNN-HLSTM deep hierarchical network model for sentiment analysis [11], and Yoon K's TextCNN model for text classification [12]. Shallow machine learning methods have limitations in feature selection, whereas deep learning automatically learns and extracts features from data without requiring manual sentiment dictionary construction, making it a current research hotspot.

Deep learning primarily employs word embedding techniques (e.g., Word2vec [13], GloVe, FastText, WordRank, text2vec) for word vector representation of text information, enabling calculation of semantic relationships between words and facilitating abstract text feature extraction. Numerous excellent deep neural network classification algorithms are available (e.g., MLP, CNN, TextCNN [12], RNN, LSTM, GRU, Bi_{GRU}, Bi_{LSTM}, and various improved neural network algorithms). Research focuses on improving and fusing these classic text representation methods and classification algorithms, such as Hu Chaoju et al.'s fusion of sentiment tags to improve Word2vec as text representation, combining attention-based Bi_{LSTM} with CNN in parallel to accept word

vector input, merging outputs with attention again, connecting to LSTM, and finally outputting sentence vectors [14]. However, such fused algorithms are relatively complex, and similar algorithmic fusion improvements are numerous [10-12, 15-18].

Both rule-based and machine learning-based sentiment classification rely on sentiment dictionaries to represent emotional features, making dictionary quality directly affect feature extraction quality. Since sentiment dictionary construction requires human prior knowledge, the process is susceptible to human factors, and dictionaries lack generalizability—features from one domain may not adapt to another. Consequently, these two methods have not become mainstream. Instead, improvements and fusions of text representation and classification algorithms in deep learning-based sentiment classification represent the current research focus. However, as text representation methods and classification algorithm improvements increase in complexity, training time extends and algorithm reproducibility becomes more challenging.

2.1 Research Approach

Addressing the aforementioned issues in deep learning-based sentiment classification, this paper selects deep learning methods to conduct applied research on commodity review text sentiment classification model construction, training, and evaluation from perspectives of reducing model complexity, decreasing training time, and lowering implementation difficulty.

To reduce model complexity, we adopt deep learning framework Embedding layer training methods and classic classification algorithms. The former requires only parameter configuration in the Embedding layer, while the latter are mostly integrated into deep learning frameworks as directly callable modules with parameter settings. This entire process requires no algorithmic improvements, thereby reducing implementation difficulty.

We combine Word2vec embedding technology for pre-training word vectors with deep learning framework Embedding layer training to form three corpus text representation methods: Word2vec-trained vectors, Embedding-trained vectors, and their combination. These are respectively matched with multiple classification algorithms. By comparing classification performance differences, we establish optimal combinations of text representation methods and classification algorithms.

First, the corpus is segmented and Word2vec vectors are pre-trained to obtain semantically informative word vectors. Based on each corpus's segmentation order, corresponding word vectors are retrieved and arranged sequentially to form pre-trained sentence vectors. These then undergo continued training through the Embedding layer, after which sentence vectors and positive/negative sentiment labels are input through the input layer into classification algorithms for iterative training through hidden and output layers, ultimately obtaining the

sentiment classification model. The main implementation framework is shown in Figure 1 [Figure 1: see original paper].

2.2 Implementation Framework

Using the Keras [19] deep learning API as the application environment, sentiment classification involves three primary tasks: text encoding (including text segmentation, dictionary establishment, sentence word index conversion, word vector training, etc.), dataset construction, model building and training, and model evaluation. Based on this framework, a detailed implementation process is designed as shown in Figure 2 [Figure 2: see original paper].

The entire process corresponds to four specific steps: (1) Corpus segmentation processing. A Chinese word segmentation tool is selected to segment the raw corpus without stop-word filtering, using regular expressions to filter non-Chinese characters. (2) Sentence word index matrix construction. All corpus samples are traversed for word frequency statistics, with each word assigned a unique ID number in frequency order to generate a word dictionary. All sample sentences are then converted into word index ID sequences, forming the sentence word index matrix. (3) Word vector matrix construction. A word vector training tool is selected with appropriate parameters to train corpus word vectors, which are then combined with word index IDs to construct the word vector matrix, where each column represents a word's vector. (4) Classification model training. A suitable classification algorithm constructs the classification model, loading the word vector matrix, sentence word index matrix, and corresponding sentiment labels into the model's input layer. Sentence word index IDs are then mapped to word vectors, followed by model training, validation, and evaluation.

3.1 Text Representation

Keras's Embedding layer provides three vector training methods controlled by setting different trainable parameter values: (1) Using review sentence word index ID numbers as word vector input with trainable=True for word vector training; (2) Using Word2vec pre-trained word vectors as Embedding layer weight values with trainable=False, preventing vector updates; (3) Using Word2vec pre-trained word vectors as initial Embedding layer weights with trainable=True, allowing continued vector training and updates. These three text representation methods are primarily formed by controlling Embedding layer data input and trainable parameters. Subsequent application sections will compare and analyze these three approaches.

3.1.1 Word2vec Word Vector Training

Word2vec employs CBOW or Skip-Gram models, utilizing contextual information to predict current word semantics, generating word vectors that map into vector space. Semantic similarity between words is obtained by calculating distances between spatial vectors. Word2vec overcomes the "vocabulary gap" and

"dimensional disaster" limitations of One-Hot vectors, better facilitating text representation [20-21].

The Python version of Word2vec [22] is used for training, with the following process: (1) Import the Word2vec class from gensim.models; (2) Set word vector dimension to 256 and training window size to 8; (3) Load segmented text files using the Text8Corpus method; (4) Define a model variable, select Skip-Gram training mode, and pass parameters and text data to the Word2Vec method; (5) Execute model.wv.save_{{{word2vec}}_{{format}}} to begin training and save results. The training outcomes contain words with corresponding vectors saved as text documents, with vector values normalized.

3.1.2 Generating Word Vector Matrix

The word vector document storage structure is "word + ' ' + vector," with each word and its vector on one line separated by spaces (e.g., "正品 0.532237 0.139422 0.062200 ……"). Based on this structure, each line is read cyclically, splitting words and vectors by spaces to generate a word vector dictionary with words as keys and vectors as values: "{key: word, value: vector}."

Keras's text preprocessing Tokenizer class performs word frequency statistics on all words, assigning each word a unique ID number in frequency order to generate a word dictionary: "{key: word, value: ID}." The word vector matrix is constructed by linking the word vector dictionary and word dictionary through the "word" key, with all word vectors sorted by their corresponding ID numbers. Finally, vectors at matrix positions 0 and dictionary length+1 are added, initialized with random values between (0, 0.001).

3.1.3 Generating Sentence Word Index Matrix

The Tokenizer class's texts_{{{to}}*{{sequences}}} function and pad*{sequences} class replace each segmented word in a sentence with its corresponding ID from the word dictionary, generating the sentence word index matrix that converts text sequences to ID sequences (e.g., "[[241 5775 2247 …,0 0 0][603 154 1 …,0 0 0]…]"). Since review sentences vary in length, uniform sentence index length is required. The specific length can be determined based on sentence length distribution—when shorter than the set value, sequences are padded with 0s at the end; when longer, excess portions are truncated.

3.2 Classification Model Construction

For this study's tasks, multiple classification algorithms were experimentally compared, ultimately selecting the TextCNN algorithm with superior classification performance. The classification model first performs mapping conversion between sentence word indices and word vectors, then conducts convolution operations on word vectors.

3.2.1 TextCNN Classification Model Structure

The TextCNN classification model includes InputLayer, Embedding, SpatialDropout1D, Conv1D, MaxPool1D, Concatenate, Flatten, Dropout, and Dense (fully connected layer with activation function). Its structure is shown in Figure 3 [Figure 3: see original paper]. Three Conv1D convolution kernels have lengths of 2, 3, and 4 respectively, with default stride 1 and relu activation functions. Each MaxPool1D pooling layer outputs the word vector with maximum features [17].

3.2.2 Sentence Word Index and Word Vector Mapping

Before training, the Embedding layer performs a lookup operation in the word vector matrix using segmented index IDs to obtain each word's vector, sequentially combining them into the sentence's word vector matrix. A mapping example is shown in Figure 4 [Figure 4: see original paper], where the left side is the sentence word index matrix, the middle is all words' word vector matrix, and the right side is the sentence vector matrix, with wi (i=0,1,…,6) being 256-dimensional vectors [23].

3.2.3 Word Vector Convolution

Each sentence matrix has size (m, 256), where m is the number of word vectors in the sentence and 256 is the vector dimension. When Conv1D kernel length is 2, convolution operates on m word vectors by moving vertically with size (2, 256), effectively extracting local correlation features for N-gram=2 word pairs—this is why convolutional neural networks process text efficiently and rapidly. Ultimately, three parallel Conv1D convolutions produce m-1, m-2, and m-3 post-convolution vectors, each undergoing MaxPool1D pooling to output one feature-maximized word vector. These undergo Concatenate processing to obtain three sentence feature vectors sent to the classifier. The classifier activation function is sigmoid, loss function is binary_{crossentropy}, and optimizer is adam.

4 Application and Analysis

4.1 Experimental Data

Commodity review data were scraped from JD.com's website. Product URLs follow the format: "http://item.jd.com/1658812308.html," where the numeric portion represents the product ID. Based on product ID rules, random product IDs were generated to form a product address list, with validity verification performed. Valid addresses were used to scrape positive and negative review data from product pages. Analyzing the HTML tag information on review pages, "content" (review content) and "rating" tags (positive=3, negative=1) were primarily extracted, with tags and symbols cleaned from content. The review content and rating information were saved as relational data. Using this method, approximately 60,000 JD.com review data were randomly collected in February

2018. After data cleaning, statistical analysis of positive/negative reviews, and filtering, 31,120 experimental corpus data were selected and labeled with positive/negative sentiment, including 15,560 positive and 15,560 negative reviews (balanced). Data were split 6:2:2 into training (18,672), validation (6,224), and test sets (6,224).

During model training, the ReduceLROnPlateau method for dynamic learning rate adjustment and the EarlyStopping method to prevent overfitting were used. These cooperatively monitor whether "validation loss (val_{loss})" continues decreasing and make corresponding training adjustments. Beyond test set accuracy, model performance was evaluated using the ROC curve's AUC value, where values closer to 1 indicate better classification performance.

4.2 Sentence Length Selection

After corpus segmentation, sentence lengths were statistically analyzed to generate a length distribution histogram shown in Figure 5 [Figure 5: see original paper]. Sentence lengths primarily distribute between 0-140 words, with the most sentences (~7,000) containing approximately 8 words. As sentence length increases, sentence count decreases, characterizing a short-text classification problem. Experiments compared lengths of 16, 32, 64, and 128 words selected from the 8-140 range. Using Embedding layer training method (Word2vec pre-trained vectors as Embedding layer weights with trainable=False) and the TextCNN classification model, results are shown in Table 1 and comparison curves in Figure 6 [Figure 6: see original paper].

Table 1 shows that model accuracy and ROC AUC values increase with sentence length, peaking at length 128 with 92.29% accuracy and 0.9772 AUC—1.93% and 0.0124 higher than length 16, respectively—while iteration count ranks third, not increasing significantly. Therefore, length 128 is appropriate.

4.3 Comparison of Different Text Representation Methods

Four text representation methods were compared: (1) "Embedding (Emb)," (2) "Word2vec (Vec)," (3) "Word2vec + Embedding (Vec+Emb)," and a variation of method , "Word2vec*TF-IDF [24] (Vec*TF-IDF)" (word vectors multiplied by TF-IDF weights). Each was tested with the TextCNN classification model, with results in Table 2 and comparison curves in Figure 7 [Figure 7: see original paper].

Results show: (1) Method "Emb" has 0.16% lower test accuracy than method "Vec," with greater overfitting on the training set. (2) Method "Vec" achieves second-best test accuracy (92.29%) and AUC (0.9772), with best training set fit. (3) Method "Vec+Emb" achieves highest test accuracy (94.02%) and AUC (0.9827)—1.73% and 0.0055 higher than method —with moderate training set overfitting and second-ranked iteration count. (4) "Vec*TF-IDF" shows lowest test accuracy and AUC but good training set fit. Method is more conducive to

accuracy improvement, while method better resists overfitting. The next step compares different classification algorithms to further validate these effects.

4.4 Comparison of Different Classification Algorithms

CNN, LSTM, GRU, Bi_{LSTM}, Bi_{GRU}, and TextCNN classification algorithms were tested with text representation methods and . Results are shown in Table 3 and comparison curves in Figure 8 [Figure 8: see original paper].

Results indicate: (1) All classification algorithms paired with method achieve higher training, validation, and test accuracy and AUC values than with method , with fewer iterations and faster convergence. TextCNN with method achieves highest test accuracy (94.02%) and AUC (0.9827)—0.53% and 0.0016 higher than second-ranked GRU, and 1.13% and 0.0050 higher than the best LSTM with method . (2) All algorithms with method show greater training set overfitting than with method , but TextCNN exhibits the least overfitting. Overall, with the current corpus size, using sentence length 128, Word2vec pre-trained 256-dimensional vectors as initial values combined with continued Embedding layer training and TextCNN classification improves both accuracy and average model performance, effectively achieving positive/negative sentiment classification with strong generalization capability.

5 Conclusion

This study utilizes Word2vec embedding technology to pre-train corpus word vectors, inputs them into Keras API's Embedding layer, and implements three commodity review text representation methods by controlling the Embedding layer's trainable parameter based on sentence word index sequences. Different text representations are matched with different classification algorithms, ultimately identifying an ideal classification model: the text representation method that continues training Word2vec vectors through the Embedding layer combined with TextCNN. This approach's main advantages are: (1) It compensates for Word2vec pre-trained vectors' accuracy reduction and extended training time, and mitigates Embedding-only training's overfitting risk; (2) Inputting Word2vec pre-trained vectors into the Embedding layer for continued training shortens training time despite iterative vector updates; (3) No algorithmic improvements are needed—simply setting different Embedding layer parameters reduces implementation difficulty and facilitates practical application.

Limitations include limited commodity review data types and lack of expansion to other e-commerce platforms. Future research will focus on collecting more corpus data, increasing training datasets, and fully mining semantic representations of emoticons and English words in Chinese corpora to further improve classification accuracy.

References

[1] Huang Ren, Zhang Wei. Research on Sentiment Orientation of Internet Product Reviews Based on word2vec [J]. Computer Science, 2016, 43(S1): 387-389.
[2] Xie Faju, Liu Chen, Tang Li. Review of Online Review Sentiment Analysis [J]. Software Guide, 2018, 17(2): 1-4,7.
[3] Wang Zhitao, Yu Zhiwen, Guo Bin, et al. Chinese Micro-blog Sentiment Analysis Based on Dictionary and Rule Set [J]. Computer Engineering and Applications, 2015, 51(8): 218-224.
[4] Zhao Gang, Xu Zan. Research on Sentiment Analysis Model of Product Reviews Based on Machine Learning [J]. Journal of Information Security Research, 2017, 3(2): 166-170.
[5] Guo Bo, Li Shouguang, Wang Hao, et al. Design and Implementation of Comprehensive E-commerce Review Analysis System—Research and Application of Sentiment Analysis and Opinion Mining [J]. Data Analysis and Knowledge Discovery, 2017, 1(12): 1-9.
[6] Rexidanmu · Tuerhongtai, Wushouer · Silamu, Yierxiati · Tuerhong. Uyghur Text Sentiment Analysis Combining Dictionary and Machine Learning Methods [J]. Journal of Chinese Information Processing, 2017, 31(1): 177-183,191.
[7] Wang Xinyu. Research on Tourism Network Evaluation Sentiment Analysis Based on Sentiment Dictionary and Machine Learning [J]. Computer & Digital Engineering, 2016, 44(4): 578-582,766.
[8] Wang Zhengcheng, Li Dandan. Short Text Sentiment Classification Based on Word Vector and Sentiment Ontology [J]. Journal of Zhejiang Sci-Tech University (Social Sciences), 2018, 40(1): 33-38.
[9] Hu Chaoju, Zhao Xiaowei. Sentiment Analysis Based on Word Vector Technology and Hybrid Neural Network [J]. Application Research of Computers, 2018, 35(12): 3556-3559,3574.
[10] Jin Zhigang, Han Yue, Zhu Qi. A Sentiment Analysis Model Combining Deep Learning and Ensemble Learning [J]. Journal of Harbin Institute of Technology, 2018, 50(11): 76-82.
[11] Liu Quan, Liang Bin, Xu Jin, et al. A Deep Hierarchical Network Model for Aspect-Based Sentiment Analysis [J/OL]. [2018-06-08]. http://kns.cnki.net/kcms/detail/11.1826.TP.20171129.2026.006.html.
[12] YOON K. Convolutional Neural Networks for Sentence Classification [C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Stroudsburg: Association for Computational Linguistics, 2014: 1746-1751.
[13] TOMAS M, KAI C, GREG C, et al. Efficient Estimation of Word Representations in Vector Space [EB/OL]. [2018-06-08]. https://arxiv.org/pdf/1301.3781.pdf.
[14] Wang Qinqin, Zhang Yuhong, Li Peipei, et al. Cross-Domain Sentiment Classification Method Based on word2vec [J]. Application Research of Computers, 2018, 35(10): 2911-2915.
[15] Cai Linsen, Peng Chao, Chen Siyuan, et al. Sentiment Analysis Based on Diversified Feature Convolutional Neural Network [J/OL]. [2018-06-08].

https://doi.org/10.19678/j.issn.1000-3428.0050338.
[16] Duan Chuanming. Comparative Analysis of Traditional Sentiment Classification Methods and Deep Learning-Based Methods [J]. Software Guide, 2018, 17(1): 22-24.
[17] Sun Chaohong. Research on Micro-blog Sentiment Classification Based on Recurrent Neural Network [D]. Hangzhou: Zhejiang Sci-Tech University, 2017.
[18] Fan Weihao, Xu Jian. User Pain Point Analysis Based on Network User Review Sentiment Computation—Taking Mobile Phone Reviews as Example [J]. Information Studies: Theory & Application, 2018, 41(1): 94-99.
[19] Keras Chinese Documentation [EB/OL]. [2018-06-08]. http://keras-cn.readthedocs.io/en/latest/.
[20] Xue Weiming, Hou Xia, Li Ning. A Text Classification Method Based on word2vec [J]. Journal of Beijing Information Science & Technology University (Natural Science Edition), 2018, 33(1): 71-75.
[21] Zhu Lei. Research on Text Classification Based on word2vec Word Vectors [D]. Chongqing: Southwest University, 2017.
[22] gensim: models.word2vec – Deep Learning with word2vec [EB/OL]. [2018-06-08]. https://radimrehurek.com/gensim/models/word2vec.html.
[23] What Exactly Are Word Vectors and Embedding? [EB/OL]. [2018-06-08]. https://spaces.ac.cn/archives/4122.
[24] Li Rui, Zhang Qian, Liu Jiayong. Micro-blog Sentiment Analysis Based on Weighted word2vec [J]. Communications Technology, 2017, 50(3): 502-506.

Author Contributions:

Li Wenjiang (ORCID: 0000-0002-9961-9836), Senior Experimentalist, Master's degree, E-mail: 115101850@qq.com: Proposed research ideas, designed research scheme, conducted experimental analysis, drafted initial manuscript.
Chen Shiqin (ORCID: 0000-0002-4277-080X), Associate Research Librarian, Master's degree: Responsible for data collection, cleaning, and classification labeling, revised final manuscript version.

Received Date: 2018-10-10
Publication Date: 2018-12-20
Responsible Editor: Liu Yuanying
Knowledge Management Forum ISSN 2095-5472 CN11-6036/C
E-mail: kmf@mail.las.ac.cn
Website: http://www.kmf.ac.cn

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