Knowledge-Based Systems 233 (2021) 107525

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

AMFF: A new attention-based multi-feature fusion method for intention recognition

Cong Liu [a], Xiaolong Xu [b,*]

[a] Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China
[b] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China

ARTICLE INFO

Article history:
Received 22 December 2020
Received in revised form 16 September 2021
Accepted 20 September 2021
Available online 22 September 2021

Keywords:
Intention recognition
Multi-feature fusion
Short text
Classification

ABSTRACT

Intention recognition is based on a dialog between users to identify their real intentions, which plays a key role in the question answering system. However, the content of a dialog is usually in the form of short text. Due to data sparsity, many current classification models show poor performance on short text. To address this issue, we propose AMFF, an attention-based multi-feature fusion method for intention recognition. In this paper, we enrich short text features by fusing features extracted from frequency-inverse document frequency (TF-IDF), convolutional neural networks (CNNs) and long short-term memory (LSTM). For the purpose of measuring the important features, we utilize the attention mechanisms to assign weights for the fusion features. Experimental results on the TREC, SST1 and SST2 datasets demonstrate that the proposed AMFF model outperforms traditional machine learning models and typical deep learning models on short text classification.

1. Introduction

Recently, question answering systems (QAS) have become a hot research topic and have drawn much attention from scientific and industrial communities. QAS is an advanced information retrieval system that includes three parts: problem understanding, information retrieval, and answer extraction. Question understanding, as the core function of QAS, reduces the range of answers and improves the speed and accuracy of answer retrieval by determining the category of answers.

In QAS, questions are usually in the form of short text. With the rapid development of online social media, short texts, such as questions in question answering systems, records of chat software and messages on forums, have started to play a vital role on the Internet. Short texts have become an important form for individuals to voice opinions and share information on online platforms [1]. Short text classification can be widely applied in many domains, ranging from user query intention classification to sentiment analysis [2,3]. Sentiment analysis exploits both computer and social sciences to better recognize, interpret, and process opinions and sentiments over the Web [4]. Compared with long texts, short texts have the following characteristics [5]:

Short length. Short text can be a small paragraph, a sentence, or even a phrase, which frequently appears in news headlines, question answering systems and online social media.

Sparse features. Short text contains only a dozen words with practical significance. It is difficult to extract words with effective features from them.

Real-time and large-scale. The number of short text messages on the Internet is large-scale and growing rapidly.

Nonstandard format. Although short text is generally concise, there are still some irregular words or spelling errors, which greatly increases the digital noise.

This indicates that feature sparseness is the key problem restricting the accuracy of the model. For short texts with few words and sparse features, it is important to use the text context semantic information to enrich its text representation [6].

Short text classification based on conventional machine learning methods is divided into rule-based methods and statistic-based methods. Rule-based methods mainly rely on expert knowledge, which classifies text by setting certain rules for datasets. Statistics-based methods use feature extraction approaches, such as the term frequency-inverse document frequency (TF-IDF), to calculate the weight of each word. However, conventional machine learning methods generally use bag-of-words to represent text, which may lose word order information and cause the text feature dimension to be too high and too sparse. Especially

* Corresponding author.
E-mail address: xuxl@njupt.edu.cn (X. Xu).

https://doi.org/10.1016/j.knosys.2021.107525
0950-7051/

in short text, severe data sparsity results in limited features available for classifiers.

In recent years, deep learning has become the mainstream method of text classification [7]. Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have become common models in natural language processing (NLP). However, these models are inevitably compromised when applied to short texts.

To address the limitation of the above models, we propose an attention-based multi-feature fusion method (AMFF) and apply it to the task of short text classification. The main contributions of this paper are summarized as follows:

- We propose a short text feature extraction method. TF-IDF features are used to build a feature dictionary, which is applied to the text. We combine it with the features extracted by CNN and LSTM to enrich the short text features.
- We use the attention mechanism to assign weights. The three types of features have different angles. To prevent feature conflicts, we use the attention mechanism to assign weights to stabilize the model.

The remainder of this paper is organized as follows: In Section 2, related work is reviewed. It also demonstrates how our work is different from existing research. In Section 3, the framework and core components of the proposed attention-based multi-feature fusion method are presented. The experimental datasets, setup, and performance evaluation results are presented in Section 4 to verify the effectiveness of the proposed approach. Finally, this paper is concluded, and future research directions are discussed in Section 5.

2. Related work

The automatic classification of short texts requires additional knowledge. Traditional methods cannot achieve the expected results in this field. The current short text classification methods can be put into two categories.

One type focuses on feature expansion. Phan et al. [8] used latent Dirichlet allocation (LDA) to discover potential semantics and extend short text semantics. Wang et al. [9] used the external entity information from knowledge bases to establish the concept model for each category, and the short text was classified by concept similarity. In addition, broader feature expansion methods have been designed. Ning et al. [10] improved the feature by performing similarity calculations on features, based on HowNet. Both Bollegala et al. [11] and Sahami et al. [12] proposed inputting short text as a search term into a search engine and calculated short text similarity based on search results. Zeng et al. [13] focused on extending features with external knowledge or pre-trained topics. However, these methods are not able to achieve good performance since feature engineering relies on domain knowledge [14]. Dulan et al. [15] proposed analyzing semantic text to identify racial comments and built a 2-class SVM based on the Microsoft Azure machine learning studio to classify short text comments. These methods are usually designed for specific scenarios, rely on domain knowledge, and have low computational efficiency, which are difficult to generalize to other scenarios.

The other type is based on deep learning. Deep neural networks, representing texts by word embedding [16], have been widely used in text classification. The word embedding layer, which is initialized by pre-trained word embedding, such as word2vec and GloVe, can effectively improve the performance of short text classification. The word embedding enables us to measure word semantics by the distance between two embedding vectors.

Two representative deep neural network models, RNN and CNN, have been widely used in text classification. Kim [17] combined CNN and word embedding and proved that CNN trained on top of pretrained word vectors is better than traditional machine learning methods in text classification. Kalchbrenner et al. [18] proposed the dynamic convolutional neural network (DCNN), which used dynamic k-max pooling to model sentences.

RNN utilizes the recurrent unit and the parameter sharing approach to perform modeling sequence data [19]. Long short-term memory (LSTM) is a typical RNN architecture that is designed to address the vanishing and exploding gradient problems of RNNs [20]. Sakar et al. [21] presented that a multilayer perceptron and LSTM network can be combined to identify real-time online users' purchase intentions. Cho et al. [22] proposed the gated recurrent unit (GRU), which has higher efficiency than LSTM. The research of Gennaro et al. [23] shows that LSTM has an excellent performance in intention classification. Khattak et al. [24] proposed a convolutional neural network and long short-term memory for efficient detection of intention in a given review, which maintains the sequence correlation and retains information for a long time span. Li et al. [25] performed context compositionality and sentiment classification based on a generalized neural tensor block and a two-channel classifier.

In recent years, combining traditional machine learning methods and deep learning methods to enrich text information has become a popular research topic. Luong et al. [26] utilized short texts from online social media channels to extract intent information, which achieved good results by combining conditional random fields (CRFs) and bidirectional long short-term memory (Bi-LSTM) on social media text posts. Ling et al. [27] proposed a novel structure of a hybrid neural network model. In feature extraction, they used embeddings from language models (ELMos) and the co-occurrence statistical features between words in Weibo. Then, the outputs of the multichannel CNN and Bi-LSTM are concatenated. Pathik et al. [28] applied latent Dirichlet allocation for aspect extraction and two-layer Bi-LSTM for sentiment classification. Cambria et al. [29] integrated logical reasoning within deep learning architectures to build a new version of SenticNet, which achieved impressive performance in sentiment analysis.

The attention mechanism [30] has been used in natural language processing, such as machine translation and the information filter of text classification. Zhang [31] used LSTM to code the feature matrix after word embedding and then input the result to the attention layer to obtain deep feature representations. Xu et al. [32] designed a neural network to integrate context-sensitive knowledge into a CNN for short text classification. These methods based on deep learning are not limited to specific scenarios, while the problem of feature sparsity has not been solved. Usama et al. [33] used the attention mechanism to calculate the attention score from the feature context generated from CNN filters, which makes the features that contribute much to the prediction task more significant. Basiri et al. [34] applied an attention mechanism on the outputs of two independent bidirectional LSTM and GRU layers to place more, or less, emphasis on different words. Yang et al. [35] combined CNN and the attention-based bidirectional gated recurrent unit (BiGRU) for sentiment analysis.

Stacking multiple feature extraction models is also a solution to the problem of feature sparsity. Görmez et al. [36] also proposed a stacked ensemble method, which systematically combines six feature extraction methods and three classifiers. Learning [37] proposed a framework for a hybrid approach with an ensemble of stacked machine learning algorithms and dictionary-based classifiers. Akhtar et al. [38] proposed a stacked ensemble method for predicting the degree of intensity for emotion and sentiment by combining the outputs obtained from deep learning

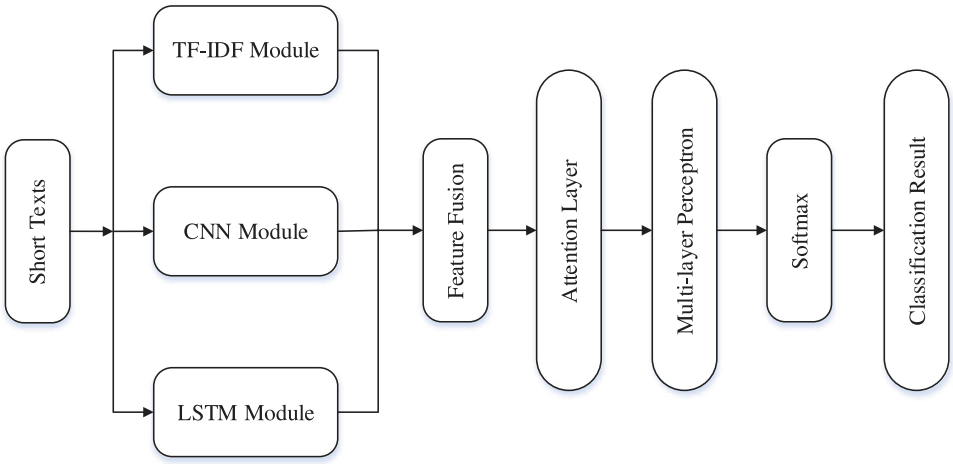

Fig. 1. The framework of AMFF.

models and the classical supervised model based on support vector regression with a multilayer perceptron network. The proposed model shows impressive results. Zhu et al. [39] proposed a method that extracts text features by integrating attention-based bidirectional extended short-term memory networks (AttBiLSTM) and multichannel CNN (MCNN) networks to improve the text representation of the model.

In addition, the pretraining model also brings new challenges. The time and space cost of pretraining methods has increased, and the performance of the model depends on the quality of the pretraining corpus. Xiao et al. [6] proposed the "MD-Intention", which combined BERT and ELMo to mine more comprehensive contextual semantic information from different dimensions to enrich the text representation.

However, this is not related to the work in this paper, which will not be discussed in detail here. The analysis of the current methods is shown in Table 1.

The AMFF model proposed in this paper is different from the existing methods:

- We use word embedding to vectorize words, not depending on specific scenarios, which improves the scalability of the model.
- We fuse features extracted from three methods to enrich short text features to resolve feature sparsity.
- We add the attention layer to judge key features in fusion automatically.

3. Model

3.1. Framework of AMFF

In this section, we present AMFF, the proposed attention-based multi-feature fusion method. The framework of AMFF is shown in Fig. 1. AMFF consists of the following four parts:

(1) Data preprocessing. The raw text usually contains digital noise, such as abbreviations and special characters. Preprocessing makes the text data cleaner, reduces irrelevant information and improves the quality of the text.

(2) Feature extraction and fusion. We use different features extracted from TF-IDF, LSTM and CNN to enrich the problem feature information.

(3) Attention weighting. The attention mechanism is used to assign different weights to each feature to distinguish the key information in the fusion feature and stabilize the effect of the model.

(4) Output of classification results. The output of the network is the probability distribution of class labels.

3.2. Feature extraction and fusion

The feature extraction includes three modules: the TF-IDF module, the CNN module and the LSTM module.

TF-IDF Module

TF-IDF consists of term frequency (TF) and inverse document frequency (IDF). TF is calculated with the following equation:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_{k=1}^{k} n_{k,j}} \tag{1}$$

where $tf_{i,j}$ represents the frequency of word w_i in document d_i and $n_{i,j}$ is the count of w_i in document d_i. The denominator is the synthesis of the occurrence times of all words in document d_i, and k is the number of different words in document d_j.

IDF is calculated with the following equation:

$$idf_i = \log \frac{n_d}{df\ (d, w_i) + 1} \tag{2}$$

where idf_i represents the reverse document frequency of word w_i in text library d and n_d is the total number of documents in d. $df\ (d, w_i)$ is the number of documents containing w_i in d, and 1 is added to prevent $df(d, w_i)$ from being 0.

The value of TF-IDF is normalized with:

$$tf - idf_{i,j} = tf_{i,j} \times idf_{i,j} \tag{3}$$

The workflow of the TF-IDF module is shown in Fig. 2:

(1) Data preprocessing. The format of the short text is not standard. To avoid additional noise, we need to clean the data. The specific steps are detailed in Section 4.2.

(2) Category enhancement. It is difficult for TF-IDF to play a role in processing short texts because the value is calculated according to text statistics. We connect the sentences of the same category after preprocessing back and forth to form a longer document. According to the categories, a corresponding number of long texts is formed, enhancing the amount of information in the TF-IDF solution process.

(3) TF-IDF calculation. We calculate the TF-IDF matrix for each type of spliced long text, in which the rows represent long text data, and each value is the TF-IDF value of the corresponding word.

(4) Feature dictionary building. After calculating the TF-IDF value, each category will generate a one-to-one TF-IDF value corresponding to the word in the document. The correspondence between multiple categories can be regarded as a matrix. We use the words in the matrix to build a dictionary and write its corresponding TF-IDF value. The larger the value of TF-IDF is the

Table 1
Analysis summary on current methods.

		Feature	Disadvantage
Feature Expansion		Extending text features to obtain richer features	Scene limitations, low computational efficiency, and difficulty to promote
Deep Learning	Neural Network	Using word embedding and deep neural networks	The problem of feature sparsity of short text is not solved.
	Attention Mechanism	Integrating attention mechanism into a deep neural network	

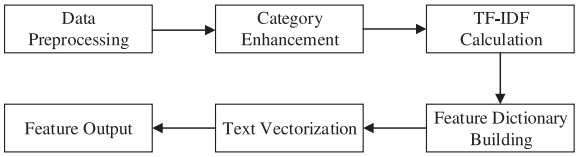

Fig. 2. The workflow of the TF-IDF Module.

more important the word is in this category. We take the largest value of the same word in the feature dictionary.

(5) Text vectorization. We obtain the corresponding TF-IDF feature of the training data by mapping the texts according to the feature dictionary.

(6) Feature matrix output. Finally, the feature matrix is output.

CNN is an optimized feedforward neural network that relies on the convolution between the input matrix and different convolution kernels. Kim [17] proposed the representative convolutional neural network model for text classification, as shown in Fig. 3.

The model consists of the word embedding layer, the convolution layer and the pooling layer. After word embedding, the word vector of each word is $x_i \in \mathbb{R}^{n\times d}$, where n is the sentence length and d is the vector dimension.

The convolutional layer extracts matrix features after word embedding, which obtains features with different semantic features by different convolution kernels. It extracts sentence features through a set of filters:

$$\boldsymbol{C}_i = f\left(w \times x_{i:i+g-1} + b\right) \tag{4}$$

where w is the weight of the convolution kernel, g is the size of the convolution kernel, $x_{i:i+g-1}$ represents a sentence vector consisting of word vectors from i to $i+g-1$, b is a bias, and is a nonlinear activation function such as *relu* or *tanh*. The feature matrix $\boldsymbol{C}$, $\boldsymbol{C} = [c_1, c_2, \ldots, c_{n-g+1}]$, is obtained through the convolution layer.

The most representative feature in the feature vector is obtained with max-pooling, which can realize feature reduction:

$$\boldsymbol{M} = \max\left(c_1, c_2, \ldots, c_{n-g+1}\right) = \max\{\boldsymbol{C}\} \tag{5}$$

These features are stitched together to form the CNN features.

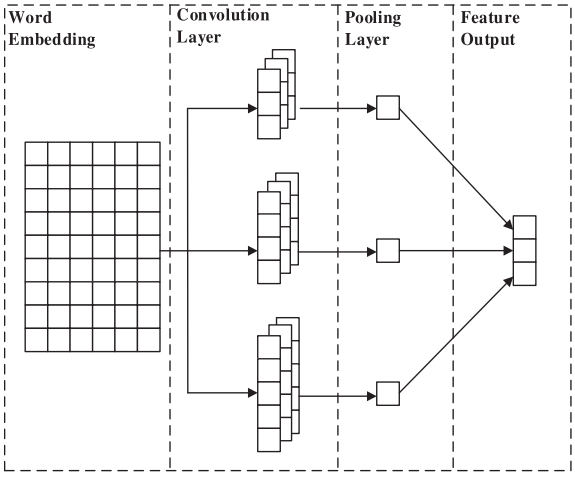

Fig. 3. The CNN model for text classification.

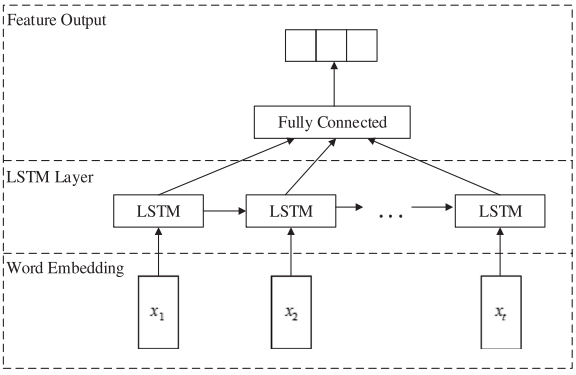

Fig. 4. The structure of the LSTM Module.

LSTM Module

This word vector matrix is input into the LSTM layer, and then the multilayer perceptron is used to adjust the feature output to a suitable size.

LSTM has memory ability, which can control the storage and release of state data. This characteristic enables the interaction between two words far apart in text. The basic external structure of LSTM and the neural connection mode are shown in Fig. 4.

In addition, LSTM can also accept and transfer the unit as well as the hidden layer status.

The input gate, forgetting gate f_t and output gate o_t updating method are shown in (6)–(8):

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right) \tag{6}$$

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right) \tag{7}$$

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{8}$$

where σ represents the *sigmoid*, W_i, W_f and W_o represent the weight matrices of the input gate, the forgetting gate, and the output gate, respectively, and b_i, b_f and b_o represent the corresponding biases.

The forgetting gate and the input gate form the updating gate, which changes the state of the unit. The updating gate is calculated with:

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh\left(W_c\left[h_{t-1}, x_t\right] + b_c\right) \tag{9}$$

where c_{t-1} is the value of c at the last moment, W_c is the weight matrix of the update gate, and b_c is its bias. h_t represents the state of the LSTM hidden layer at the current time step, $h \in R^{n\times 1}$. n is

the dimension of the hidden layer. h_t is calculated with:

$$h_t = o_t \otimes \tanh(c_t) \tag{10}$$

The gate structure of the LSTM unit is used to limit the amount of information, which makes LSTM cells remember the historical information. The gates make up for the shortcomings of the RNN network.

The output obtained with the three modules is a vector. The CNN module has three different convolution kernels to generate three vectors. We adjust the length of the vector to a uniform length. The specific adjustment is to add a fully connected layer and obtain a vector of equal length by setting the number of nodes. The splicing process involves splicing according to the long side of the vector. Finally, we can obtain the corresponding feature matrix.

3.3. Attention weighting

After feature stitching, the matrix is input to the attention layer. We use the attention mechanism to assign to the feature matrix. To assign weights and not lose the feature information, we keep the output and input dimensions unchanged. We choose the self-attention mechanism for this layer:

$$z = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

where Q, K and V are 3 different vectors for each feature, which are obtained by multiplying the feature vector X by different weight matrices, W^Q, W^K and W^V. QK^T is the score. The gradient stability is normalized by dividing by . The *softmax* function is used as the activation, and the final result is obtained by multiplying V.

3.4. Classification layer

The attention-weighted feature matrix is input to the multi-layer perceptron for further learning. It is adjusted to the appropriate size and input to the *softmax* function:

$$P(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}} \tag{12}$$

where x_i is the input, and the classification result is the category with the highest probability after the softmax.

4. Datasets and experimental setup

4.1. Datasets

We implemented experiments on three typical datasets, TREC, SST-1, and SST-2, as shown in Table 2:

TREC [40]: The TREC dataset is a popular problem classification dataset. In TRECs, questions are divided into six categories: value, human, description, entity, number and abbreviation. There were 5452 labeled questions in the training set and 500 labeled questions in the test set. We utilized 10% of the training set as the development set.

SST-1 [41]: The SST-1 dataset is the Stanford sentiment treebank dataset. In SST-1, there are 11800 movie comments with five labels, including very positive, positive, neutral, negative, and very negative. There are 8544, 1101, and 2210 labeled records in the training, development, and test set, respectively.

SST-2 [41]: The SST-2 dataset is built based on SST-1. In SST-2, all neutral comments were removed. The very positive and positive categories were merged into one category, and the very negative and negative categories were merged into one category. Extra phrases are added to SST-2 to enhance the classification performance. SST-2 is for binary classification.

Table 2
Datasets for experiments.

Name	Class number	Train/Dev/Test
TREC	6	4906/546/500
SST-1	5	8544/2240/1101
SST-2	2	6920/1821/872

Table 3
Parameters.

Parameter	Class number
e	100
l	30
h	3, 4, 5
p	0.5
m	128
Pooling	1-max pooling
Optimizer	Adam
Learning rate	0.001

4.2. Data preprocessing

Most of the raw text is unstructured and full of noise. To obtain clean text, we need to preprocess the data with: (1) replacing the abbreviation and upper case with the complete format and lower case, (2) using regularization to remove punctuation, numbers and other special characters, (3) using the tokenizer to segment text with spaces, (4) checking texts with spell checking tools, in order to avoid the misspelling, (5) and deleting the stop words that have no impact on the classification, such as “a”, “is”, etc.

We use GloVe [42] to initialize the embedding layer. The vectors have the dimensionality of 300. Words not presented in the set of pretrained words are initialized randomly.

4.3. Parameters setting

We chose the parameter values that made AMFF achieve the best results on the dev set of SST-2. We initialize words not by random sampling the uniform distribution on [- 0.1, 0.1]. The sentence length l is fixed at 30.

In the CNN and LSTM module, for all datasets we set filter windows (h) of 3, 4, 5 with 128 feature maps (m) each, the vector size (e) of the word embedding as 100, the Adam optimizer with 0.001 learning rate, dropout rate (p) of 0.5, the hidden layer of LSTM as 128 and attention nodes as 50. We used accuracy as the metric. The detailed parameters are shown in Table 3.

4.4. Baseline methods

We compared AMFF with following methods:

TF-IDF+SVM: TF-IDF+SVM calculates TF-IDF features to form a list of vectors and then uses SVM to classify the feature vectors.

CNN: Kim [17] combined word embedding with CNN. This method uses the pre-trained word vector to quantize text and then uses multiple convolution kernels to extract text features in parallel to improve text classification performance.

DCNN: DCNN, proposed by Kalchbrenner et al. [18], is a CNN with k-max pooling.

GRU: Cho et al. [22] proposed the GRU improved by LSTM, which has fewer parameters and performs better in convergence than LSTM. The model includes Gru feature extraction and *softmax*.

Attention-based LSTM: Zhang [31] designed an attention-based LSTM. Attention-based LSTM combines LSTM with the attention mechanism to perform feature extraction to obtain deep

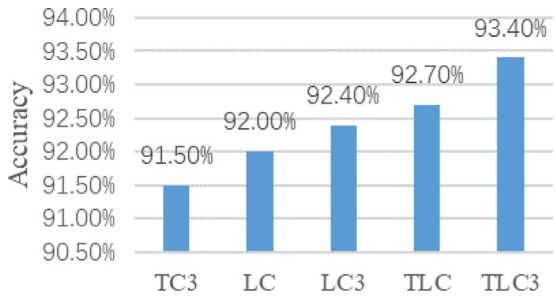

Fig. 5. The accuracy of different feature combinations.

Table 4
Accuracy of compared models on different datasets.

Model	TREC	SST-1	SST-2
TF-IDF+SVM	82.1%	–	–
DCNN	90.2%	45.2%	82.5%
CNN	91.4%	45.0%	83.6%
GRU	91.0%	42.3%	80.6%
Attention-based LSTM	92.2%	46.0%	84.2%
CCRCNN	93.7%	45.3%	84.1%
AMFF	93.4%	46.3%	84.6%

feature representations, which are input into *softmax* to complete classification.

CCRCNN: Xu et al. [32] designed CCRCNN, a neural network to integrate context-sensitive knowledge into CNN.

4.5. Results and analysis

Table 4 shows the classification accuracy of different methods on the three datasets.

Overall Performance

Compared with TF-IDF+SVM, methods based on neural networks show better overall performances. This proves that the utilization of the GloVe word vector can significantly improve the performance of the model. Approaches based on neural networks can effectively compose semantic text information.

Compared with GRUs, both CNN and the proposed AMFF achieve better results. This illustrates that convolution kernels of different sizes are more suitable for constructing deeper semantic representations of texts than recurrent-based methods.

Compared with CNN and DCNN, AMFF outperforms them in all cases. The reason is that the window-based structure ignores the sequential features so that it cannot capture the contextual information well. These results demonstrate the effectiveness of AMFF utilizing the features extracted by LSTM.

Compared with the attention-based LSTM, the results show that richer features used by AMFF are helpful for short text classification. We can also see that the attention-based LSTM outperforms GRU, which demonstrates that the attention mechanism can ensure the integrity of features and prevent the loss of important information.

Feature effect

We implemented experiments on the fusion of different features on the TREC dataset. The results are shown in Fig. 5.

In Fig. 5, T, L and C represent the features extracted by the TF-IDF, LSTM, and CNN modules, respectively. C3 represents the result output with three different convolution kernels in the CNN module; TC3 is the fusion of one TF-IDF feature and CNN features calculated with three kinds of convolution kernels. C refers to the result obtained by only the convolution kernel with a height of three. In the comparison between TC3 and CNN in Table 4, large margin improvements are not observed on the dataset. This reason may be that the two methods wreck the sequential features of texts and cannot complement each other. It is further proved that LC3 has a noticeable improvement.

In comparing LC and LC3, TLC and TLC3, we explore adjusting the size of CNN features before the fusion layer. We can see that the accuracy is decreased. These results show that convolution kernels of different sizes in CNN extract features in different dimensions. If fused in advance, some features will be lost, which is equivalent to reducing the weight of CNN extracted features.

The excellent fusion effect of CNN and LSTM indicates that the serialization information and local spatial information are complementary information forms in text features. The information of the LSTM module has the greatest impact on the results, indicating that serialization information is the most significant feature of the text. The improved TF-IDF method proposed in this paper can further improve the accuracy and effectiveness.

Attention effect

We implemented experiments with the model with and without the attention layer on SST-2 to explore the effects of the attention layer. The results are shown in Fig. 6.

In Fig. 6, Attention_acc is the model's accuracy with the attention layer, while NonAttention_acc is the model's accuracy without the attention layer. It can be clearly seen that the accuracy of the model has been improved, and the convergence speed and stability of the model have also been improved significantly. If the features are fused into the classifier directly, the accuracy is improved to some extent, whereas the model becomes unstable, and the accuracy fluctuates greatly. The reason is that the three

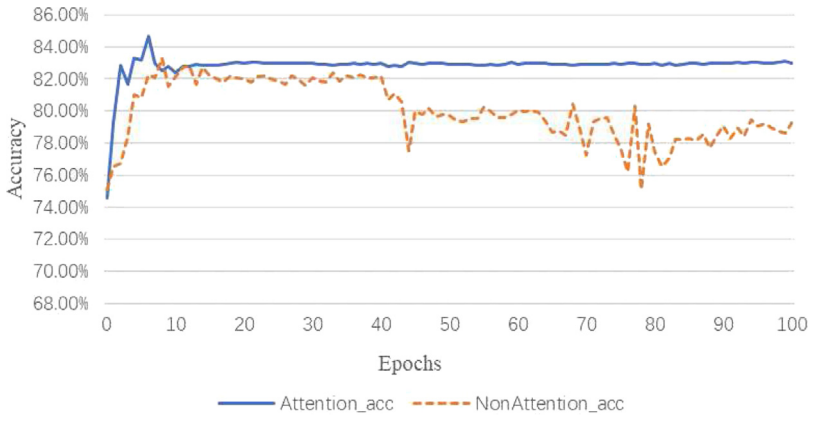

Fig. 6. The accuracy of different feature combinations.

features are extracted from different approaches, which makes them quite different.

If different features cooperate and play their respective roles, the model's performance will be improved; otherwise, the fusion of features will be useless or even harmful to the model's performance. The attention layer makes the model weight the features before importing them to the classification layer. This makes features with positive effects gain high weights and reduces the impact of noise.

5. Conclusion

In this paper, we proposed AMFF, the attention-based multi-feature fusion method for intention recognition. AMFF enriches the features of short texts by fusing statistical-based features, serialized features, and deep features extracted by different convolution kernels. To address the problem of unclear focus caused by multiple features, we use the attention mechanism to assign weights to features, which solves information loss in feature extraction and fusion. Experiments on all three public datasets show that AMFF outperforms the baseline machine-learning-based models and deep-learning-based models. However, some problems are to be solved, such as the overabundant parameters and the simple feature stitching approach. In the future, we plan to simplify the model to ensure the performance of the model and improve the weighting approach.

CRediT authorship contribution statement

Cong Liu: Discussions, analyses and implementation of the proposed method, Implementing the experiments and writing the manuscript. **Xiaolong Xu:** Discussions, analyses and implementation of the proposed method, Revising the manuscript.

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.

Acknowledgments

We want to thank the reviewers and editor for their comments and advice for helping us improve the quality of this paper. This work was supported by the National Natural Science Foundation of China under Grant 62072255. All authors read and approved the final manuscript.

References

[1] Y.Y. Zhang, J. Li, Y. Song, et al., Encoding conversation context for neural keyphrase extraction from microblog posts, in: Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 1676–1686, http://dx.doi.org/10.18653/v1/N18-1151.

[2] Y. Meng, J.M. Shen, C. Zhang, et al., Weakly-supervised neural text classification, in: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, 2018, pp. 983–992, http://dx.doi.org/10.1145/3269206.3271737.

[3] C.C. Aggarwal, C.X. Zhai, A survey of text classification algorithms, in: C.C. Aggarwal, C.X. Zhai (Eds.), Mining Text Data, Springer, Boston, MA, 2012, pp. 163–222, http://dx.doi.org/10.1007/978-1-4614-3223-4_6.

[4] Y. Susanto, E. Cambria, B.C. Ng, et al., Ten years of sentic computing, Cogn. Comput. (2021) 1–19, http://dx.doi.org/10.1007/s12559-021-09824-x.

[5] S. Ge, Y. Ye, X. Du, et al., Short text classification: A survey, J. Multimedia 5 (2014) 635–643, http://dx.doi.org/10.4304/jmm.9.5.635-643.

[6] M. Xiao, M. Yao, Y. Li, Short-text intention recognition based on multi-dimensional dynamic word vectors, J. Phys. Conf. Ser. 1 (2020) 01208, http://dx.doi.org/10.1088/1742-6596/1678/1/012080.

[7] D.Y. Tang, B. Qin, T. Liu, Deep learning for sentiment analysis: successful approaches and future challenges, Wiley Interdiscipl. Rev.: Data Min. Knowl. Discov. 6 (2015) 292–303, http://dx.doi.org/10.1002/widm.1171.

[8] X. Phan, L. Nguyen, S. Horiguchi, Learning to classify short and sparse text & web with hidden topics from large-scale data collections, in: Proceedings of the 17th International Conference on World Wide Web, 2008, pp. 91–100, http://dx.doi.org/10.1145/1367497.1367510.

[9] F. Wang, Z.Y. Wang, J.R. Li, et al., Concept-based short text classification and ranking, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 1069–1078, http://dx.doi.org/10.1145/2661829.2662067.

[10] Y.H. Ning, X.H. Fan, Y. Wu, Short text classification based on domain word ontology, Comput. Sci. 3 (2009) 142–145, http://dx.doi.org/10.3969/j.issn.1002-137X.2009.03.038.

[11] D. Bollegala, Y. Matsuo, M. Ishizuka, Measuring semantic similarity between words using web search engines, in: Proceedings of the 16th International Conference on World Wide Web, 2007, pp. 757–766, http://dx.doi.org/10.1145/1242572.1242675.

[12] M. Sahami, T.D. Heilman, A web-based kernel function for measuring the similarity of short text snippets, in: Proceedings of the 15th International Conference on World Wide Web, 2006, pp. 377–386, http://dx.doi.org/10.1145/1135777.1135834.

[13] J. Zeng, J. Li, Y. Song, et al., Topic memory networks for short text classification, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 3120–3131, http://dx.doi.org/10.18653/v1/D18-1351.

[14] L. Hu, T. Yang, C. Shi, et al., Heterogeneous graph attention networks for semi-supervised short text classification, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, 2019, pp. 4821–4830, http://dx.doi.org/10.18653/v1/D19-1488.

[15] D.S. Dias, M.D. Welikala, N.G. Dias, Identifying racist social media comments in sinhala language using text analytics models with machine learning, in: Proceedings of the 2018 18th International Conference on Advances in ICT for Emerging Regions, 2018, http://dx.doi.org/10.1109/ICTER.2018.8615492.

[16] G.E. Hinton, Learning distributed representations of concepts, in: Proceedings of the 8th Annual Conference of the Cognitive Science Society, 1986, pp. 1–12.

[17] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of Empirical Methods in Natural Language Processing, 2014, pp. 1746–1751, http://dx.doi.org/10.3115/v1/D14-1181.

[18] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, 2014, pp. 655–665, http://dx.doi.org/10.3115/v1/P14-1062.

[19] C. Xu, W.R. Huang, H.W. Wang, et al., Modeling local dependence in natural language with multi-channel recurrent neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2019, pp. 5525–5532, http://dx.doi.org/10.1609/aaai.v33i01.33015525.

[20] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 8 (1997) 1735–1780, http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[21] C.O. Sakar, S.O. Polat, M. Katircioglu, et al., Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks, Neural Comput. Appl. 10 (2019) 6893–6908, http://dx.doi.org/10.1007/s00521-018-3523-0.

[22] K. Cho, B. van Merriënboer, C. Gulcehre, et al., Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the Empiricial Methods in Natural Language Processing, 2014, pp. 1724–1734, http://dx.doi.org/10.3115/v1/D14-1179.

[23] G.D. Gennaro, A. Buonanno, A.D. Girolamo, et al., Intent classification in question-answering using LSTM architectures, in: Progresses in Artificial Intelligence and Neural Systems, Smart Innovation, Systems and Technologies, 2014, pp. 115–124, http://dx.doi.org/10.1007/978-981-15-5093-5_11.

[24] A. Khattak, A. Habib, M.Z. Asghar, et al., Applying deep neural networks for user intention identification, Soft Comput. 3 (2021) 2191–2220, http://dx.doi.org/10.1007/s00500-020-05290-z.

[25] W. Li, W. Shao, S. Ji, et al., Bieru: bidirectional emotional recurrent unit for conversational sentiment analysis, 2020, arXiv preprint arXiv:2006.00492, https://www.researchgate.net/publication/341816987_BiERU_Bidirectional_Emotional_Recurrent_Unit_for_Conversational_Sentiment_Analysis.

[26] T.L. Luong, M.S. Cao, D.T. Le, et al., Intent extraction from social media texts using sequential segmentation and deep learning models, in: Proceedings of the 2017 9th International Conference on Knowledge and Systems Engineering, 2017, pp. 215–220, http://dx.doi.org/10.1109/KSE.2017.8119461.

[27] M. Ling, Q. Chen, Q. Sun, et al., Hybrid neural network for sina weibo sentiment analysis, IEEE Trans. Comput. Soc. Syst. 4 (2020) 983–990, http://dx.doi.org/10.1109/TCSS.2020.2998092.

[28] N. Pathik, P. Shukla, An efficient sentiment analysis using topic model based optimized recurrent neural network, Int. J. Smart Sensing Intell. Syst. 11 (2021) 1–12, http://dx.doi.org/10.21307/ijssis-2021-011.
[29] E. Cambria, Y. Li, F. Xing, S. Poria, K. Kwok, SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 105–114, http://dx.doi.org/10.1145/3340531.3412003.
[30] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014, arXiv preprint arXiv:1409.0473, http://de.arxiv.org/pdf/1409.0473.
[31] C. Zhang, Text Classification Based on Attention-Based LSTM Model, Nanjing University, 2016.
[32] J. Xu, Y. Cai, X. Wu, et al., Incorporating context-relevant concepts into convolutional neural networks for short text classification, Neurocomputing 386 (2020) 42–53, http://dx.doi.org/10.1016/j.neucom.2019.08.080.
[33] M. Usama, B. Ahmad, E. Song, et al., Attention-based sentiment analysis using convolutional and recurrent neural network, Future Gener. Comput. Syst. 113 (2020) 571–578, http://dx.doi.org/10.1016/j.future.2020.07.022.
[34] M.E. Basiri, S. Nemati, M. Abdar, et al., ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis, Future Gener. Comput. Syst. 115 (2021) 279–294, http://dx.doi.org/10.1016/j.future.2020.08.005.
[35] L. Yang, Y. Li, J. Wang, et al., Sentiment analysis for E-commerce product reviews in Chinese based on sentiment lexicon and deep learning, IEEE Access 8 (2020) 23522–23530, http://dx.doi.org/10.1109/ACCESS.2020.2969854.
[36] Y. Görmez, Y.E. Işık, M. Temiz, et al., FBSEM: A novel feature-based stacked ensemble method for sentiment analysis, Int. J. Inf. Technol. Comput. Sci. 12 (2020) 11–22, http://dx.doi.org/10.5815/ijitcs.2020.06.02.
[37] S. Rani, N.S. Gill, Hybrid model for Twitter data sentiment analysis based on ensemble of dictionary based classifier and stacked machine learning classifiers-SVM, KNN and C5.0, J. Theoret. Appl. Inf. Technol. 98 (2020) 624–635.
[38] M.S. Akhtar, A. Ekbal, E. Cambria, How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble application notes, IEEE Comput. Intell. Mag. 15 (2020) 64–75, http://dx.doi.org/10.1109/MCI.2019.2954667.
[39] X. Zhu, S. Yin, Z. Chen, Attention based BiLSTM-MCNN for sentiment analysis, in: Proceedings of the 5th IEEE International Conference on Cloud Computing and Big Data Analytics, 2020, pp. 170–174, http://www.researchgate.net/publication/341500911_Attention_Based_BiLSTM-MCNN_for_Sentiment_Analysis.
[40] X. Li, D. Roth, Learning question classifiers: the role of semantic information, in: Proceedings of the 19th International Conference on Computational Linguistics, 2002, pp. 1–7, http://www.researchgate.net/profile/Xin_Li236/publication/220597341_Learning_question_classifiers_the_role_of_semantic_information_Nat_Lang_Eng/links/5691d8ea08aee91f69a5221a.pdf.
[41] R. Socher, A. Perelygin, et al., Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 2013, pp. 1631–1642, https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf.
[42] J. Pennington, R. Socher. C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, 2014, pp. 1532–1543, http://dx.doi.org/10.3115/v1/D14-1162.