Information Sciences 518 (2020) 39–55

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Leveraging multiple features for document sentiment classification

Li Kong[a], Chuanyi Li[a,*], Jidong Ge[a], FeiFei Zhang[a], Yi Feng[a], Zhongjin Li[b], Bin Luo[a]

[a] *State Key Laboratory for Novel Software Technology, Software Institute, Nanjing University, China, 210093*
[b] *School of Computer Science and Technology, Hangzhou DIanzi University, China, 310000*

A R T I C L E   I N F O

Article history:
Received 12 April 2019
Revised 12 November 2019
Accepted 6 January 2020
Available online 8 January 2020

Keywords:
Sentiment classification
Deep learning
Feature engineering

A B S T R A C T

Sentiment classification is an important research task in Natural Language Processing. To fulfill this type of classification, previous works have focused on leveraging task-specific features. However, they only notice part of the related features. Also, state-of-the-art methods based on neural networks often ignore traditional features. This paper proposes a novel text sentiment classification method that learns the representation of texts by hierarchically incorporating multiple features. More specifically, we design different representations for sentiment words according to the polarity of labeled texts and whether negation exists; we distinguish words with different part-of-speech tags; emoticons, if there are, are to optimize the word vectors obtained in the previous step; apart from word embeddings, character embeddings are also trained. We use a deep neural network to get a sentence-level representation from both word and character sequence. For documents with at least two sentences, we use a hierarchical structure and design a rule to give more weight to import sentences empirically to get a document-level representation. Experimental results on open datasets demonstrate that our method could effectively improve the sentiment classification performance compared with the basic models and state-of-the-art methods.

1. Introduction

Sentiment analysis, also referred to as sentiment mining, is an important component of Natural Language Processing (NLP), with the aim of helping users analyze and reason the affection contained in the subjective texts. Sentiment analysis has great practical significance since people's opinions have great effect on decision-making [1]. The government can collect people's attitudes towards a policy so that it can better serve the public [2]; Consumers can make suitable purchasing decisions by identifying the sentiment orientation of a large number of online product reviews [3]; Companies are interested in customers' comments on a new product to see whether improvements are required to appeal to the public so that they can improve the quality of their products and boost their sales eventually [4]. It has drawn great attention in both academia and industry. Sentiment classification is a special case of text classification task where the categories are sentiment orientations. According to the demands, texts are categorized as subjective or objective; positive, negative or neutral [5]. Generally, sentiments and opinions can be analyzed at different levels of granularity. Sometimes fine-grained labels are required such as

* Corresponding author.
E-mail address: lcy@nju.edu.cn (C. Li).

https://doi.org/10.1016/j.ins.2020.01.012
0020-0255/© 2020 Elsevier Inc. All rights reserved.

"happiness", "like", "sadness", "dislike" or one to five/ ten star ratings [6]. Current research mainly focuses on coarse-grained classification, but given enough labeled training data, fine-grained classification can be realized in similar ways. According to the specific requirements, we can further divide the task into sentence-level [7], document-level and aspect-level [8]. Sometimes we want an overall sentiment orientation of a document instead of sentiment of a single sentence or opinion towards a target, for example, we need to know whether a person is for or against a new policy, a political party by predicting the sentiment orientation of a whole tweet he (she) posts online [9]. Document sentiment classification is of great importance in our life.

Nowadays, neural networks (NN), especially deep neural networks (DNN), have achieved notable performance in Computer Vision (CV) and becomes an innovation in machine learning. A common NN consists of an input layer, several hidden layers and an output layer. DNN is widely used in many NLP tasks, including machine translation, named entity recognition and so on, including the sentiment classification task. It is a powerful tool in the field of Artificial Intelligence (AI) in that it can automatically learn syntactic and semantic information [10] from large amount of corpus. Its ability of identifying and disentangling the underlying explanatory factors hidden in the observed milieu of low-level sensory data helps get rich and desirable features for the following task [11]. Convolutional Neural Network (CNN [12]) and Recurrent Neural Network (RNN [11]) are the two most popular DNN models.

Since NLP is more than machine learning models, we should also consider characteristic of the data and specific needs of the task. So far, several features have been noticed in text sentiment classification, including:

Word sentiment: Sentiment words, including positive words and negative words, can directly convey the author's attitude to some extent. There are some universal sentiment lexicons (also referred to as affective lexicons) [13] and we can use a sentiment lexicon to extract sentiment information to guide sentiment classification.

POS tag: Words of different part-of-speech (POS) tags contribute differently to the semantics and sentiment expression of the text. For instance, verb, adverb, noun, adjective usually contain more information than conjunction, preposition and pronoun. This motivates researchers to take POS tags into consideration in different ways while learning word representations [14].

Emoticon: Recently, online texts, like tweets or online reviews, are allowed to contain various emoticons, like 👏 ("applause") or 😡 ("angry"). Most of the time we can directly see what the emotion is since they are designed in the form of human face and are indicators of how people feel. People tend to use an emoticon that conveys the same sentiment orientation as the text itself.

In addition, unlike other alphabetic writing systems such as English, languages such as Chinese are logosyllabic. A character can be a meaningful word on its own or as part of a polysyllabic word. Single characters also contain rich semantic information and the character sequence can also be transformed into an input matrix. In addition, character embeddings can also be used to solve the out-of-vocabulary (OOV) problem. If a word does not appear in the pre-trained vocabulary lookup table, average of embeddings of characters in the word can be regarded as its word embedding as a substitute. To adapt the method to alphabetic writing systems, we can also split words into single letters and train an embedding for each letter. We use the term "**character**" to refer to both a character and a letter.

In this paper, we focus on text sentiment classification at document level following a word/character-sentence-document structure. If we treat the whole document as a long sentence and input it to a model, it will be difficult for the model to learn from the previous words since the distance between the current words and previous words is too large. As a result, much information will be lost. So we adopt a two-step strategy to firstly learn sentence representations and next the document representation. At sentence level we consider more task-specific features than existing works. Specifically, we incorporate sentiment word information, POS tag information, emoticon information and character information into a deep neural network, including CNN and RNN model in our work. The way of learning sentence-level representation can also be regarded as a process of resource fusion [15], which can be applied in other tasks. The way of combining texts, emoticons and other information can also be used in combing information from different sources or modes. At document level, if the text consists of more than one sentence, we adopt a hierarchical structure [16] to learn an embedding for the document according to the sentence sequence. Besides, according to commonsense, when we write, we tend to highlight the key point in the beginning or the end. So we make a 'man-made rule' that gives the 1) first sentence 2) the last sentence and 3) sentences containing more sentiment words and typical emoticons extra weight to get a weighted average of embedding of each sentence in the document. Embeddings obtained from the two methods are then concatenated as the final document representation. The thought of combining empiricism (e.g., machine learning) and rationalism (e.g., rules) is also applicable to other NLP tasks.

The overall process of our model is shown in Fig. 1. Given a document, some pre-processing steps are required, including splitting the document into single sentences, word segmentation (for logosyllabic languages), removing stopwords, part-of-speech tagging and sorting emoticons. The sentence-level representation involves words, characters, POS tags and emoticons. The document-level representation is the combination of sentence representations based on a neural network and the customized rule mentioned above. A sentiment classifier is trained to predict the orientation of the document.

The main contributions of our work are as follows:

- Mix strategy at document level: We leverage a hierarchical structure and apply a man-made rule together to combine representation for each sentence into a document-level representation for document sentiment classification;

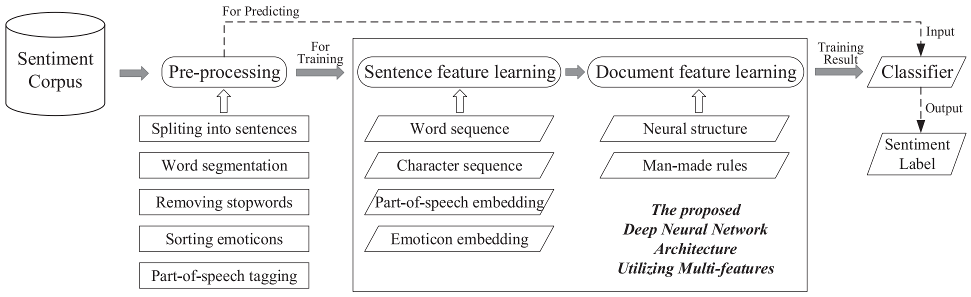


Fig. 1. The overview of the proposed method.

- Multiple features at sentence level: We incorporate sentiment environment, POS tag, emoticon information and character information into a deep neural model to extract sentence-level features for further combination;
- Different embeddings for words: We consider sentiment orientation of words and negation to design different embeddings for words. We also use embedding of characters to define embedding of words not in pre-trained vocabulary table;
- Verification on real datasets: Experimental results from public datasets demonstrate that our model considering multiple features can significantly improve the document sentiment classification performance.

The rest of the paper is organized as follows. Section II introduces the related work on text sentiment classification. Some background knowledge and problem definition are provided in Section III. In Section IV we will introduce our approach in the first stage that combines multiple features with a deep neural model to get the sentence-level representation. Section V tries a mix strategy which treats document as a hierarchical structure and combines a man-made rule to learn the document-level representation. The experimental results conducted on public Weibo datasets are provided in Section VI. Section VII concludes the paper and presents future work.

2. Related work

Sentiment analysis is a classical NLP task that aims to study the emotions, opinions, evaluations, appraisals, and attitudes of people from text content. There are two types of methods for text sentiment classification. The first is a heuristic method based on sentiment knowledge. For example, a simplest way is to aggregate the polarity of individual words to determine the overall sentiment score of a document with the help of a sentiment lexicon such as LIWC [17] and SentiStrength [18], sometimes along with semantic rules [19,20]. The second type is the machine learning method based on features, which usually obtains better performance. Pang et al made the first attempt to successfully introduce machine learning to sentiment classification and tried Support Vector Machine (SVM), Maximum Entropy (ME) and Naive Bayes (NB) [21]. Many works have explored features useful for sentiment analysis. Martineau used Delta TFIDF to weight word scores before applying bag-of-words features [22]. Word bigram features and log-count ratio are proven to be effective when included in a simple classifier [23]. To realize unsupervised classification, Lin et al adopted Latent Dirichlet Allocation (LDA) and achieved promising results [24].

With the development of deep neural network in recent years, CNN, RNN and their variants are also widely used in sentiment classification. Kalchbrenner proposed Dynamic CNN to learn the structure of sentences so that sentiment semantics can be better obtained [25]. Chen et al also combined multiple attentions with a RNN network non-linearly to strengthen the power of capturing sentiment features [26]. Zhang et al proposed a three-way enhanced CNN, which uses SVM with NB features (NB-SVM) to optimize CNN. Specifically, they construct a component named confidence divider and design a confidence function to distinguish the classification quality of CNN. NB-SVM is utilized to reclassify the predictions with weak confidence [27]. Ruder et al utilized review information to assist aspect-level sentiment classification with a hierarchical bidirectional Long Short-Term Memory (LSTM), a popular variant of RNN [28]. In sentence-level classification, Qian et al attempted to model the linguistic role of sentiment lexicons, negation words, and intensity words based on a LSTM [29]. In some works, CNN and RNN are creatively combined together into a hybrid model to better catch features and conduct classification. A pooling layer common in CNN, like max-pooling or average-pooling, can be added to the output of a RNN model instead of doing the original softmax operation directly. Or the convolutional layer with different filter sizes is used first to catch features and then the features are lined into a sequence according to the position in the text so that RNN can be applied [30].

Since texts of different domains are very different, it is important to take task-specific information into account when obtaining representation of the text. In sentiment classification task, sentiment orientation of words is an intuitive indicator and plays an important part. Pre-trained word vectors can be refined according to the sentiment orientation or the intensity score so that words with different orientations can be distinguished [31]. Hu et al investigated word influence in neural sentiment classification by promoting word sentiment and negation as attentions [32]. Fares et al also used sentiment words to conduct sentiment analysis by computing and propagating scores in a lexical-affective graph, which is built with the help of a knowledge base [33]. In social media texts, some emoticons, which can be crawled in the form of texts in square

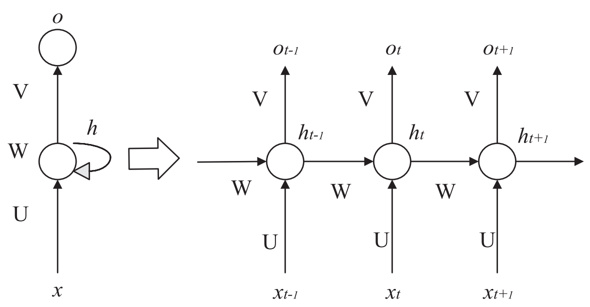


Fig. 2. RNN model.

brackets, like "[applause]" (😊) or "[angry]" (😡), can clearly show people's mood. He et al put forward an Emotion-semantics enhanced Multi-channel CNN with emoticon matrix as another input. Before putting data into a CNN, they conducted a matrix multiplication between the emoticon embedding matrix and the traditional pre-trained word embedding matrix [34]. When it relates to product reviews, product descriptions are used to build a topic distribution for prediction and the model outperforms current topic modeling methods [35]. Many traditional models only use word embeddings to compose sentence representations and ignore the syntactic information of words. Huang et al discovered that encoding POS tag in neural networks can enhance sentence representation. In Recursive NN, they learned tag-specific composition functions and tag embeddings; in LSTM, they used tag embeddings to control the gates [14]. Usually a word in logosyllabic languages has several composed parts – characters. Apart from words, characters also contain rich semantic information. Actually, word character-radical composition has been used in Chinese word embedding algorithms [36,37]. Zhang et al proposed character-level CNN to guide text classification [38]; Conneau et al also proposed an architecture that operates at character level and uses only small convolutions and pooling operations for text processing [39].

However, existing methods usually focus on a few of the text features. Also, when using a deep neural network to predict sentiment orientation, they pay little attention to rules [40] (also can be seen as features). Since when we conduct a specific task, we should focus on not only a novel model but also the particularity of the task and the data, we should consider task-specific rules as well as more task-specific features to learn better document representation.

3. Background and problem definition

Since we choose RNN and CNN as the basic models and add multiple features to both of them to improve classification performance, we will firstly introduce background knowledge of the two models to get readers familiar with our method. Specifically, we use a variant of RNN called Gated Recurrent Unit (GRU) [14,36,37]. Problem definition is presented next and briefly clarifies the input and output of this work. In the following sections, scalars are referred to as lower-case fonts (e.g., index *t*. The exception is *N* which refers to the number of samples in the training set), vectors are referred to as bold lower-case fonts (e.g., input $\boldsymbol{x_t}$ at time *t*) and matrices are referred to as bold upper-case fonts (e.g., weight matrix $\boldsymbol{W}$).

3.1. Gated Recurrent Unit

In traditional neural networks, it is assumed that all inputs and outputs are independent of each other. However, in many cases, each operation depends on the previous calculation results and history information is needed. RNN is a network designed to capture history information in the previous steps with a recurrence structure when processing words in sequence. As is shown in the right part of Fig. 2, this process can be seen as multiple replications of the same neural network, and each neural network module will pass the message to the next one so that history information can be recorded. $\boldsymbol{x_t}$ stands for input at index *t* (the *t*th word in the sentence), $\boldsymbol{o_t}$ stands for the output of the model, $\boldsymbol{h_t}$ stands for the hidden state of the model and similarly when it comes to index (*t-1*) and (*t*+*1*). As we can see, at index *t*, $\boldsymbol{o_t}$ is decided by $\boldsymbol{h_t}$ and $\boldsymbol{h_t}$ is decided by both $\boldsymbol{x_t}$ and $\boldsymbol{h_{t-1}}$. In NLP, it means the representation of current word will be directly affected by its previous words. The three matrices $\boldsymbol{U}$, $\boldsymbol{W}$ and $\boldsymbol{V}$ are the linear relationship parameters of the model and they are shared in the whole RNN network, which is different from common DNN.

However, when the sentence is very long, RNN will fail to learn information from words distant from the current word. Usually in a simple RNN, the memory is very short. Information at step *t* is transmitted to the next step in $\boldsymbol{h_t}$. $\boldsymbol{h_t}$ then takes part in forming $\boldsymbol{h_{t+1}}$ with $\boldsymbol{x_t}$, which means effect of information in the previous step will be reduced. In order to store long-time memory, we use GRU instead. GRU uses a gating mechanism to track the state of sequences instead of separate memory cells. There are two types of gates: the reset gate $\boldsymbol{r}$ and the update gate $\boldsymbol{z}$. They together control how information is updated to the state, as is shown in Fig. 3.

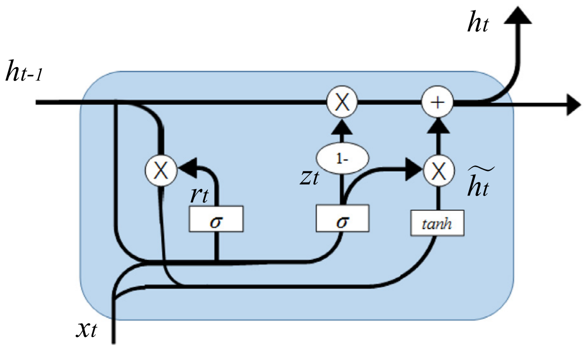


Fig. 3. GRU cell.

The GRU transition equations computing the new state at time t are as follows:

$$\begin{aligned} z_t &= \sigma(W_z[x_t, h_{t-1}] + b_z) \\ r_t &= \sigma(W_r[x_t, h_{t-1}] + b_r) \\ \widetilde{h_t} &= \tanh(W_h[x_t,\ r_t * h_{t-1}] +\ b_h) \\ h_t &= (1 - z_t) * h_{t-1} + z_t *\ \widetilde{h_t} \end{aligned} \quad (1)$$

where $\boldsymbol{W_z}$, $\boldsymbol{W_t}$ and $\boldsymbol{W_h}$ are the weight matrixes, $\boldsymbol{b_z}$, $\boldsymbol{b_t}$ and $\boldsymbol{b_h}$ are bias vectors. From the equations, we can see that new state $\boldsymbol{h_t}$ is decided by both state in the previous time step $\boldsymbol{h_{t-1}}$ and a candidate state $\widetilde{\boldsymbol{h_t}}$. The update gate $\boldsymbol{z_t}$ is used to control how much "old" information in the previous time step is kept and how much "new" information is introduced to current state. The reset gate is used in calculating $\widetilde{\boldsymbol{h_t}}$ and controls how much information from the previous time step is passed to the candidate state.

Attention mechanism is designed to extract important parts to the semantics of the text and has been successfully applied in sequence learning tasks. It is often used together with RNN to improve the performance. Formally, for a sentence s with length m, the sentence vector $\boldsymbol{s}$ is represented by the weighted sum of the hidden states as follows:

$$\begin{aligned} s &= \sum_{t=0}^{m} \alpha_t h_t \\ a_t &= softmax(u_t) \\ u_t &= v^T \tanh(W h_t + b) \end{aligned} \quad (2)$$

where α_t represents the weight of $\boldsymbol{h_t}$ and it can be seen as the importance of the tth word in NLP. The measure $\boldsymbol{u_t}$ shown above is simply lineal and non-lineal transformation of each state and other similarity metrics [41,16] are also viable according to the demand of the specific task. $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{v}$ are learnable parameters.

3.2. Convolutional Neural Network

A CNN is a specially designed feed-forward network widely used in sentence modeling [42–44]. Inspired by the brain of creatures, each hidden unit in CNN can only be connected to part of the input units, which is referred to as local connectivity. In NLP, it means a hidden unit is only related to words in a window size. The basic structure of CNN is present in Fig. 4 [45].

Firstly, the input layer is a lookup layer. It maps each document in the corpus to a list of word vectors, which is the concatenation of word embeddings. In the convolutional layer, the input of each neuron is connected to the local receptive domain in the previous layer and the local feature is also extracted. When it refers to the text, the local domain can be the word sequence in a fixed window size. Once this is done, positional relationship between the local feature and other features is determined. Each computing layer of the network consists of multiple feature maps, each feature map is a plane and the weights of all neurons on the plane are equal. In feature mapping structure, an activation function is used such as Relu and Sigmoid, which makes feature mapping displacement invariant. Each convolution layer in CNN is closely followed by a pooling layer to reduce the feature resolution. The most popular pooling methods are max pooling which keeps only the largest value within an area and average pooling which keeps the average value instead. It can consequently reduce training time and release overfitting. Next, a fully connected layer will flatten the learned vector and basic neural network can be applied.

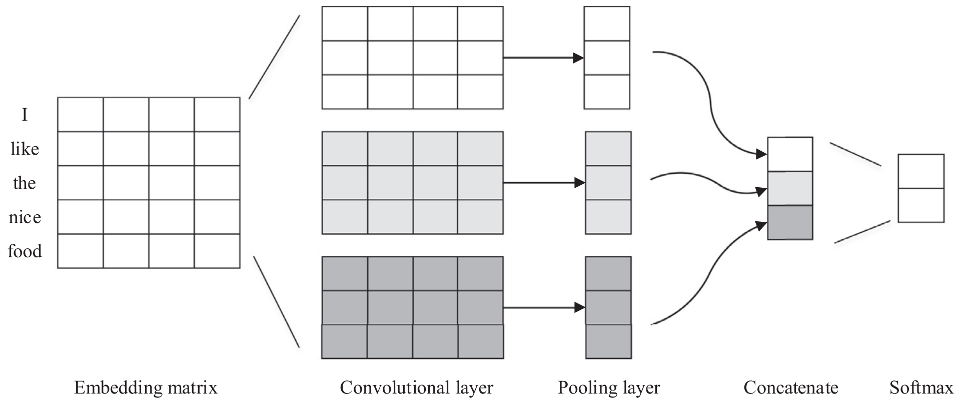


Fig. 4. CNN model for NLP.

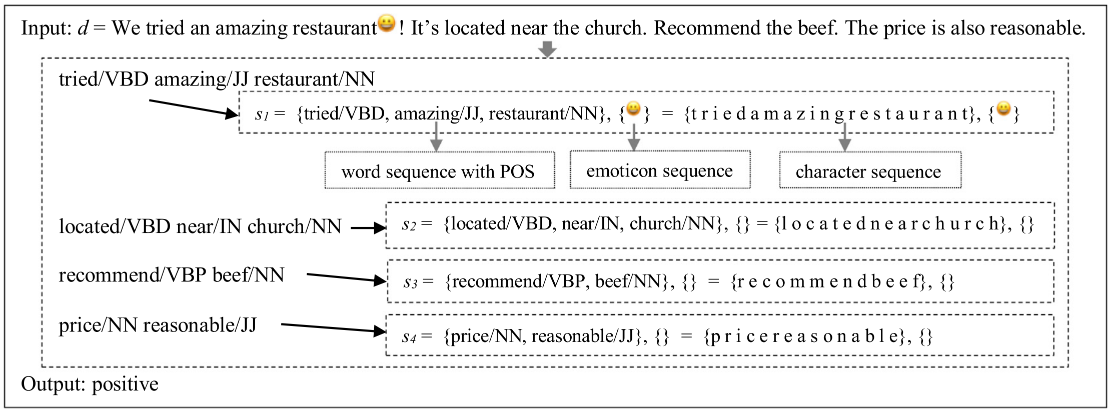

Input: d = We tried an amazing restaurant😀! It's located near the church. Recommend the beef. The price is also reasonable.
tried/VBD amazing/JJ restaurant/NN
s1 = {tried/VBD, amazing/JJ, restaurant/NN}, {😀} = {t r i e d a m a z i n g r e s t a u r a n t}, {😀}
word sequence with POS
emoticon sequence
character sequence
located/VBD near/IN church/NN
s2 = {located/VBD, near/IN, church/NN}, {} = {l o c a t e d n e a r c h u r c h}, {}
recommend/VBP beef/NN
s3 = {recommend/VBP, beef/NN}, {} = {r e c o m m e n d b e e f}, {}
price/NN reasonable/JJ
s4 = {price/NN, reasonable/JJ}, {} = {p r i c e r e a s o n a b l e}, {}
Output: positive

Fig. 5. Example of input and output in document sentiment classification.

3.3. Problem Definition

Our task is to predict the sentiment orientation given a document. We denote a document d in the corpus as a sequence of single sentences $\{s_1, s_2...s_k\}$ and k is the number of sentences it contains. The ith sentence in d, s_i, is further composed of a sequence of single words $\{w_1, w_2... w_m\}$ with length m and s_i can also be denoted as a sequence of characters $\{c_1, c_2... c_n\}$ with length n (usually $n>m$). The emoticon sequence $\{e_1, e_2...e_p\}$ with length p, if there exists, is separately extracted from s_i since emoticons are insensitive to the location and have no direct semantic association with the context words.

For document-level sentiment classification, the input is $d = \{s_1, s_2...s_k\}$ ($k\geq1$) where $s_i = \{w_1, w_2... w_m\}$ ($m\geq1$) and $s_i = \{c_1, c_2... c_n\}$ ($n\geq1$), if s_i contains emoticons, then $s_i = \{w_1, w_2... w_m\}, \{e_1, e_2...e_p\}$ ($m\geq1$, $p\geq1$) and $s_i = \{c_1, c_2... c_n\}, \{e_1, e_2...e_p\}$ ($n\geq1$, $p\geq1$). External resources include a sentiment lexicon and a negation lexicon. The output is a label of a certain sentiment type of d. According to specific need, output space = {subjective, objective} or {positive, negative, neutral}. Fig. 5 shows a simple example of document sentiment classification where input is an English document and output is a sentiment label. The document is split into k single sentences (after removing stopwords) and obviously k is 4. Each sentence s_i is represented by the word sequence with POS tags, character sequence and emoticon sequence. In s_1, we point out how each sequence is obtained from the sentence and obviously m is 3, n is 22, p is 1.

4. Sentence-level feature learning

Since our framework is a hierarchical structure from both words and characters to sentences and then to the document, in this section, we will first go through the first stage that learns the sentence-level representation from both the word sequence and the character sequence. The process of learning the document-level representation will be provided in the next section. Notice that the focus is not on how to design a more complex model, but on how to neatly utilize multiple features. As we can see in Fig. 6 (it takes CNN as a basic model), apart from the words themselves, we also take characters, word sentiment and negation, POS tags and emoticons into consideration. More specifically, for each sentiment word, we design different sentiment embeddings in different environments. After POS tagging, every word in the sentence is assigned with a POS tag. Each environment-specific word embedding is then concatenated with the POS tag embedding. In this case,

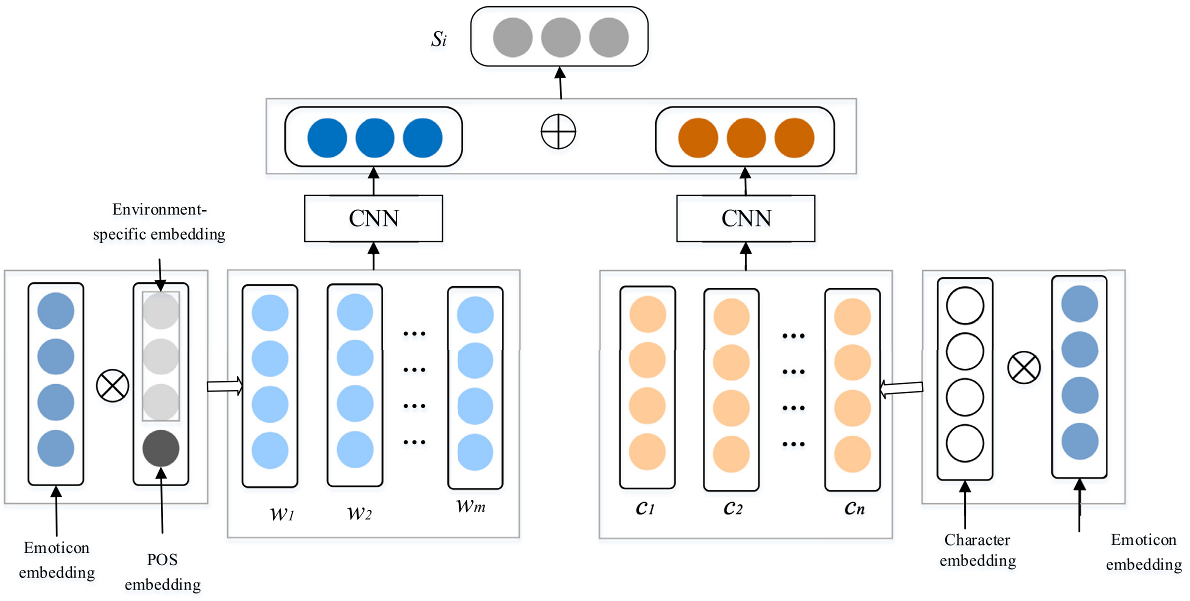


Fig. 6. Sentence-level feature learning.

two same words with different POS tags are represented by different embedding vectors. Since online texts often contain emoticons that are very likely to convey the author's sentiment attitude, we take emoticon information into consideration. We imitate the way in [34] to implement multiplication between the concatenation of word embedding and POS tag embedding and the average emoticon embedding of each sentence. Words can be further split into single characters which contain rich semantic information as well. So in addition to word embedding matrix, we design a character embedding matrix as another input matrix of the neural network. When it comes to RNN, characters are used to update the gates. We will demonstrate how to combine multiple features with a DNN model in each of the subsection below.

4.1. Environment-specific word embedding

In the sentiment classification task, sentiment words, many of which have been collected and sorted in a sentiment lexicon in previous research, are important indicators [46]. A sentiment lexicon and labeled texts must be prepared in advance. It is easy to satisfy due to existing sentiment lexicons of different languages, abundant online data and the aid of crawler. Sentiment words are special in sentiment analysis system in that they convey sentiment signals independently even if the context texts are vague. Negation words such as "not" or "rarely" usually change the original sentiment orientation of sentiment words (we call it a 'negation environment') and we must also consider negation information. For example, there is a simple movie review "The music is **not good**". Commonly the word "good" is a positive sentiment word, but the negation word "not" right before it makes this combination a negative expression. Based on the observation, we wish to leverage sentiment lexicon in a neural model. The sentiment lexicon we use in this work is the mixture of the lexicons provided by Liu et al [47] and Kong et al [48]. A feasible way is to modify the input word embedding based on whether it is a sentiment word and whether it is in a 'negation environment', which we will introduce later. Embedding considering the two factors are defined as 'Environment-specific Word Embedding' and we try to design two different special embedding vectors for each sentiment word. Unlike [32] we do not design different representations for negation words. Instead, negation words are used to check if the sentiment words are in a negation environment.

In order to get labeled texts, we collect large amount of texts online with a crawler tool – Scrapy.[1] Human labeling is expensive and time-consuming, so we assign a sentiment label in an automatic way. We make the assumption that if a post contains emoticon(s) and only positive emoticon(s), it is positive; if it contains emoticon(s) and only negative emoticon(s), it is negative. Of course we have to admit that the rule does not appeal to all microblogs and may have errors. However, it is a simple and easy-understanding strategy to realize automatic labeling. By this way, positive and negative texts are classified automatically and used to train embeddings for sentiment words. We use word2vec, a widely-used toolkit published by Google, to learn word embeddings from context. A corpus with only positive texts is used to train sentiment embeddings in positive environment and another corpus with only negative texts is used to train sentiment embeddings in negative environment.

[1] https://scrapy-chs.readthedocs.io/zh_CN/0.24/.

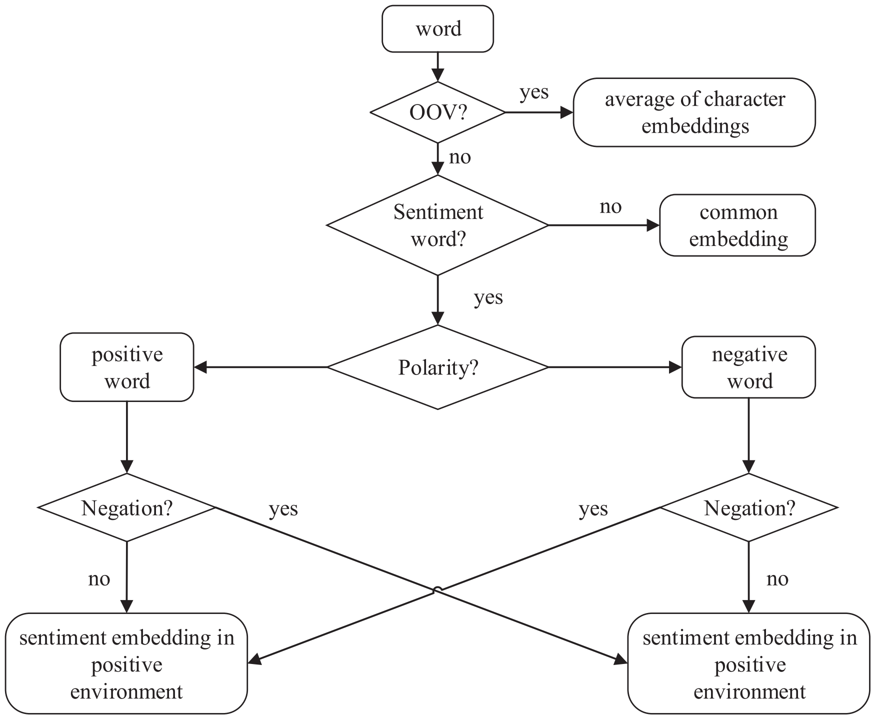

word
OOV?
yes
average of character embeddings
no
Sentiment word?
no
common embedding
yes
Polarity?
positive word
negative word
Negation?
yes
yes
Negation?
no
no
sentiment embedding in positive environment
sentiment embedding in positive environment

Fig. 7. Choice of sentiment word embedding.

As we explained above, negation words around sentiment words can reverse the original polarity, so it is necessary to take negation words into consideration. Since negation words often appear right before sentiment words, we set a window size and if there exist negation word(s) before the sentiment word within the size, we think it is in a negation environment and the sentiment orientation will be reversed. Besides, the number of negation words here should be an odd number so that double negatives can be avoided. The concept of negation environment is actually a simple implementation of polarity shift detection. Since we do not focus on this task, our policy tends to appeal to common phenomena and more robust implementation is present in [49], which could be helpful in our future work.

Which embedding to choose for each word when learning representation for a new sentence is shown in Fig. 7. We firstly check if the word is included in our pre-trained vocabulary. If not, we use the average of embeddings of characters in the word as the word embedding. Embedding of characters will be introduced in part D. Otherwise, we check if it is a sentiment word by referring to the sentiment lexicon. For common words, we simply choose the embedding trained with unlabeled texts by word2vec. For a positive word, if it is in a negation environment, we choose its sentiment embedding in negative environment, otherwise its sentiment embedding in positive embedding. The policy for negative words is similar.

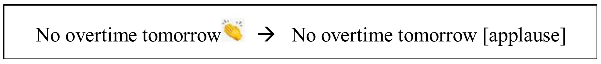
4.2. POS Tag

Words of different POS tags may make different contributions to the sentence semantics. Take the sentence with word POS tag labeled as an example. “This_DT movie_NN is_VBZ a_AN success_NN.” The noun “success” directly conveys the positive attitude towards the object “movie”, while the determiner “this”, the verb “is” (but verbs are often important) and the article “a” each has very little information and is not able to help guide sentiment classification of the sentence. Also, words have the same structure or spelling but with different POS tags have different semantic meanings and should be treated differently.

Different strategies to combine POS tags with Recursive Neural Network are explored in [14]. However, we just take a simple measure and concatenate an environment-specific embedding with its POS tag embedding to adapt to more general networks. The representation of the tth word w_t attached with a POS tag, $\boldsymbol{e_t} \in \mathrm{R}^{dw+dp}$ is further defined as:

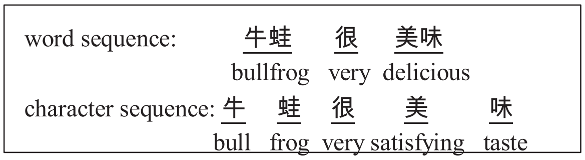
$$e_t = v_t \oplus pos_t \tag{3}$$

where $\boldsymbol{v_t} \in \mathrm{R}^{dw}$ is the embedding by referring to the lookup table and d_w is the dimension of word embedding, $\boldsymbol{pos_t} \in \mathrm{R}^{dp}$ is the pre-trained embedding of the POS tag of w_t and d_p is the dimension of POS embedding. $\oplus$ stands for the concatenation of two vectors.



No overtime tomorrow → No overtime tomorrow [applause]

Fig. 8. Example of emoticon in a microblog shown in crawled form.



word sequence:	牛蛙		很	美味	
	bullfrog		very	delicious	
character sequence:	牛	蛙	很	美	味
	bull	frog	very	satisfying	taste

Fig. 9. Example of Chinese character sequence.

4.3. Emoticon embedding

Nowadays, various emoticons are provided in different online platforms and are popular among users since they can vividly convey emotions in the form of images, usually facial expression. For example, emoticons such as and clearly contain positive emotion while , and so on usually come with negative emotion. Even if someone publishes a tweet only with such emoticon(s) and without any word, we still can infer the rough sentiment orientation. Compared to words, the usage of emoticons is more flexible in that they can appear in any position of the sentence, no matter the beginning, the middle or the end as long as they do not split up a complete word or phrase. They are also insensitive to the order. What's more, the removal of emoticons usually won't make a sentence incomplete. As a result, we should treat emoticons in a different way from words. Notice that a crawled emoticon is not presented in the form of image but text, namely the name of the emoticon between a pair of square brackets. Below shows a crawled tweet with a positive emoticon, as is shown in Fig. 8.

As a result, we should treat an emoticon as a special "word". We manually collect typical emoticons that are frequently used and can clearly show sentiment orientation in advance. Controversial annotations are fixed by majority voting. Then for every sentence, we can use regular expressions to find out typical emoticons contained. Emoticons that fail to show obvious orientation are simply ignored. Average of embedding vector of each typical emoticon is calculated as emoticon embedding of this sentence and is used to multiply with $\boldsymbol{e_t}$ obtained in the previous step. Affected by emoticons occurring in the sentence, embedding of w_t is then represented as:

$$e_t = e_t \odot \frac{1}{p}\sum_{j\le p} emo_j \tag{4}$$

where $\boldsymbol{emo_j} \in \mathrm{R}^{de}$ is the embedding vector of the jth typical emoticon in the sentence and d_e is the dimension, p is the number of emoticon(s) in the sentence. $\boldsymbol{a} \odot \boldsymbol{b}$ stands for the element-wise multiplication of vector $\boldsymbol{a}$ and $\boldsymbol{b}$.

If a sentence contains no typical emoticon, this step is skipped. After that, we get final input word embedding matrix, which is sent to a neural network for further progressing. That is, s_i is represented as:

$$\begin{matrix} fw_i = \mathrm{GRU}(e_1,\ e_2 \ldots e_m;\ emo_1,\ emo_2 \ldots\ emo_p) \\ fw_i = \mathrm{CNN}(e_1,\ e_2 \ldots e_m;\ emo_1,\ emo_2 \ldots\ emo_p) \end{matrix} \quad \text{or} \tag{5}$$

4.4. Character embedding

A word, no matter from English or Chinese or any other language, is often composed of more than one character. Some characters have a clear meaning while the others are meaningless unless that they combine into words. For example, there is part of a Chinese review for a restaurant "牛蛙/很/美味" (The bullfrog is very delicious). In Fig. 9, **错误!未找到引用源.**, we list both the word sequence and character sequence of the review together with the English translation. Clearly, the single characters also contain rich semantic information, which can help improve representation of the whole sentence.

Therefore, if the basic model is CNN, given a text, we treat it as a sequence of single characters. Processing on the character sequence is the same as that what we do on the word sequence. We train embedding for each character using an unlabeled corpus and the tool word2vec. Just like the word embedding matrix, the character embedding matrix is firstly multiplied with the emoticon matrix and then fed into a neural network. The learned feature vector from the character sequence with length n is defined as:

$$fc_i = \mathrm{CNN}(c_1,\ c_2 \ldots c_n;\ emo_1,\ emo_2 \ldots\ emo_p) \tag{6}$$

Finally, the embedding vector of s_i, is represented as $\boldsymbol{f_i}$, the concatenation of $\boldsymbol{fw_i}$ and $\boldsymbol{fc_i}$, as we can see in the left part of Fig. 10:

$$f_i = fw_i \oplus fc_i \tag{7}$$

As we define in 3, $\boldsymbol{fw_i}\oplus\boldsymbol{fc_i}$ stands for the concatenation of vector $\boldsymbol{fw_i}$ and $\boldsymbol{fc_i}$ as the final sentence representation.

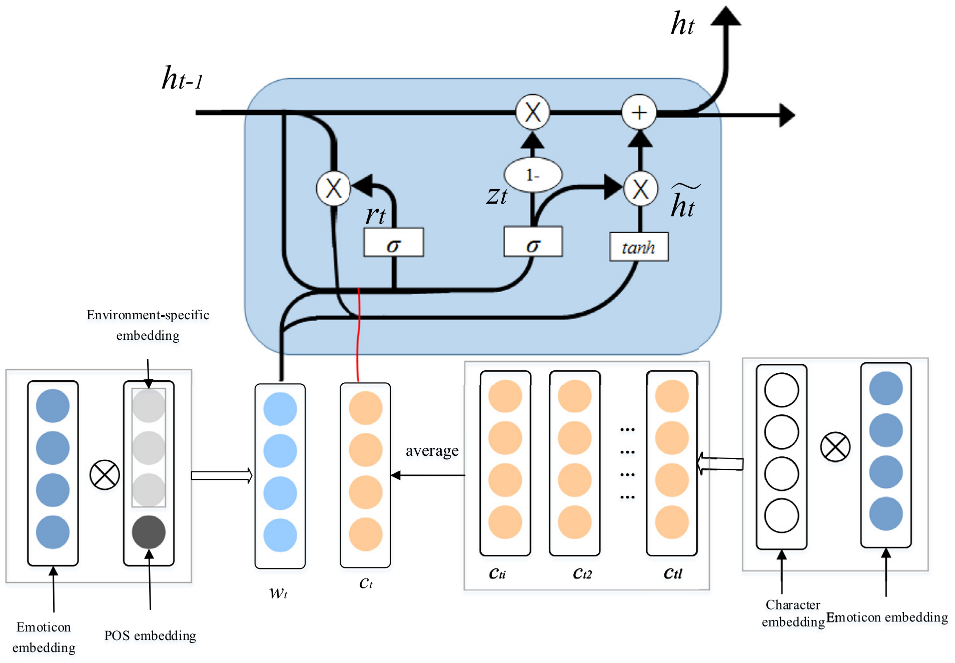


Fig. 10. Updated GRU cell with character information.

On the other hand, if the basic model is RNN (we choose GRU in our work), we use character(s) of the word as additional knowledge to enhance the model. At each time step t, we use the average character embedding of w_t to update the reset gate and update gate, which is illustrated in **错误!未找到引用源.** Compared to traditional GRU, formulas related to $\boldsymbol{z_t}$ and $\boldsymbol{r_t}$ have some changes. Suppose w_t contains l characters, average of embedding of $\boldsymbol{c_{t,1}}...\boldsymbol{c_{t,l}}$ is denoted as $\boldsymbol{c_t}$, which is input into the GRU cell for updating $\boldsymbol{z_t}$ and $\boldsymbol{r_t}$. At time step t, generation of $\boldsymbol{h_t}$ is shown in 8. The new data flow is highlighted in red in **错误!未找到引用源.**, so do the changes in 8. In this case, there is no separate feature vector from the character sequence based on GRU, but instead $\boldsymbol{f_i}$ equals $\boldsymbol{f_{wi}}$, which is represented with an updated GRU model.

$$\begin{aligned} z_t &= \sigma(W_z[e_t, h_{t-1},\ c_t] + b_z) \\ r_t &= \sigma(W_r[e_t, h_{t-1},\ c_t] + b_r) \\ \widetilde{h_t} &= \tanh(W_h[e_t,\ r_t * h_{t-1}] +\ b_h) \\ h_t &= (1 - z_t) * h_{t-1} + z_t *\ \widetilde{h_t} \end{aligned} \tag{8}$$

If the text has only one single sentence, the sentence-level feature is actually the document-level feature. Otherwise, we further apply a hierarchical structure and try to learn the document-level feature based on the sequence of sentence vectors.

5. Document-level feature learning

As mentioned above, if the text consists of more than one sentence, we further utilize a hierarchical structure to get document feature from the sequence of sentences. In addition, a man-made rule that pays more attention to important parts in a document is applied to enhance the performance. Suppose the document is made up of k ($k \geq 2$) single sentences, the process taking CNN-based model as an example is present in Fig. 11 and RNN-based model also follows this way. If the document only contains one sentence, document representation equals sentence representation and this step is skipped.

That is, each sentence in a document is treated the same way as we treat each word in a sentence. We construct a document representation by firstly building representations of sentences from word/character representations and then aggregating sentence representations in the same way in the previous step into a document representation. For example, if the basic model is RNN, when dealing with the sequence of sentences, the state of ith sentence, $\boldsymbol{h_i}$, will be calculated by gates with both representation of current sentence $\boldsymbol{s_i}$ and the previous hidden state $\boldsymbol{h_{i\text{-}1}}$ of (i-1)th sentence. Then we get a NN-based document embedding vector.

Besides, according to the writing habit of many people, the first or the last sentence usually contains the conclusive information and contributes more [50], which motivates us to give more weight to the first and the last sentences in a document to better catch the core information. Also, since sentiment words and emoticons are important indicators of sentiment orientation as we mentioned in Section IV, we assume that the more sentiment words and typical emoticons a

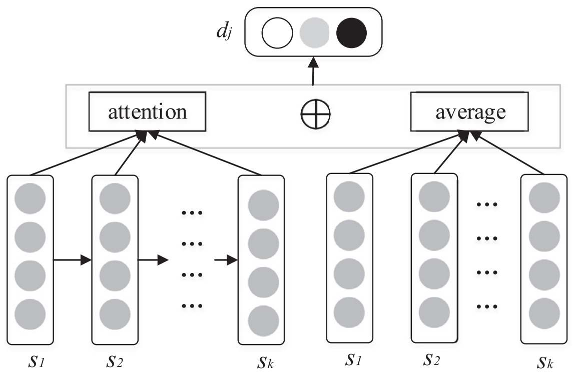


Fig. 11. Document-level feature learning.

sentence contains, the more important the sentence is in the document representation. Concretely, we define the rule-based document representation as:

$$\begin{aligned} &\mathrm{AVE}(f_1,\ f_2,\dots,f_k) = \sum_{0\le i\le k} \beta_i f_i \\ &\beta_i = \mathrm{softmax}(v_i) \\ &v_i = \frac{q}{k} * \begin{cases} 2 & i = 0 \text{ or } k \\ 1 & otherwise \end{cases},\ q = |sentiment\ words| + |typical\ emoticon| + 1 \end{aligned} \tag{9}$$

where AVE() stands for the weighted average of sentence representations, β_i stands for the weight of ith sentence according to our man-made rule. We firstly assign a weight to each sentence according to the total number of sentiment words and typical emoticons in the sentence. For the first and the last sentence, their weights are doubled since any of them may be the conclusion sentence. Then we add a softmax operation over the weights of sentences in the document so that their sum equals 1.

Next, the NN-based embedding vector and the rule-based embedding vector are concatenated:

$$\begin{aligned} f_d &= \mathrm{GRU}(f_1, f_2 \dots f_k) \oplus \mathrm{AVE}(f_1, f_2 \dots f_k) \\ f_d &= \mathrm{CNN}(f_1, f_2 \dots f_k) \oplus \mathrm{AVE}(f_1, f_2 \dots f_k) \end{aligned} \quad \text{or} \tag{10}$$

The document-level feature vector is then fed into a classifier to compute the probability distribution of the sentiment label of the whole document. The classifier i designed as a two-layer feed-forward network with RELU as the activation function followed by a softmax function so that the probability values add up to 1. The label with the highest probability is the predicted sentiment label. In neural networks, the loss function for optimization is defined as negative cross-entropy between the gold sentiment label and the proposed sentiment label at training:

$$L = -\sum_{i=1}^{N} l_i * \log(y_i) + R \tag{11}$$

where l_i is the ground truth label of d_i and y_i is the estimated label computed by our model. N is the number of documents in the dataset. $R = \lambda||\psi||_2^2$ is the L2 regularization and ψ contains all the parameters of the model.

6. Experimental evaluation

In this section we will present the performance of our method on sentiment classification task compared with several existing methods. We will implement two kinds of sentiment classification, including: 1) subjective/objective classification that predicts whether a document contains any emotion and 2) positive/negative sentiment classification that predicts its sentiment orientation. We call the first type of classification "sub/obj classification" and the latter type "pos/neg classification" later in this paper.

6.1. Datasets and settings

We use two datasets to conduct evaluation to prove the usability of our method. They are offered by microblog sentiment classification/ emotion analysis task in NLPCC2013[2] and NLPCC2014[3] (SC2013 and SC2014 for short). Texts are mainly

[2] http://tcci.ccf.org.cn/conference/2013/.
[3] http://tcci.ccf.org.cn/conference/2014/.

```
<weibo id=”203” emotion-type=”happiness”>
  <sentence id=”1”>教室终于安空调了! </sentence> (The classroom is finally air-conditioned!)
  <sentence id=”2”>我们终于得到救赎了❤ ! </sentence> (We are finally saved ❤ !)
</weibo>
```

Fig. 12. An example in Weibo.

Table 1
Polarity of emotion labels in the microblog datasets.

Polarity		Chinese	English
Objective		无	none
subjective	positive	高兴 喜好	happiness like
	negative	厌恶 愤怒 恐惧 悲伤	sadness anger disgust fear
	neutral	惊讶	surprise

Table 2
Statistics of datasets.

Dataset		Training set	Test set
SC2013-sub/obj	Subjective	2172	955
	Objective	1828	1158
SC2014-sub/obj	Subjective	5416	698
	Objective	6591	32
SC2013-pos/neg	Positive	965	450
	Negative	1094	480
SC2014-pos/neg	Positive	2700	405
	Negative	2429	293

Chinese and are crawled from Weibo, a popular social software in China. Texts in both of the two datasets may contain some emoticons and documents have been split into single sentences in advance and it is easy to automatically realize by checking the punctuation. Each document is assigned with an emotion label so that we can decide whether it is subjective or objective, positive or negative. An example in present in Fig. 12 and we translate the texts into English. Sometimes emotion labels are given in English while sometimes in Chinese, so we sort the emotion labels and clarify the corresponding polarity as is shown in Table 1.

Since we find the number of sentences with a neutral sentiment (“surprise”) is far too small compared with the number of positive or negative sentences, which will make the network difficult to train, we do not consider neutral sentences. We also delete some of the data in the training set to balance the distribution while the test set is untouched. Statistics of SC2013 and SC2014 for different classification purposes are shown in Table 2. As we can see from Section IV, features used in our method are also common in many other systems (including shopping websites, news websites and so on) and languages. Although we conduct experiments on microblog datasets, our proposed method is also applicable to other applications.

For both common words and sentiment words, d_w is set to 100. Initial embeddings of the most frequent 50 POS tags are sampled from a uniform distribution with range (-0.5, 0.5). Embedding of other infrequent POS tags uses zero initialization. d_p is set to 20 and according to the research of Huang et al [14], d_p almost does not affect the performance. The dimension of character embedding is set to 120, the same as the addition of d_w and d_p. We set d_e to 120 since the emoticon embedding is to multiply with 1) the concatenation of word embedding and POS tag embedding; 2) the character embedding. Emoticon embedding vectors are initialized by sampling from a normal distribution $N(0, 0.5)$.

In every experiment, we shuffle the training set and 10% is further split as a validation set to adjust the hyper-parameters of the model. As we try to fix a classification task, we use negative cross-entropy as the loss function. Also, we adopt RMSProp as the optimizer with an initial learning rate of 0.001, a minibatch size from 10 to 30 according to the size of the training data and dropout regularization with probability 0.5 and 0.4 for MF_CNN and MF_RNN respectively to avoid overfitting. Size of kernel is set to 32 in MF_CNN. According to the length of texts, we set the maximum length of each sentence to 100 and 200 for words and characters respectively which means the first 100 words and 200 characters in the sentence are considered; we set the maximum length of each document to 10, which means the first 10 sentences in the microblog is considered. Similarly, maximum length of each sentence for emoticons is set to 2. Epochs in training is set to 20.

6.2. Baselines

Our model that considers features mentioned in Section IV based on a simple CNN model is named **M**ultiple-**F**eature CNN (MF_CNN) and similarly, our model based on a simple RNN model is named **M**ultiple-**F**eature RNN (MF_RNN). In order to prove the value of our work, we compare our method with a variety of existing methods, including:

Table 3
Sub/obj and Pos/neg classification results.

Method	SC2013-sub/obj		SC2014-sub/obj		SC2013-pos/neg		SC2014-pos/neg	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
Majority	0.5430		0.5489		0.5313		0.5264	
Count_senti	0.4974	0.4958	0.6041	0.3766	0.7043	0.7008	0.8097	0.8118
Ave_SVM	0.6509	0.6220	0.8832	0.8655	0.6935	0.6522	0.8928	0.8926
CNN	0.6753	0.6750	0.9054	0.8752	0.7495	0.7490	0.9270	0.9298
CNN+emo	0.7031	0.7090	0.9102	0.9017	0.7720	0.7690	0.9552	0.9508
3W-CNN	0.6921	0.6905	0.9173	0.9112	0.7622	0.7523	0.9490	0.9402
MF_CNN	0.7480	0.7235	0.9754	0.9649	0.8161	0.7880	0.9802	0.9672
biGRU	0.6802	0.6801	0.9167	0.9114	0.7516	0.7512	0.9427	0.9411
biGRU+att	0.6966	0.6962	0.9221	0.9127	0.7634	0.7603	0.9570	0.9559
Transformer	0.7045	0.6820	0.9452	0.9086	0.7594	0.7584	0.9485	0.9372
Bert	0.7647	0.7517	0.9861	0.9550	0.8096	0.7919	0.9788	0.9659
MF_RNN	0.7335	0.7314	0.9958	0.9748	0.8537	0.8534	0.9943	0.9941

Majority: It is a basic baseline approach also used in [51], which simply assigns the majority label in the training set to each sample in the test set. Since the classification tasks in our evaluation are all binary classification, we check which label is more in the training set (suppose the label is 1) and every sample in the test set is labeled as 1.

Count_senti: It is an unsupervised method which relies on a high-quality sentiment lexicon. In some previous work, if the text contains more positive words than negative words, it is labeled as “positive”, otherwise “negative”. We imitate this tack and define that if a text contains sentiment word(s), no matter which type, it is “subjective”, otherwise “objective”.

Ave_SVM: The representation of the text is defined as the average of the embedding vector of each word in the text. Then the representation vector is put to a SVM classifier (default setting in sklearn[4] except that C=2) for prediction.

CNN: In our experiments, we try the common CNN model and TextCNN proposed by [43] and we find that the later one is better for our task. So we use TextCNN both for comparison and as the basic of MF_CNN. Sizes for three filters are set to 5, 10 and 15 respectively. Dropout rate is set to 0.5. Note that we also implement the model in a hierarchical structure.

CNN+emo: We implement the method proposed in [34], which adopts the multiplication of word embedding matrix and emoticon embedding matrix as the input of a textCNN model to realize hierarchical feature learning.

3W-CNN: We implement the method in [27]. Firstly, a confidence divider and a confidence function is designed to distinguish the classification quality of CNN. Secondly, NB-SVM is used to reclassify the predictions with weak confidence.

biGRU: Since GRU and LSTM are both effective variants of RNN and GRU has fewer parameters, we implement GRU model in our evaluation. In order to catch information both backwards and forwards, we adopt biGRU and dropout rate is 0.4. For the sake of fairness, we choose biGRU as the basic of MF_RNN.

biGRU+att: We apply self-attention on biGRU to improve the performance. Also, MF_RNN adopts attention mechanism.

Transformer: Transformer is a model consisting of an encoder-decoder architecture [52], which is based solely on attention mechanisms. Although it is initially proposed to conduct neural translation, we can use the decoder part for feature extraction to conduct text classification. In training, we change minibatch size and maximum length to the same value as our system.

Bert: Bidirectional Encoder Representation from Transformer (Bert) is a language representation model published by Google [53]. It pre-trains representations by jointly conditioning on both left and right context in all layers. It can be further fine-tuned with an additional output layer to conduct different tasks. We choose its variant that supports Chinese and also change minibatch size and maximum length.

We would like to stress that a multiple features-enhanced model is ought to be compared with the corresponding basic model. That is, for example, if MF_CNN outperforms CNN and its variant(s), but is not as good as RNN variants, it does not deny our work because the point is the promotion that multiple features bring.

6.3. Classification result

Purpose of document sentiment classification is to predict if the document is 1) for sub/obj classification, subjective or objective; 2) for pos/neg classification, positive or negative. Accuracy (Acc) and Macro-averaged F-measure (Macro-F1) are calculated as the metrics to measure the result of classification. The higher Accuracy and Macro-F1 are, the better the performance is.

Results of subjectivity classification conducted on SC2013 and SC2014 are shown in the left part of Table 3. Statistics of pos/neg classification are presented in the right part of Table 3. Accuracy of Majority is around 0.5 in every experiment

[4] https://scikit-learn.org/stable/.

Table 4
Feature Ablation results based on CNN model.

Method	SC2013-sub/obj		SC2014-sub/obj		SC2013-pos/neg		SC2014-pos/neg	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
MF_CNN	0.7480	0.7235	0.9754	0.9649	0.8161	0.8080	0.9802	0.9672
MF_CNN-senti	0.7040	0.7036	0.9534	0.9521	0.7839	0.7603	0.9723	0.9618
MF_CNN-POS	0.7350	0.7205	0.9681	0.9595	0.8116	0.8056	0.9703	0.9580
MF_CNN-emo	0.7313	0.7210	0.9510	0.9467	0.8038	0.7923	0.9671	0.9575
MF_CNN-char	0.7175	0.7136	0.9655	0.9651	0.7759	0.7766	0.9580	0.9290
MF_CNN-hier	0.7018	0.7003	0.9690	0.9625	0.7461	0.7304	0.9743	0.9741
MF_CNN-rule	0.7593	0.7590	0.9475	0.9450	0.7838	0.7830	0.9862	0.9860

because there exists no obvious data skew in the training set. Also, we notice that except Majority, all methods have much better performance on SC2014 than SC2013. Maybe the texts in SC2014 are more properly written and easier to understand for the computer, while Majority is insensitive to this. Count_senti does not require labeled training data, but the performance is not good. By reading microblogs, we find when people express a personal opinion, they sometimes do not use obvious sentiment words. For example, we human can see that '天天都有这种新闻!' (There is this kind of news every day!) expresses bore or anger, but the computer cannot understand. It achieves fair results in pos/neg classification task, demonstrating that people tend to use sentiment words to express concrete emotions, but the performance is still worse than machine learning-based methods. Ave_SVM achieves better results than Count_senti, but still worse than other neural network-based methods and stability of this method cannot be guaranteed, no matter how large the dataset is. Classification results based on deep neural networks are commonly better, demonstrating the powerful learning ability of DL. For example, in sub/obj classification, they can achieve at least 66% (biGRU) accuracy on SC2013 and 90% (CNN) accuracy on SC2014. Compared to CNN, He's method [34] can improve the classification result by incorporating emoticon information. With an enhance model to reclassify samples with low confidence, 3W-CNN has an advantage over CNN in all the datasets. 3W-CNN performs worse than CNN+emo in three datasets and that further proves tasks are closely related to specific features. Attention mechanism is also proven effective in our task, bringing up to 1.64% improvement in Accuracy and 1.61% improvement in Macro-F1 when applied on SC2013 in sub/obj classification. Performance of Transformer is comparable to that of biGRU+att. Utilizing Masked LM for bidirectional language modeling, Bert successfully beats the baselines. In particular, it achieves 76.47% accuracy and 75.17% Macro-F1 on SC2013 in sub/obj classification, even better than MF_CNN and MF_RNN (but still inferior to MF_RNN on other datasets). We guess that is because after the removal of objective texts, some documents in our subjective datasets are not long and Bert does not go well for short texts, it fails to make the most of it in pos/neg task.

As we can see, after applying multiple features, performances of CNN and GRU models both get improved. In sub/obj classification task, considering features mentioned in our work, MF_CNN obtains 4.49 points in Accuracy and 1.45 points in Macro-F1 high than CNN+emo, the best CNN-based method for comparison (when applied on SC2013). On the other hand, combining multiple features with RNN yields 7.37% promotion in Accuracy and 6.21% promotion in Macro-F1 at most compared to biGRU+att, the best RNN-based method (when applied on SC2014). Multiple features we utilize also work in pos/neg classification. For CNN model, features mentioned in our work render 4.41% promotion in Accuracy and 1.90% promotion in Macro-F1 at most compared to CNN+emo. And for RNN model, MF_RNN outperforms biGRU+att by 9.03 points in Accuracy and 8.71 points in Macro-F1 at most. MF_RNN achieves the best classification performance in three out of four tasks. Also, GRU performs better than CNN on our datasets and maybe that is because RNN can capture sequence information in texts, which is very important in sentiment classification task. The results demonstrate the advantage of the combination of different features we use in this work, no matter the basic model is CNN or RNN. To what extent each feature contributes is further discussed in the next section.

6.4. Feature ablation

In order to prove the contributions of the features mentioned in our work, we try to drop a part every time and conduct both sentence-level subjectivity and sentiment classification with the same settings to see whether the feature really improves the performance and to what extent.

Sentiment words: Firstly, we ignore the particularity of sentiment words and negation words. All word embedding vectors are trained from unlabeled corpus and treated equally. The incomplete edition of MF_CNN and MF_RNN is referred to as MF_CNN-senti and MF_CNN-senti respectively. We realize classification tasks utilizing MF_CNN-senti and MF_RNN-senti on SC2013 and SC2014 and the results compared to the complete version are shown in Tables 4 and 5 respectively. As we can see, if no more attention is paid to word sentiment, classification performance will be damaged. By comparison, we can notice that sentiment and negation information promotes 4.76% in Accuracy and 2.99% in Macro-F1 at most in sub/pos classification. It also promotes 6.54% in Accuracy and 6.52% in Macro-F1 at most in pos/neg classification. The results clearly prove that sentiment words are of great importance in conveying sentiment, especially concrete sentiment attitudes. Usage of most sentiment words is straightforward.

Table 5
Feature Ablation results based on RNN model.

Method	SC2013-sub/obj		SC2014-sub/obj		SC2013-pos/neg		SC2014-pos/neg	
	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1	Acc	Macro-F1
MF_RNN	0.7335	0.7314	0.9958	0.9848	0.8537	0.8534	0.9943	0.9941
MF_RNN-senti	0.7113	0.6702	0.9825	0.9772	0.7883	0.7882	0.9905	0.9901
MF_RNN-POS	0.7275	0.7250	0.9894	0.9784	0.8402	0.8342	0.9926	0.9935
MF_RNN-emo	0.7198	0.7126	0.9914	0.9825	0.8350	0.8312	0.9886	0.9853
MF_RNN-char	0.7018	0.7018	0.9662	0.9658	0.7888	0.7873	0.9802	0.9672
MF_RNN-hier	0.7052	0.7049	0.9925	0.9705	0.7763	0.7745	0.9771	0.9705
MF_RNN-rule	0.7300	0.7297	0.9845	0.9694	0.8463	0.8452	0.9971	0.9965

POS tag: If we drop POS information, then sentiment or common word embedding does not need to concatenate with a POS tag embedding. Notice that in this case, the dimension of both emoticon embedding and character embedding is set to 100. Performances of the incomplete models, MF_CNN-POS and MF_RNN-POS, are listed in Tables 4 and 5 respectively. The results show that by considering POS tag information, we can get 1.30% promotion in Accuracy and 0.64% promotion in Macro-F1 at most in sub/obj classification. Promotion in pos/neg classification at most is 2.35% in Accuracy and 3.22% in Macro-F1 respectively. The improvement is not obvious, partly because of the quality of POS tagging. It may relate to the initialization method too.

Emoticon: In this part, we just ignore emoticons in the text. The simplified models without emoticon embedding matrix in this part are referred to as MF_CNN-emo and MF_RNN-emo. From the data in Table 4, on SC2013 and SC2014, when emoticon embedding takes part in sentence representation, 1.67% promotion in Accuracy and 1.18% promotion in Macro-F1 at most in sub/obj classification are achieved. Emoticon information also brings 1.35% promotion in Accuracy and 1.57% promotion in Macro-F1 at most in pos/neg classification as we can see in Table 5. The improvement is more obvious than that of POS tag and if we collect more typical emoticons and sort them by sentiment orientation in advance, we may catch more useful information and the promotion may be more obvious. We can also consider more sentiment-related symbols.

Character: When we drop character information, the right part in Fig. 6 is removed in this situation and only the sequence of words is decoded. According to the statistics, in pos/neg classification, considering character information yields 3.17% in Accuracy and 2.96% in Macro-F1 at most, while in pos/neg classification, 6.49% in Accuracy and 6.61% in Macro-F1 information in Chinese language. For languages like Chinese, apparently we cannot guarantee the result of word segmentation is absolutely right. The mistakes in word segmentation will unavoidably have a bad effect on word sequence feature extraction and character sequence may make up for this to some extent. Also, as we have stated before, character embeddings can be used to represent words that do not appear in the pre-trained vocabulary.

Since we treat a document as a sequence of single sentences, apart from features considered at sentence level, we also conduct experiments on a higher level to check the contributions of the hierarchical structure (although actually it is not a feature, it is still a different way to handle long texts) and the man-made rule.

Man-made rule: We name the methods that ignore the man-made rule MF_CNN-rule and MF_RNN-rule respectively. The classification results present in Tables 4 and 5 show that by applying the rule, 1.13% promotion in Accuracy and 1.54% promotion in Macro-F1 at most in sub/obj classification are achieved. It renders 2.47% promotion in Accuracy and 1.68% promotion in Macro-F1 at most in pos/neg classification. But overall, the improvements brought by the man-made rule in a document are not obvious. In fact, in some case (for example, in sub/obj classification conducted on SC2013), it even renders negative effect (this result in fact proves the effectiveness of a hierarchical structure of document feature learning). The bad result is partly because that our texts from the social network are not structured and formal enough to contain so many conclusion sentences. As a result, only using the rule cannot catch the key point of the document. If the rule is applied to other copra, it may achieve better results.

Hierarchical structure: The enhanced models that drop the hierarchical structure by only averaging of the embedding vectors of sentences in the document (different weights are applied) are referred to as MF_CNN-hier and MF_RNN-hier respectively, when we use CNN and RNN as the basic model. It produces 4.62% promotion in Accuracy and 2.65% promotion in Macro-F1 at most in sub/obj classification; 7.00% promotion in Accuracy and 7.76% promotion in Macro-F1 at most in pos/neg classification. Compared to man-made rule, putting the sentence representation sequence into a neural network is less sensitive to the quality of texts. For example, the attention mechanism can learn the weights of different sentences automatically. Decoding single sentences first and then learning document representation with sentence representation can release loss of information as the text gets much shorter after being split into single sentences. The drawback of a hierarchical structure is the relatively long running time compared to the corresponding basic model.

More details of feature ablation experiments are presented in Tables 4 and 5.

Overall, the features mentioned in this paper all contribute to the improvement of text sentiment classification to some extent. In comparison, the semantics of sentiment words is the most remarkable. Maybe it is because in social networking people tend to use frank expressions to show their sentiment orientation straightforward. It further demonstrates that NLP tasks need not only a complex machine learning model, but also understanding of the specific task. For sentiment classification, an important point is sentiment words. Statistics prove that treating a document as a sequence of sentences to further employ a CNN or RNN model and sometimes paying extra attention to important sentences (the first and the last sentence, sentences containing sentiment words and typical emoticons) gets the best classification results for formal documents. For informal texts, it may be better to just employ a hierarchical structure.

7. Conclusion and future work

In this paper, we realize text sentiment classification by hierarchically combining multiple features with different deep neural network models. We devise different representations for sentiment words in different environments and leverage POS information to improve original word embedding. If the text contains emoticon, emoticon matrix is also applied to enhance the input word representations. In addition, representations for characters are learned to catch more syntactic information. A hierarchical structure and man-made rule are employed together to further get document-level representation. As these features can also be caught in other systems, our method has flexibility to be used in other applications. Also, although machine learning is rather effective, rules are still important in many other NLP tasks. Experimental results on public datasets demonstrate that the features do improve the classification performance and our method effectively outperforms existing methods.

Since tweets online are published by different users, it will be meaningful to consider user information in our future work. For example, history information can reflect the user' bias; identity of the user may have an effect on how the user thinks (like how the user sees a current affair); if a user follows and is followed by another user, they two may have similar opinion towards the same target. We believe user information will help enhance performance of sentiment classification (also can be used in other classification scenarios like review spam detection [54] and user identity detection [55] et al). In addition, given time information of the text, we can extend the original task to a time-evolving sentiment classification task so that we can see temporal dynamic [56] change of a certain topic. What's more, we would like to extend our method to other similar tasks, for example, for different tasks, we can also adopt task-specific word embedding.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Li Kong: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing - original draft, Writing - review & editing, Formal analysis. **Chuanyi Li:** Conceptualization, Methodology, Resources, Writing - review & editing, Supervision, Project administration, Formal analysis, Funding acquisition. **Jidong Ge:** Funding acquisition, Writing - review & editing. **FeiFei Zhang:** Writing - review & editing. **Yi Feng:** Writing - review & editing. **Zhongjin Li:** Writing - review & editing. **Bin Luo:** Writing - review & editing.

Acknowledgment

This work was supported by the National Key R&D Program of China (2016YFC0800803), the National Natural Science Foundation of China (No. 61802167, 61802095), and the Fundamental Research Funds for the Central Universities.

References

[1] E. Cambria, D. Das, S. Bandyopadhyay, et al., A Practical Guide to Sentiment Analysis, Springer, Cham, Switzerland, 2017.
[2] J. Bertot, P. Jaegrt, D. Hansen, The impact of polices on government social media usage: issues, challenges, and recommendations, Gov. Inf. Q. 29 (1) (2012) 30–40.
[3] D. Kang, Y. Park, Review-based measurement of customer satisfaction in mobile service: sentiment analysis and VIKOR approach, Expert Syst. Appl. 41 (4) (2014) 1041–1050.
[4] X. Yang, G. Yang, J. Wu, Integrating rich and heterogeneous information to design a ranking system for multiple products, Decis. Support Syst. 84 (2016) 117–133.
[5] I. Chaturvedi, E. Ragusa, P. Gastaldo, et al., Bayesian network based extreme learning machine for subjectivity detection, J. Franklin Inst. 355 (4) (2018) 1780–1797.
[6] B. Pang, L. Lee, Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, 2005, pp. 115–124.
[7] Y. Zhang, Z. Zhang, D. Miao, et al., Three-way enhanced convolutional neural networks for sentence-level sentiment classification, Inf. Sci. 477 (2019) 55–64.
[8] Y. Liu, J. Bi, Z. Fan, A method for multi-class sentiment classification based on an improved one-vs-one (OVO) strategy and the support vector machine (SVM) algorithm, Inf. Sci. 394 (2017) 38–52.
[9] D. Preotiuc-Pietro, Y. Liu, D. Hopkins, et al., Beyond binary labels: political ideology prediction of twitter users, in: Proceedings of the 55th Annual Meeting on Association for Computational Linguistics, 2017, pp. 729–740.
[10] X. Li, Y. Rao, H. Xie, et al., Bootstrapping social emotion classification with semantically rich hybrid neural networks, IEEE Trans. Affect. Comput. 8 (4) (2017) 428–442.
[11] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, The MIT press, Cambridge, MA, USA, 2016.
[12] Y. LeCun, B.E. Boser, J.S. Denker, et al., Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[13] L. Gatti, M. Guerini, M. Turchi., SentiWords: deriving a high precision and high coverage lexicon for sentiment analysis, IEEE Trans. Affect. Comput. 7 (4) (2016) 409–421.

[14] M. Huang, Q. Qian, X. Zhu, Encoding syntactic knowledge in neural networks for sentiment classification, ACM Trans. Inf. Syst. 35 (3) (2017) 26 1-26:27.
[15] Y. Liu, L. Zhang, L. Nie, et al., Fortune teller: predicting your career path, in: The 30th AAAI Conference on Artificial Intelligence, 2016, pp. 201–207.
[16] Z. Yang, D. Yang, C. Dyer, et al., Hierarchical attention networks for document classification, in: Proceedings of NAACL-HLT, 2016, pp. 1480–1489.
[17] J.W. Pennebaker, L.E. Francis, R.J. Booth, Linguistic Inquiry and Word Count: LIWC, Lawrence Erlbaum Associates Mahwah Nj, 1999.
[18] M. Thelwall, K. Buckley, G. Paltoglou, et al., Sentiment in short strength detection informal text, JASIST 61 (12) (2010) 2544–2558.
[19] N. Kaji, M. Kitsuregawa, Building lexicon for sentiment analysis from massive collection of html documents, in: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007, pp. 1075–1083.
[20] D. Rao, D. Ravichandran, Semi-supervised polarity lexicon induction, in: Proceedings of the 12th Conference of the European Chapter of the ACL, 2009, pp. 675–682.
[21] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up? Sentiment classification using machine learning techniques, in: Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2002, pp. 79–86.
[22] J. Martineau, T. Finin, TFIDF Delta, An improved feature space for sentiment analysis, in: Proceedings of the 3rd International ICWSM Conference, 2009, pp. 258–261.
[23] S. Wang, C. Manning, Baselines and bigrams: simple, good sentiment and topic classification, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, 2012, pp. 90–94.
[24] C. Lin, Y. He, Joint sentiment/topic model for sentiment analysis, in: CIKM, 2009, pp. 375–384.
[25] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modeling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 655–665.
[26] P. Chen, Z. Sun, L. Dong, et al., Recurrent attention network on memory for aspect sentiment analysis, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 463–472.
[27] Y. Zhang, Z. Zhang, D. Miao, et al., Three-way enhanced convolutional neural networks for sentence-level sentiment classification, Inf. Sci. 477 (2019) 55–64.
[28] S. Ruder, P. Ghaffari, J.G. Breslin, A hierarchical model of reviews for aspect-based sentiment analysis, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016, pp. 999–1005.
[29] Q. Qian, M. Huang, J. Lei, et al., Linguistically regularized LSTM for sentiment classification, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017, pp. 1679–1689.
[30] X. Wang, W. Jiang, Z. Luo, Combination of convolutional and recurrent neural network for sentiment analysis of short texts, in: Proceedings of 26th International Conference on Computational Linguistics, 2016, pp. 2428–2437.
[31] L. Yu, J. Wang, K.R. Lai, et al., Refining word embeddings using intensity scores for sentiment analysis, IEEE Trans. Audio Speech Lang. Process. 26 (3) (2018) 671–681.
[32] Q. Hu, J. Zhou, Q. Chen, et al., SNNN: promoting word sentiment and negation in neural sentiment classification, in: The 32nd AAAI Conference on Artificial Intelligence, 2018, pp. 3255–3262.
[33] M. Fares, A. Moufarrej, E. Jreij, et al., Unsupervised word-level affect analysis and propagation in a lexical knowledge graph, Knowl.-Based Syst. 165 (2019) 432–459.
[34] Y. He, S. Sun, F. Niu, et al., A deep learning model enhanced with emoticon semantics for microblog sentiment analysis, Chin. J. Comput. Sci. 40 (4) (2017) 773–790.
[35] R. Amplayo, S. Lee, M. Song, Incorporating product description to sentiment topic models for improved aspect-based sentiment analysis, Inf. Sci. 454–455 (2018) 200–215.
[36] R. Yin, Q. Wang, R. Li, et al., Multi-granularity Chinese word embedding, in: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2016, pp. 981–986.
[37] J. Yu, X. Jian, H. Xin, et al., Joint embeddings of Chinese words, characters, and fine-grained subcharacter components, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2017, pp. 286–291.
[38] X. Zhang, J. Zhao, Y. Lecun, Character-level convolutional networks for text classification, Advances in Neural Information Processing Systems, the, MIT Press, 2015, pp. 649–657.
[39] A. Conneau, H. Schwenk, L. Barrault et al., Very deep convolutional networks for natural language processing, arXiv preprint arXiv:1606.01781 2 (2016).
[40] O. Appel, F. Chiclana, J. Carter, H. Fujita, A hybrid approach to the sentiment analysis problem at the sentence level, Knowl.-Based Syst. 108 (2016) 110–124.
[41] O. Vinyals, L. Kaiser, T. Koo, et al., Grammar as a foreign language, Advances in Neural Information Processing Systems, the, MIT Press, 2015, pp. 2773–2781.
[42] R. Collobert, J. Weston, L. Bottou, et al., Natural language processing (almost) from scratch, J. machine learning research 12 (8) (2011). 2493–537.
[43] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[44] C. Nogueira dos Santos, M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts, in: the 25th International Conference on Computational Linguistics,, 2014, pp. 69–78.
[45] S. Li, Z. Zhao, T. Liu, et al., in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Initializing Convolutional Filters with Semantic Features for Text Classification, 2017, pp. 1885–1890. 2017.
[46] X. Fu, W. Liu, Y. Xu, L. Cui, Combine HowNet lexicon to train phrase recursive autoencoder for sentence-level sentiment analysis, Neurocomputing 241 (2019) 18–27.
[47] B. Liu, Sentiment analysis and subjectivity, in: N. Indurkhya, F.J. Damerau (Eds.), A Chapter in Handbook of Natural Language Processing, 2010.
[48] L. Kong, C. Li, J. Ge, et al., Construction of microblog-specific Chinese sentiment lexicon based on representation learning, in: The Pacific Rim International Conferences on Artificial Intelligence, 2018, pp. 204–216.
[49] S. Li, S. Lee, Y. Chen, et al., Sentiment classification and polarity shifting, in: International Conference on Computational Linguistics, 2010, pp. 635–643.
[50] Z. Lin, X. Jin, X. Xu, et al., Make it possible: multilingual sentiment analysis without much prior knowledge, in: IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), 2014, pp. 79–86.
[51] J. Wang, J. Li, S. Li, et al., Aspect sentiment classification with both word-level and clause-level attention networks, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 4439–4445.
[52] A. Vaswani, N. Shazeer, N. Parmar, et al., Attention is all you need, in: NIPS, 2017, pp. 6000–6010.
[53] J. Devlin, M. Chang, K. Lee, et al., BERT: pre-training of deep bidirectional transformers for language understanding, in: NAACL-HLT, 2019, pp. 4171–4186.
[54] E. Cardoso, R. Silva, T. Almeida, Towards automatic filtering of fake reviews, Neurocomputing 308 (2018) 106–116.
[55] C. Akimushkin, D.R. Amancio, O.N. Oliveira Jr, Text authorship identified using the dynamics of word co-occurrence networks, PLoS ONE 12 (1) (2017) e0170527.
[56] H. Li, Y. Liu, N. Mamoulis, et al., Translation-based sequential recommendation for complex users on sparse data, IEEE Trans. Knowl. Data Eng. (2019), doi:10.1109/TKDE.2019.2906180.