A Multi-task Text Classification Model Based on Label Embedding Learning

Yuemei Xu^{1(✉)}, Zuwei Fan^{2,3}, and Han Cao¹

¹ School of Information Science and Technology, Beijing Foreign Studies University,
Beijing, China

xuyuemei@bfsu.edu.cn

² Institute of Information Engineering, Chinese Academy of Science, Beijing, China

³ School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. Different text classification tasks have specific task features and the performance of text classification algorithm is highly affected by these task-specific features. It is crucial for text classification algorithms to extract task-specific features and thus improve the performance of text classification in different text classification tasks. The existing text classification algorithms use the attention-based neural network models to capture contextualized semantic features while ignores the task-specific features. In this paper, a text classification algorithm based on label-improved attention mechanism is proposed by integrating both contextualized semantic and task-specific features. Through label embedding to learn both word vector and modified-TF-IDF matrix, the task-specific features can be extracted and then attention weights are assigned to different words according to the extracted features, so as to improve the effectiveness of the attention-based neural network models on text classification. Experiments are carried on three text classification task data sets to verify the performance of the proposed method, including a six-category question classification data set, a two-category user comment data set, and a five-category sentiment data set. Results show that the proposed method has an average increase of 3.02% and 5.85% in F1 value compared with the existing LSTMAtt and SelfAtt models.

Keywords: Text classification · Label embedding · Attention mechanism · Multi-task

1 Introduction

Thanks to the rapid development of Internet technology, text information on the Internet are explosively increased and has raised higher requirement for data

This work is supported by the Fundamental Research Funds for the Central Universities (No. 2022JJ006).

© The Author(s) 2022

W. Lu et al. (Eds.): CNCERT 2021, CCIS 1506, pp. 211–225, 2022.

https://doi.org/10.1007/978-981-16-9229-1_13

organization and management. Text classification algorithm is one of the efficient ways to manage such huge volumes of text information, and has been applied on different scenarios, such as question classification, sentiment classification, spam detection, topic classification. Different text classification tasks own specific features. It is crucial for text classification algorithms to extract these task-specific features and to guarantee its performance on different classification tasks.

Traditional text classification mainly uses machine learning algorithms, such as logistic regression [7], decision tree [4], support vector machine [5] and naive Bayes [8]. These algorithms have achieved good performance on text classification tasks, but they require manual text features extraction, which is cumbersome and costly. What's more, the manual extraction of text feature cannot be adjusted dynamically when the text classification task changes.

Recently, with the rapid development of deep learning algorithms and models, text classification has gradually shifted from models based on machine learning to deep learning models based on neural network. The neural network models support automatic feature extraction. Therefore, it exhibits better potential in multi-task text classification than the traditional machine learning based models. For example, Kalchbrenner et al. [14] proposed a Dynamic Convolutional Neural Network (DCNN) model in 2014. It was for the first time that Convolutional Neural Networks (CNN) was applied to the field of text classification. Later, Kim proposed a simpler TextCNN model [21]. Subsequently, other neural network models, such as RNN [17], BiGRU [15], and LSTM [6], were proposed to solve the problem of CNN model, which only captures the local semantic features of the text while ignores the relationship between text sequence.

In order to extract features that are more relevant to the text classification task, attention mechanism is introduced on the basis of deep learning models [2]. The attention mechanism enables the deep learning models to focus on the words that are more relevant to the classification task through assigning more weights to the important words. It has been proved that the attention-based text classification algorithms (e.g., LSTMAtt [22], SelfAtt [9]) have achieved better performance than BiGRU and LSTM on text classification tasks. However, these attention-based text classification algorithms assign weights to different words mainly according to the text semantics, without considering the impact of task-specific features. For example, for the sentence “the food quality is decent, but the price is very steep”, the objective words “decent”, “steep” should be assigned more attention weight in sentiment classification task, whereas words of “food” and “price” should be assigned more weight in topic classification task. If the attention weight assignment can be dynamically adaptive to different text classification tasks, the effectiveness of text classification can be greatly improved [19].

Therefore, for the purpose of exploiting both text semantic features and task-specific features, label embedding is introduced to the existing attention-based text classification algorithm in this paper. By considering task-specific features, words and their category labels to which they belong are embedded to a joint space for attention weight learning, so that the attention weight of each word incorporates its importance according to different classification labels. In

specific, a modified TF-IDF matrix generated in a supervised manner for multi-task text classification is used to incorporate category label information for text classification. Then the words having important impact on text classification are found and given greater weight, which helps greatly improve the efficiency of text classification. In order to verify the adaptive of the proposed model on different text classification tasks, experiments are conducted on 3 different data sets, including a TREC data set with six-category question classification, a CR data set with two-category customer review classification and a Stanford SST-1 data set with five-category sentiment classification. The results show that text classification performance of the proposed model is better than the existing LSTMAtt and SelfAtt methods, and the F1 value is improved by 3.02% and 5.85% on average compared with that of LSTMAtt and SelfAtt respectively. Among the three data sets, the proposed model performs in the TREC data set, improving respectively F1 values of 6.18% and 10.86% on average compared with LSTMAtt and SelfAtt.

The rest of this paper is organized as follows. Section 2 reviews a survey of the related work. Section 3 presents the proposed label embedding-based text classification model. Section 4 shows the experiment settings and the experimental results for verifying the effectiveness of our model. Finally, Sect. 5 concludes the paper.

2 Related Work and Background

2.1 Text Classification

The existing text classification research can be divided into two categories: one is traditional text classification methods based on machine learning and the other is deep learning classification methods based on neural network models.

Traditional machine learning based text classification methods need to define feature calculation rules, such as Bag-of-words model (BOW) [11], TF-IDF model [18], topic model [3] and etc., to construct text representations. The BOW model considers the text as a collection of words and constructs text representation in terms of the words that appear in the text. Therefore, the BOW model only reflects the frequency of words but cannot reflect the importance of words. Joachims et al. proposed the TF-IDF model [18]. In specific, TF refers to term frequency and if a term appears in a document more times, its TF value is higher. IDF refers to inverse document frequency and if there are fewer documents containing the term, its IDF is higher. TF-IDF model believes that when a term's TF and IDF are higher, the term is more important and has better classification ability. However, the word representations constructed in BOW or TF-IDF models are independent of each other and the contextual relationship between words and semantic information cannot be well represented. Subsequently, topic model Latent Dirichlet Allocation (LDA) was proposed [3]. LDA constructs the text representation from the perspective of the probabilistic generative model, and obtains the text-topic probability distribution and the word-topic probability distribution. The topic model solves the problem caused in the BOW and TF-IDF models, but still has the problems of high dimension and high sparseness.

After constructing the text representation, the early text classification methods used machine learning algorithms, such as logistic regression [7], decision tree [4], support vector machine [5] and naive Bayes [8] as text classifiers to classify text. These traditional classification methods achieved good results but required to construct text features manually. What's more, the text features constructed by manual extraction cannot adapt to the dynamic adjustment of text classification tasks.

The text classification methods based on deep learning methods do not need to construct the text features manually. They obtain word representations of semantic information by using word embedding models, such as static word embedding models (Word2Vec [13] and Glove [10]) and dynamic word embedding models (ELMO [16]). Then the vector representation of the text is obtained through word representations and is taken as input of neural network models, such as CNN and RNN, to predict the text category. Kalchbrenner [14] applied CNN to the field of text classification for the first time in 2014, which achieved good results. Subsequently, other neural network models, such as RNN [17], BiGRU [15], and LSTM [6], were successively proposed to solve the problem that CNN can only capture the local semantic features of text, but cannot learn sequence correlation. Among these neural network models, LSTM has outstanding performance in text classification tasks and solves the problem of gradient vanishing and gradient explosion. LSTM adopts a gate structure to selectively let information through, therefore, it can effectively capture long-distance information and better express the contextual semantic information of the text.

Fig. 1. Framework of the label embedding-based attention model for text classification.

2.2 Attention Mechanism

Attention mechanism was first proposed in the field of computer vision and then was introduced to deal with the machine translation task [2]. Recently, attention mechanism has been widely used in various natural language processing tasks, such as sentiment analysis, dialogue systems and text classification.

The exiting works mainly used the attention mechanism to help neural network models such as CNN and RNN select more important features. For instance, the ATAE-LSTM model [20] proposed to combine the attention mechanism with LSTM model, where the attention weight was computed over the hidden layers of the LSTM by combining the aspect information. The attention mechanism in ATAE-LSTM model could concentrate on different parts of a sentence if different aspects are taken as input. Lin et al. proposed a sentence-level SelfAtt model [9], which extracted different aspects of the sentence into multiple vector representations and learned the hidden state information between these vector representations through the attention mechanism. Thus, it can extract different key information of the sentence. These existing studies show that the combination of attention mechanism and neural network model can successfully improve the effectiveness of text representation.

However, these existing text classification models mainly use the attention mechanism to learn the implicit relationship between the input and output, for the purpose of further improving the text representation. But the impact of the classification goals of different tasks, namely task categories, on the attention weight is largely ignored.

3 Methodology of Text Classification Model

3.1 Framework Overview

Figure 1 is the flowchart of the proposed label embedding-based attention model (termed as LabelAtt model). The establishment process includes 3 steps: (1) The first step is text vector representation. It is designed to obtain word embedding representation of each word in the text through both the Golve(or word2vector) model and the modified TF-IDF method. Then word vector representation of text document is generated and taken as input of LSTM model to generate the initial text vector representation. (2) The second step is jointly learning of label embedding and attention weight. This step takes the initial text vector representation and label embedding as input. The neural network model then is used to learn the attention weight of each word and output label-aware text representation. (3) The third step is text classification prediction. It takes the label-aware text representation as input and uses a fully connected layer with sigmoid activation function to predict the category of text. These 3 steps will be discussed in the following sections in detail.

3.2 Problem Statement

Let $\mathbf{s} = \{s_1, s_2, \dots, s_N\}$ be a set of text documents, where s_j represents a text j and N is the total number of texts. Let y_j be the text category of s_j . s_j contains a sequence of words $\{w_1, w_2, \dots, w_n\}$, where n is the number of words in s_j . The importance of words in the text classification task is learned through the attention weight mechanism. Words are assigned with different weight values according to their importance relevant to the text classification task. The greater the importance is, the more the weight value is assigned.

$\mathbf{l} = \{l_1, l_2, \dots, l_K\}$ is defined as a set of category labels and K is the total number of categories in text classification tasks. The value of K is different in different text classification tasks. For instance, K equals to 6 in the six-category text classification tasks and the set of category labels can be represented as $\mathbf{l} = \{0, 1, 2, 3, 4, 5\}$. Corresponding to the category set $\mathbf{l}$, we define the category label tensor as $\mathbf{v} = \{v_1, v_2, \dots, v_K\}$, where $\mathbf{v} \in \mathcal{R}^{K \times D}$ and D is the dimension of word embedding vector. $\mathbf{v}$ is initialized randomly and will be iteratively updated during model training.

3.3 Attention Learning on Word Embedding

Text documents are pre-trained by using Golve model and then the word embedding vector of each text, denoted as $\mathbf{e} = \{e_1, e_2, \dots, e_n\}$, is obtained, where $e_i \in \mathcal{R}^D$ is the word embedding of word w_i and D is the dimension of word embedding vector. Furthermore, LSTM is used to pre-encode the vector representation of each text. LSTM is a special recurrent neural network and is capable to capture the long-distance information in the text sequence through the Gates structure. Therefore, the text vector obtained by LSTM encoding can better represent the sequence information and context semantics of the text. The pre-encoded vector representation is defined as the initial text vector representation $H, H \in \mathcal{R}^{n \times D}$.

Given $\mathbf{l}$ as the set of category labels, and we define $B = \{b_1, b_2, \dots, b_n\}$ as the semantic attention weight matrix of $\{w_1, w_2, \dots, w_n\}$, where b_i represents the attention weight of word w_i over K labels, $b_i \in \mathcal{R}^K$. Then the attention weight matrix is calculated based on the label embedding, as showed in Eq. (1).

$$B = [(\mathbf{v} \otimes H^T) \odot \hat{B}]^T \quad (1)$$

where $\hat{B}$ is the normalized matrix of B . The symbol $\odot$ represents the division of each corresponding element in the two matrices.

The max pooling operation is performed over each element b_i in B and then the maximum weight value of each words over all labels is obtained, which is considered to be the attention weight of each word w_i , calculated as:

$$\beta = \text{max_pooling}(b_i) \quad (2)$$

Finally, $\beta = \{\beta_1, \beta_2, \dots, \beta_n\}$ is normalized by a softmax function to ensure that the sum of the attention weights of all words is equal to 1, which is defined

as label attention weight vector $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$. The calculation is shown in Eq. (3).

$$\alpha = \text{softmax}(\beta) \quad (3)$$

Let T_α be the weighted average of α and the initial text vector representation H , calculated in Eq. (4).

$$T_\alpha = \sum_{i=1}^n \alpha_i \times h_i \quad (4)$$

Where α_i is the label attention weight of w_i and h_i is the initial text vector representation of w_i , $T_\alpha \in \mathcal{R}^D$.

We feed T_α into our model to predict the text classification distribution of the input text.

$$\tilde{y}_j^\alpha = \text{softmax}(T_\alpha) \quad (5)$$

where $\tilde{y}_j^\alpha$ is the text classification label of s_j predicted by our model.

The cost function is defined as the average cross entropy that measures the difference between the category label predicted by our model and the classification annotation of text s_j , calculated as:

$$f_{loss(\alpha)} = - \sum_{j=1}^N y_j \log \tilde{y}_j^\alpha \quad (6)$$

Where N is the number of texts in training set and y_j is the classification annotation of text s_j .

3.4 Attention Learning on Modified TF-IDF Matrix

A modified TF-IDF matrix is used to learn the importance of words for classification task and the generated TF-IDF matrix can be used to supervise the attention weight learning of each word.

TF-IDF is one of the most commonly used weighting metrics to measure the relationship of words to documents. The modified TF-IDF method proposed by Jin et al. [23] uses the known classification information of training data set to compute the weights of each word term in different text classification tasks. Each word appears in the training data set corresponds to K TF-IDF values and each value corresponds to the texts associated to one category label.

Let $\mathbf{tfi} = \{tfi_1, tfi_2, \dots, tfi_n\}$ be the TF-IDF matrix of word sequence $\{w_1, w_2, \dots, w_n\}$ of text s_j , $\mathbf{tfi} \in \mathcal{R}^{n \times K}$, where $tfi_i = \{tfi_{i,1}, tfi_{i,2}, \dots, tfi_{i,n}\}$ represents the TF-IDF weight vector of word w_i over K category labels. Let $tf(w_i, l_k)$ be the frequency of word w_i in text category l_k , which is calculated as:

$$tf(w_i, l_k) = \frac{nw_{i,k}}{\sum_q nw_{q,k}} \quad (7)$$

where $nw_{i,k}$ is the number of word w_i occurrences in the text category l_k , and $\sum_q nw_{q,k}$ is the number of all words in the text category l_k .

Inverse document frequency is defined to eliminate the influence of common words, denoted as $idf(w_i, l_k)$ intuitively, if a word w_i has less other text category containing it, its representativeness of category l_k should be given higher value. $idf(w_i, l_k)$ is calculated as follows.

$$idf(w_i, l_k) = \log \frac{|j : s_j \in l_k|}{|k : w_i \in l_k| + 1} \quad (8)$$

where $|j : s_j \in l_k|$ is the number of texts in category l_k and $|k : w_i \in l_k|$ is the number of categories containing the word w_i . If a word doesn't appear in the training data set, the value of $|k : w_i \in l_k|$ is zero and 1 will be added to avoid zero division.

By integrating Eq. (7) and Eq. (8), we can get the TF-IDF weight vector of word w_i in category l_k as:

$$tfi_{i,k} = tf(w_i, l_k) \times idf(w_i, l_k) \quad (9)$$

For each category l_k , we repeat the above calculations as shown in Eq. (7)–Eq. (9) and then obtain the modified TF-IDF matrix. Similar with the operations in the attention learning on word embedding vector, LSTM is used to pre-encode the TF-IDF matrix and the output of LSTM is considered as the text vector representation under modified TF-IDF, denoted as TFI .

Then the initial text vector representation H in Eq. (1)–Eq. (3) is replaced by TFI . After attention weight matrix calculation, maximum pooling operation and softmax normalization, TF-IDF attention weight vector is obtained and denoted as $\gamma = \{\gamma_1, \gamma_2, \dots, \gamma_n\}$.

Let T_{tf} be the weighted average of γ and TFI , which is calculated in Eq. (10).

$$T_{tf} = \sum_{i=1}^n \gamma_i \times tfi_i \quad (10)$$

We feed T_{tf} into our model to predict the text classification distribution of the input text.

$$\tilde{y}_j^{tf} = \text{softmax}(T_{tf}) \quad (11)$$

where $\tilde{y}_j^{tf}$ is the text classification label of s_j predicted according to T_{tf} .

Similarly, the cost function is defined as the average cross entropy that measures the difference between the category label predicted by our model and the classification annotation of text s_j , calculated as:

$$f_{loss(tf)} = - \sum_{j=1}^N y_j \log \tilde{y}_j^{tf} \quad (12)$$

Where N is the number of texts in training set and y_j is the classification annotation of text s_j .

In order to learn the label-aware text vector representation according to both contextual semantic and the task-specific features, the final cost function is defined as follows:

$$\text{loss} = \lambda \times f_{\text{loss}(\alpha)} + (1 - \lambda) \times f_{\text{loss}(tf)} \quad (13)$$

where λ is a tradeoff parameter ($0 \leq \lambda \leq 1$). The weight of $f_{\text{loss}(\alpha)}$ can be increased by choosing a larger value of λ .

4 Experiment Evaluation

In this section, in order to evaluate the performance of the proposed model (termed as LabelAtt) on different text classification tasks, three different datasets are selected. On each dataset, LabelAtt model is compared with two existing models, namely LSTMAtt [20] and SelfAtt [1] models, which are based on attention mechanism but without label embedding. LSTMAtt [20] and SelfAtt [1] are regarded as the baseline models to verify the effectiveness of LabelAtt on improvement of attention mechanism.

4.1 Dataset and Parameter Settings

The experiment datasets are TREC [22], Customer Review (CR) [23], and SST-1 [24]. Detailed information on these datasets is shown in Table 1.

Table 1. Information of the datasets TREC, CR and SST-1.

Dataset	Number of labels	Numbers of samples	Training set	Test set
TREC	6	5952	5452	500
CR	2	3769	2638	1131
SST-1	5	10754	8544	2210

The TREC data set is a 6-category question classification data set containing 6 labels: definition, entity, person, place, data, and abbreviation. The CR data set is a 2-category user comment data set. All the comments in this data set have been labeled either positive or negative. The SST-1 is Stanford Sentiment Treebank, a 5-category sentiment data set. The data set includes five categories of text of different levels of emotion score, and divides the score from 0 to 1: 0–0.2, 0.2–0.4, 0.4–0.6, 0.6–0.8, and 0.8–1, which respectively representing very negative, negative, neutral, positive, and very positive.

The dimension of the hidden layer is 100. The dimension of the word embedding vector needs to be identical with the hidden layer, which is also equal to 100; batch size is 16; epoch is 100; The learning rate of training on TREC and CR datasets is set to 1×10^{-3} , and that of training on SST-1 dataset is set

to 2×10^{-5} . The penalty term coefficient of LabelAtt and SelfAtt is 0.1, and LSTMAtt does not introduce the penalty term coefficient. In the experiment, we used Accuracy, Precision, Recall, and F1 to evaluate the performance of the classification model.

4.2 Experiment Result

The prediction accuracy of each text classification model on TREC, CR and SST-1 datasets is shown in Table 2.

Table 2. Comparison of accuracy of each model on TREC, CR and SST-1 %.

	TREC	CR	SST-1
LSTMAtt	88.8	79.92	41.13
SelfAtt	86.00	78.16	35.02
LabelAtt	92.2	81.96	43.17

It is clear that the prediction accuracy of LabelAtt model on TREC dataset reaches 92.2%, which is much better than that of LSTMAtt model (88.8%) and SelfAtt model (86%). The performance is improved by 2.49% and 3.4% respectively. The prediction accuracy of LabelAtt model on CR dataset is 81.96%, which is about 4.9% higher than that of SelfAtt model (78.16%). The prediction accuracy of LabelAtt model on SST-1 dataset is 43.17%, which is about 23.3% higher than that of SelfAtt model (35.02%).

Table 3. Comparison of Recall, Precision and F1 of each model on three datasets (%).

	Recal			Precision			F1		
	TREC	CR	SST-1	TREC	CR	SST-1	TREC	CR	SST-1
LSTMAtt	85.46	81.97	36.42	56.16	85.9	13.64	65.11	83.89	19.74
SelfAtt	82.71	81.14	33.49	50.47	84.05	11.82	60.43	82.57	17.26
LabelAtt	86.43	86.55	37.51	63.06	85.36	14.23	71.29	85.95	20.57

Recall, Precision, and F1 of each text classification model on TREC, CR and SST-1 dataset are shown in Table 3 respectively. F1 is used as the comprehensive evaluation index of Recall and Precision. LabelAtt model obtained the highest F1 on the three datasets, followed by LSTMAtt model, and SelfAtt model performed the worst. Compared with LSTMAtt model, our model improves F1 by 3.02% on average on the three datasets, and the highest improvement is 6.18% on TREC dataset. Compared with SelfAtt model, F1 of our model on the three datasets has been improved by 5.85% on average, and that on TREC dataset

has been improved by 10.86% at the highest. The results indicate that our model can improve the attention mechanism through label embedding, obtain text representation closer to the text classification goal, and improve the effect of text classification. Meanwhile, the applicability of our model for different classification tasks is proved by experiments on datasets of different classification tasks.

4.3 Text Classification Visualization Analysis

Visualization analysis is adopted to verify the effectiveness of LabelAtt model on text classification tasks and demonstrate the characteristics of task classification in attention weight assignment. Considering that the performance of LSTMAtt is better than that of SelfAtt, the visual analysis in this section only shows the comparison between LabelAtt model and LSTMAtt model.

In this paper, the t-SNE [12] dimension reduction method is applied to reduce the high-dimensional text representation learned by the model to two-dimensional vectors, and then a scatter plot is drawn for visualization.

Fig. 2. Classification effect of LabelAtt model during different epochs (TREC data set). (Color figure online)

Table 4. Comparison of accuracy of LabelAtt model and LSTMAtt model on TREC data set %.

	Epoch 1	Epoch 5	Epoch 20	Epoch 25
LabelAtt	0.766	0.866	0.912	0.918
LSTMAtt	0.726	0.834	0.880	0.882

Figure 2 shows the effect of LabelAtt model on text classification in TREC data set as the training times increase. For comparison, Fig. 3 shows the classification effect of LSTMAtt model in the same data set and the same training stage. As shown in Fig. 2, when the epoch is equal to 1, the text representation of LabelAtt model is in a chaotic state. With the increase of training times, when the epoch is equal to 5, the edge of each label becomes clear and clustered by color. When the epoch is equal to 20 and 25, the effect of text classification of LabelAtt model has reached a good state with relatively clear edges and few scattered points.

Fig. 3. Classification effect of LSTMAtt model during different epochs (TREC data set). (Color figure online)

As shown in Fig. 3, when the Epoch is equal to 1, the text representation of the LSTMAtt model is also in a chaotic state. With the increase of training times, when the Epoch is equal to 5, the edge of each label become clearer and clustered by color. When the Epoch is equal to 20, compared with LabelAtt model in Fig. 2 which the text classification effect has reached a better state, there are still more points scattered in LSTMAtt model. In order to examine the visual text classification results as shown in Fig. 2 and Fig. 3, the text classification accuracy of LabelAtt model and LSTMAtt model on the TERC data set when the Epoch is equal to 1, 5, 20 and 25 are shown in Table 4. It can be found that the accuracy of LabelAtt model is significantly higher than that of LSTMAtt model at the beginning of training. When Epoch = 1, the accuracy of LabelAtt model is 0.714 whereas the accuracy of LSTMAtt model is 0.298. With the increase of training times, the accuracy of both LabelAtt model and LSTMAtt model gradually improves, while LabelAtt model converges faster and has higher accuracy.

The result indicates that with the increase of training times, the convergence rate of LabelAtt model is faster and the classification accuracy is higher.

5 Conclusion

In this paper, we exploit the problem of multi-task text classification and introduce a novel label embedding based model to solve the problem. With the proposed model, we could effectively (1) extract the semantic features of context and the task-specific features (2) adapt to different text classification datasets (3) improve the convergence rate of text classification. We evaluate our model on three different datasets with different classification tasks. The results show that the proposed model is effective for text classification, revealing more in-depth information and more accuracy text classification results than the state-of-the-art methods.

References

1. Akata, Z., Perronnin, F., Harchaoui, Z., Schmid, C.: Label-embedding for image classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(7), 1425–1438 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. *Comput. Sci.* **9**(8), 835–841 (2014)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. *J. Mach. Learn.* **3**(8), 993–1022 (2012)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. *Wadsworth. Biometrics* **40**(3), 358–367 (1984)
5. Burges, C.J.: A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **2**(2), 121–167 (1998)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**(8), 1735–1780 (1997)
7. Ifrim, G., Bakir, G., Weikum, G.: Fast logistic regression for text categorization with variable-length n-grams. In: *ACM SIGKDD*, pp. 354–362. ACM (2008)

8. Kim, S.B., Han, K.S., Rim, H.C., Myaeng, S.H.: Some effective techniques for Naive Bayes text classification. *IEEE Trans. Knowl. Data Eng.* **18**(1), 1457–1466 (2006)
9. Lin, Z., et al.: A structured self-attentive sentence embedding. *arXiv Preprint [arXiv:1703.03130](https://arxiv.org/abs/1703.03130)* (2018)
10. Liu, Y., Liu, Z., Chua, T.S., Sun, M.: Topical word embeddings. In: *AAAI Conference on Artificial Intelligence*, pp. 2418–2424. ACM (2015)
11. Wallach, H.M.: Topic modeling: beyond bag-of-words. In: *International Conference on Machine Learning*, pp. 977–984. ACM (2006)
12. Van der Maaten, L., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**(11), 2579–2605 (2008)
13. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. *arXiv Preprint, [arXiv:1301.3781](https://arxiv.org/abs/1301.3781)* (2017)
14. Kalchbrenner, N., Grefenstette, E., Blunsom, P.: A convolutional neural network for modelling sentences. In: *Proceedings of the Conference 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 1252–1262. ACM (2014)
15. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines Vinod Nair. In: *Proceedings of the 27th International Conference on Machine Learning*, pp. 807–814. ACM (2010)
16. Peters, M.E., et al.: Deep contextualized word representations. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, vol. 3, pp. 567–577. ACM (2018)
17. Socher, R., Lin, C.C.Y., Ng, A.Y., Manning, C.D.: Parsing natural scenes and natural language with recursive neural networks. In: *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 129–136. ACM (2011)
18. Joachims, T.: A probabilistic analysis of the Rocchio algorithm with TfIdf for text categorization. *Computer Science Technical Report, CMU-CS-96-118* (1996)
19. Wang, G., et al.: Joint embedding of words and labels for text classification. *arXiv preprint [arXiv: 1805.04174](https://arxiv.org/abs/1805.04174)* (2018)
20. Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: *Conference on Empirical Methods in Natural Language Processing*, pp. 606–615. ACM (2016)
21. Chen, Y.: Convolutional neural networks for sentence classification. *arXiv Preprint [arXiv: 1408.5882](https://arxiv.org/abs/1408.5882)* (2015)
22. Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 606–615. IEEE (2016)
23. Jin, Z., Lai, X., Cao, J.: Multi-label sentiment analysis base on BERT with modified TD-IDF. In: *International Symposium on Product Compliance Engineering-Asia*, pp. 865–871. IEEE (2020)
24. Richard, S., Alex, P., Jean, Y.W., et al.: Stanford Sentiment Treebank dataset. <http://nlp.stanford.edu/sentiment>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.