2-2022

CrowdTC: Crowd-powered learning for text classification

Keyu YANG

Yunjun Gao

Lei LIANG

Song BIAN

Lu CHEN

See next page for additional authors

CrowdTC: Crowd-powered Learning for Text Classification

KEYU YANG, YUNJUN GAO, LEI LIANG, SONG BIAN, and LU CHEN,
Zhejiang University, China
BAIHUA ZHENG, Singapore Management University, Singapore

Text classification is a fundamental task in content analysis. Nowadays, deep learning has demonstrated promising performance in text classification compared with shallow models. However, almost all the existing models do not take advantage of the wisdom of human beings to help text classification. Human beings are more intelligent and capable than machine learning models in terms of understanding and capturing the implicit semantic information from text. In this article, we try to take guidance from human beings to classify text. We propose Crowd-powered learning for Text Classification (CrowdTC for short). We design and post the questions on a crowdsourcing platform to extract keywords in text. Sampling and clustering techniques are utilized to reduce the cost of crowdsourcing. Also, we present an attention-based neural network and a hybrid neural network to incorporate the extracted keywords as human guidance into deep neural networks. Extensive experiments on public datasets confirm that CrowdTC improves the text classification accuracy of neural networks by using the crowd-powered keyword guidance.

CCS Concepts: • **Information systems** → **Content analysis and feature selection**; *Document topic models;*

Additional Key Words and Phrases: Text classification, crowdsourcing, keyword extraction, neural networks

ACM Reference format:
Keyu Yang, Yunjun Gao, Lei Liang, Song Bian, Lu Chen, and Baihua Zheng. 2021. CrowdTC: Crowd-powered Learning for Text Classification. *ACM Trans. Knowl. Discov. Data.* 16, 1, Article 15 (June 2021), 23 pages.
https://doi.org/10.1145/3457216

1 INTRODUCTION

Text classification (a.k.a. text categorization or text tagging) [22] is a key content analysis task that has received much attention from both academia and industry. It has a wide range of real-life applications such as sentiment analysis [1], spam detection [61], topic labeling [17], and intent detection [26], to name but a few. For example, (i) Merchants detect and capture consumer preferences by listening to consumers' reviews; (ii) E-mail service providers analyze email contents in order to filter spam.

This work was supported in part by the National Key R&D Program of China under Grant No. 2018YFB1004003 and the NSFC under Grants No. 62025206 and 61972338.
Authors' addresses: K. Yang, Y. Gao (corresponding author), L. Liang, S. Bian, and L. Chen, College of Computer Science, Zhejiang University, 38 Zheda Road, Hangzhou 310027, China; emails: {kyyang, gaoyj, leiliang, songbian, luchen}@zju.edu.cn; B. Zheng, Singapore Management University, 178902 Singapore; email: bhzheng@smu.edu.sg.

1556-4681/2021/06-ART15 $15.00
https://doi.org/10.1145/3457216

In recent years, deep neural network models [9, 10, 44, 66, 67] have achieved state-of-the-art performance in text classification. They learn to represent a text with an implicit feature vector, and feed the vector into a *softmax* function to calculate the probability of each class label for the given text. **Recurrent Neural Network** (**RNN**), **Convolutional Neural network** (**CNN**), and Transformer are the representative neural network architectures used to represent the text.

Those deep models learn the representation from all the words in text without relying on or considering any prior knowledge or pre-existing guidance. Nevertheless, it is not a secret that certain words in text are more important than the others in terms of text classification, and signals from some words provide an explicit indication about the class label. For instance, words that express *happiness*, *excitement*, *hope*, and *inspiration* correspond to the *positive* emotional category; on the other hand, words that express *fear*, *anger*, *sadness*, and *disgust* are associated with the *negative* emotional category. This inspires us to design a new approach which can fully utilize those important *keywords* to guide the training of the deep neural networks. We expect the neural networks equipped with the useful prior knowledge could further improve the accuracy of text classification.

Many *keyword extraction* approaches [16, 40] have been proposed to automatically extract keywords in text using machine learning algorithms. Although considerable studies have been devoted to keyword extraction over the years, the task of extracting relevant keywords with high quality is far from being solved. Human beings are more capable than machine learning algorithms in terms of capturing keywords in text. Moreover, the development of crowdsourcing platforms, such as **Amazon Mechanical Turk** (**AMT**)[1], Figure Eight[2], and Upwork[3], makes it easier for individuals and businesses to solicit human intelligence for machine-hard problems. The keywords could be extracted from text with the help of *the crowd workers* (i.e., thousands of ordinary workers in the crowdsourcing platform). Motivated by this, we propose *Crowd-powered learning for Text Classification* (*CrowdTC* for short) in this article. To our knowledge, CrowdTC is the first attempt to use crowdsourcing platforms to efficiently extract and utilize the keywords to guide the deep neural networks for text classification. Towards this, there are two main challenges to be addressed.

The first challenge is *how to design cost-efficient crowdsourcing questions to capture the human being's guidance?* For text classification, we could consult the crowd workers for every single text exhaustively. Nonetheless, the brute-force method is time-consuming and expensive. Even if the consulting fee is one penny each record, it would cost ten thousand dollars for a text dataset of one million records. To this end, we introduce the concept of *keyword*, which refers to the word that has a greater impact on the classification than the other words, in the text, and only ask the crowd workers to identify keywords from the sampled text dataset. Then, we expand the keyword set based on clustering in the *word embedding space*, utilizing the fact that those words belonging to the similar semantic categories are proximal to each other in the word embedding space [45].

The second challenge is *how to incorporate the extracted keywords as guidance into deep neural networks?* Deep neural networks are notorious for the un-interpretability. Thus, it is intractable to feed the external guidance into deep neural networks. Towards this, we design two types of neural networks, namely, *KA-RNN* and *HDNN*, to embrace the extracted keywords. KA-RNN is an attention-based RNN model whose *loss function* has been customized to emphasize the keyword signals. **HDNN** is a **hybrid deep neural network** that combines a standard CNN (or RNN) with a **Fully Connected Network** (**FCN**) to integrate the information of original text with the keyword signals. To sum up, this article makes the following four key contributions.

[1] https://www.mturk.com.
[2] https://www.figure-eight.com.
[3] https://www.upwork.com.

—We present CrowdTC. To the best of our knowledge, it is the first attempt to utilize the keywords extracted from the crowd workers to improve the performance of deep learning for text classification.
—We design a crowd-powered framework to capture high-quality human being's guidance in the form of keywords with a low monetary cost. In the framework, we utilize the proximity of similar semantic words in the embedding space, and then employ sampling and clustering techniques to reduce the cost, and meanwhile retain the performance.
—We propose two different models, i.e., KA-RNN and HDNN, to incorporate the extracted keywords into deep neural networks. KA-RNN redesigns the loss function of attention-based RNN to emphasize the keyword signals. HDNN builds a hybrid deep neural network that combines the standard CNN (or RNN) with FCN to enable the fusion of original text and keyword information.
—We conduct extensive experiments to demonstrate the effectiveness of our proposed techniques, and show that our CrowdTC models could improve the accuracy of neural networks for text classification based on the crowd-powered keyword guidance.

The remainder of this article is organized as follows. We first review related work in Section 2, and introduce the background of text classification and deep neural networks in Section 3. We then elaborate the framework of CrowdTC in Section 4, present the cluster-based crowdsourcing in Section 5, and detail KA-RNN and HDNN models in Section 6 and Section 7, respectively. Experimental evaluation is reported in Section 8. Finally, we conclude the article in Section 9.

2 RELATED WORK

In this section, we overview the existing studies in *text classification*, *keyword extraction*, and *crowdsourcing*.

2.1 Text Classification

Existing studies on text classification can be categorized into two groups, including *feature-based models* and *deep learning models*.

Traditional text classifiers are feature-based models, relying on handcrafted features to perform the classification. They represent a text as a sparse vector, and feed it into the classifier. Cavnar et al. [6] propose an N-gram-based approach for text classification. Bag-of-words [52] is another efficient way to extract the features. Post and Bergsma [43] exploit more complex features such as *POS tagging* and *dependency parsing* to improve the performance of text classification. Naïve Bayes, maximum entropy classification, and support vector machines are popular classifiers [39]. Nonetheless, feature-based models neglect the context of the text, and hence, cannot capture deep semantic information.

To overcome such an issue, deep learning models have become popular for this task. A comprehensive survey for deep learning-based text classification approaches could be found in [36]. Those models map a text to a dense vector to extract deep semantic information. RNN and CNN are two popular deep learning models. Kim [20] uses CNN to classify sentences by encoding a sentence with multiple convolutional filters. Zhang et al. [67] present an empirical exploration of CNN for text classification. Tang et al. [50] leverage **Long Short-Term Memory** (**LSTM**, a special kind of RNN) [18] to model the relation of sentences. Yang et al. [64] propose a hierarchical attention RNN to better capture the important information of a document. Conneau et al. [9] use very deep CNN in text classification, which achieves good performance. Yogatama et al. [66] present a discriminative LSTM model to place documents in the semantic space, such that embedding of documents are close to embedding of their respective labels. Qiao et al. [44] propose a method of learning and utilizing task-specific distributed representations of N-gram for text

classification. Besides, some studies attempt to combine CNN with RNN. Lai et al. [23] propose RCNN, which captures contextual information with the recurrent structure and constructs the text representation by a CNN. Wang et al. [56] propose a regional CNN-LSTM model to compute valence-arousal ratings from texts for dimensional sentiment analysis. Xiao and Cho [62] utilize both convolution and recurrent layers to efficiently encode character inputs. Wang [54] turns to a large window size of CNN to capture long-term dependencies, and uses more general recurrent units to achieve better performance. Recently, Devlin et al. propose BERT [10], which uses a novel transformer architecture to pre-train the text representations for downstream tasks, such as text classification. Yang et al. further improve BERT, and propose XLNet [63] with the autoregressive formulation. Even though the transformer-based architecture achieve state-of-the-art results on many natural language processing tasks, it requires a large corpus and huge computation resource (e.g., it needs at least 4 days in 16 TPU chips to complete the pre-training of BERT [10]). Although those deep neural network models achieve state-of-the-art performance for text classification, they do not make full use of the important information carried by individual keywords.

Another related line of work is keyword-driven text classification. Eschewing the need of labeled texts, they provide a weakly-supervised schema for text classification based only on keyword-level description of each category. Song and Roth [48] consider the labels as seeds, and classify the text by embedding both labels and documents in the same semantic space. Similarly, Wang et al. [55] propose an attention work that measures the compatibility of embeddings between documents and labels. Meng et al. [32] leverage seed keywords to generate pseudo documents and refine the model through a self-training module that bootstraps on unlabeled documents. This method is later extended to handle hierarchical text classification [33]. Mekala et al. [29] further present a contextualized weakly supervised classification framework. Recently, Meng et al. [34] utilize BERT to learn the semantics of label names for text classification. Different from the above approaches, in this article, we consider both the extracted keywords from the crowd workers and the labeled text training samples to build the supervised deep learning model with the human guidance for text classification.

2.2 Keyword Extraction

Many existing keyword extraction approaches try to extract keywords from the text automatically by using machine learning algorithms. They can be clustered into two categories, i.e., *unsupervised* approaches and *supervised* approaches.

Among unsupervised approaches, TFIDF [19] is a classic keyword extraction method. It measures the importance of a word by comparing the frequency of the word in a text to its frequency in a large corpus. Several alternative approaches have been proposed, such as KP-Miner [11], RAKE [46], SBKE [2], and YAKE [4]. They further consider other statistical factors (i.e., word length, word position, etc.) to determine the importance of a word. In addition to the above statistical methods, graph-based methods have also been proposed, such as TextRank [35], SingRank [53], ExpandRank [53], and TopicRank [3]. They represent the input text as a graph, and rank its nodes according to their scores using graph-based ranking methods to sort the importance of keywords by order.

Next, we introduce the supervised keyword extraction approaches. KEA, one of the most widespread algorithms, Witten et al. [59]. It inputs two features (i.e., TFIDF and the term's first occurrence) into the Naïve Bayes algorithm to determine whether a word is a keyword or not. It is further improved by several follow-up studies [28, 38]. Caragea at al. [5] try to leverage neighborhood information to extract keyword. Meng et al. [30] use a deep neural network model to predict keywords for scientific text. Besides, there are several methods could serve as keyword extraction

alternatives, such as topic modeling-like methods [14, 31]. More detailed and in-depth reviews for keyword extraction could be found in [16, 40].

Although considerable research has been devoted to this topic, the task of extracting relevant keywords with high accuracy is far from being solved. In this article, we design a crowd-powered framework to extract high-quality keywords with a low monetary cost.

2.3 Crowdsourcing

In addition to text classification, many important data management and analysis tasks cannot be completely addressed by automated processes [24]. Crowdsourcing is an effective technique to harness the capabilities of people (i.e., the crowd workers) to apply human computation for such tasks. The development of crowdsourcing platforms makes it an active research area in the data management community.

Those crowdsourcing platforms allow computer scientists to integrate the power of human intelligence into their computational workflows. Take query processing as an example. Many crowds-based query processing systems have been implemented, such as CrowdDB [13], Qurk [27], and Deco [41]. They use optimization techniques to reduce the number of questions the crowd workers have to answer. In the field of image recognition, Welinder and Perona [58] propose a crowd-based algorithm to determine the ground truth for images from noisy annotations. For entity resolution, Vesdapunt et al. [51] study the problem of completely resolving an entity graph using crowdsourcing; Wang et al. [57] present ACD, a crowd-based algorithm for data deduplication, which achieves high accuracy at moderate costs of crowdsourcing. Besides, many studies [25, 37] apply active learning techniques to reduce the crowdsourcing cost for collecting and annotating data. Nevertheless, the above methods are applied-dependent, and thus, they cannot be directly applied to text classification.

In this article, we extract the keywords in text via the crowdsourcing platform, and leverage the extracted keywords to guide deep neural networks to perform the task of text classification.

3 PRELIMINARIES

In this section, we formally introduce *text classification* and two representative types of *deep neural networks*.

3.1 Text Classification

Text classification (a.k.a. text categorization or text tagging) is the task of assigning a set of predefined categories to text. As an example, given a text about the movie review below:

"It's easily the best film I've seen this year, the story of the film is pretty great."

A machine learning model (classifier) could take this text as input, analyze the content, and assign the sentimental polarity (class), i.e., *positive*, to this text.

3.2 Deep Neural Networks

In the following, we introduce the two representative types of deep neural networks, i.e., RNN and CNN.

3.2.1 Recurrent Neural Network. RNN is a type of neural networks that conditions the model on all previous words in the corpus. Figure 1 illustrates the RNN architecture where each rectangular box represents a hidden layer at a time step t. Each such layer holds a number of neurons, and performs a weighted sum operation on its inputs followed by a non-linear activation operation (such as $tanh()$, $sigmoid()$, and $ReLU()$). At each time step t, the output of the previous step h_{t-1}

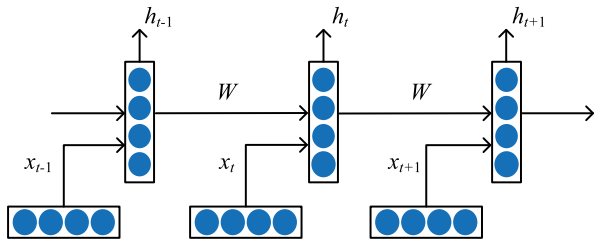


Fig. 1. Illustration of RNN.

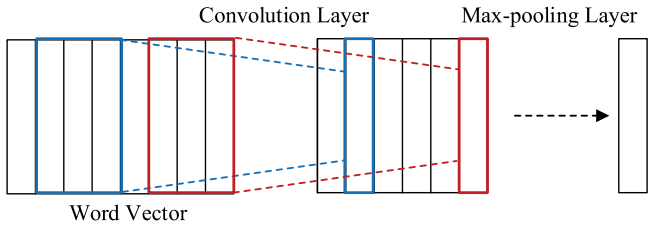


Fig. 2. Illustration of CNN.

and the next word embedding vector x_t in the text will be input to the hidden layer to conduct the hidden representation h_t in step t as follows:

$$h_t = \sigma\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right),$$

where $\sigma()$ is a non-linear activation function and both $W^{(hh)}$ and $W^{(hx)}$ are the weight matrices.

We can observe that the hidden representation of step t depends upon all the previous input vectors. The t-th step state can be expressed by:

$$h_t = RNN(x_t, x_{t-1}, \ldots, x_1).$$

The output of the hidden state in the last step could represent the text, and hence could be the indicator of text classification.

Hidden states at each step depend on all the previous inputs, but sometimes neglect the key information, which might hurt the overall performance of the classifier [65]. Gating mechanisms have been developed to address the limitation of RNN, resulting in two prevailing RNN types, i.e., LSTM [18] and **Gated Recurrent Unit (GRU)** [8]. Both LSTM and GRU can perform text classification, but we use GRU as the default RNN unit (detailed in Section 6), because GRU is faster to train and more suitable for processing large-scale data.

3.2.2 Convolutional Neural Network. Unlike RNN that models the whole sequence and captures the long-term dependencies, CNN is a class of neural networks that extracts local and position-invariant features. Figure 2 depicts the CNN architecture. CNN takes word vectors, i.e., d-dimensional dense vectors, as input. It uses the convolution layer to represent learning from sliding w-grams. For an input sequence with n word vectors, $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$, let vector $c_i \in \mathbb{R}^{wd}$ be the concatenated embeddings of w entries, i.e., $x_i, x_{i-1}, \ldots, x_{i-w+1}$, where w is filter width and $w \leq i \leq n$. The convolution layer generates the representation $p_i \in \mathbb{R}^d$ for the w-gram $x_i, x_{i-1}, \ldots, x_{i-w+1}$ using the convolutional weights $W \in \mathbb{R}^{d \times wd}$:

$$p_i = \sigma\left(Wc_i + b\right)$$

where $\sigma()$ is a non-linear activation function and $b \in \mathbb{R}^d$ is the bias.

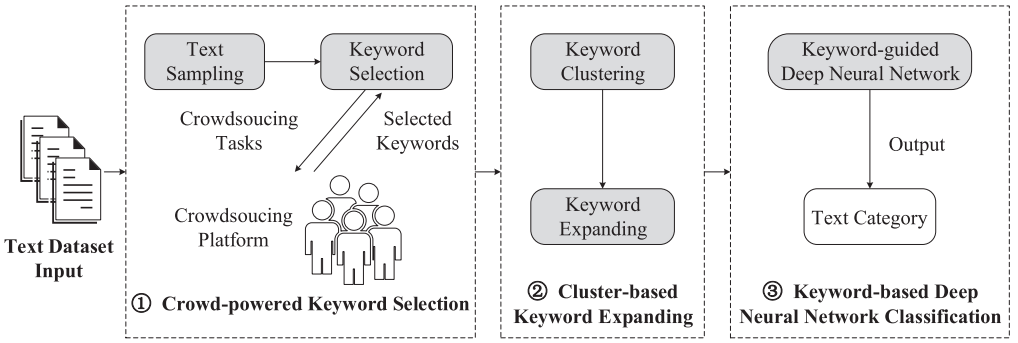


Fig. 3. The Framework of CrowdTC.

After the convolution layer, it uses max-pooling layer to extract the main information. For all w-gram representations p_i, a hidden representation $h \in \mathbb{R}^d$ is generated by max-pooling: for each element h_j of h, $h_j = \max(p_{1,j}, p_{2,j}, \ldots), j = 1, \ldots, d$. That is to say, the max-pooling layer generates the hidden representation h_j by extracting the maximum value in each dimension of all the representations p_i. The hidden features could represent the text, and be the indicator of text classification.

4 OVERVIEW OF CROWDTC

In this section, we overview the framework of our proposed CrowdTC . As shown in Figure 3, CrowdTC consists of three main stages in the following.

—**Stage 1.** Crowd-powered Keyword Selection.
—**Stage 2.** Cluster-based Keyword Expanding.
—**Stage 3.** Keyword-based Deep Neural Network Classification.

In the first stage, we try to select keywords in text by soliciting the human cognitive ability via the crowdsourcing platform. The keywords are the words in the text with a greater impact on the category of text than other words. In order to reduce the monetary cost of crowdsourcing, we sample the input texts, and ask the crowd workers to select the keywords in sampled texts, instead of consulting the crowd workers for every single text's classification.

In the second stage, we expand the selected keyword set by using the clustering method. We understand the side-effect of sampling as it might reduce the positive impact of keywords on classifier's accuracy. In order to compromise the negative impact of sampling, we adopt a clustering approach to expand the keywords as a remedial action. The inspiration behind is that those words that have similar semantic information are expected to be located in a neighboring area after being mapped into the word embedding space [45]. Thus, we use the classic clustering method to capture word clusters in the word embedding space, and expand the keyword set based on those word clusters. The keywords are extracted in the first and second stages. The details will be introduced in Section 5.

In the last stage, the extracted keywords are utilized as human guidance to guide deep neural network training. In order to incorporate the human guidance (i.e., keywords) into deep neural networks, we design two different neural network models for text classification, i.e., KA-RNN and HDNN, to embrace the extracted keywords. KA-RNN redesigns the *loss function* for the attention-based RNN model to emphasize the keyword signals. HDNN is a hybrid deep neural network model that combines the standard CNN (or RNN) with FCN to fuse the original text information and keyword signals. We will detail the two models in Section 6 and Section 7, respectively.

5 CLUSTER-BASED CROWDSOURCING

In this section, we first explain how to consult the crowd workers to select keywords in sampled dataset, and then, we describe how to expand the keyword set through the clusters in the word embedding space.

5.1 Sampling and Crowdsourcing

As stated in Section 1, it is impossible and unaffordable to ask the crowd workers to help in classifying *each single* text in large corpus. To tackle this issue, we use the concept of *keyword*, which refers to a word in the text that is informative and has a greater impact on text classification than other words. We extract the keywords and utilize them to guide the deep neural networks for text classification.

Accordingly, we consult the crowd workers for the keywords in the corpus on the crowdsourcing platform. We adopt the sampling approach to sample only a small portion of large corpus to further reduce the monetary cost. Specifically, we post the crowdsourcing tasks on AMT, and select at least three keywords per sampled text from the crowd workers. As to be presented in Section 8, our proposed method could achieve good performance even if only 1‰ of the corpus is sampled.

To be more specific, the crowdsourcing task in this article is to ask the crowd workers to choose at least three keywords from each given sampled text. For instance, recall that the text example with the positive label presented in Section 3.1 as follows:

"It's easily the best film I've seen this year, the story of the film is pretty great."

The crowdsourcing task aims to consult the crowd worker with the request "Please choose at least three words that have greater impact on the positive label of this sentence". Then, the crowd worker would reply with three words "*best*, *pretty*, and *great*". Unlike the other words, in the specified text, the three words indicate clearly the positive label for the text, and can be employed to guide the training of deep neural networks.

Based on the above processing, we are able to collect the set of keywords from the sampled text via the crowdsourcing platform AMT. Next, we introduce how to expand the keywords to guide text classification.

5.2 Keyword Clustering

The main reason that the small sample size does not deteriorate the performance of our model is that we expand the small keywords set contributed by the crowd workers via clustering. The effectiveness of clustering is guaranteed by the fact that words sharing similar semantic meanings are expected to be close to each other in the word embedding space [45].

For illustration purposes, we adopt **principal components analysis** (**PCA**) to convert the word embedding vector from a high-dimensional space to a two-dimensional space, and visualize a word embedding space example in Figure 4. It is observed that the vectors of the words with similar semantic meanings are close to each other. Take words *unfortunately* and *disappointed* that have negative sentiment polarity as an example. They are close and located on the lower left in Figure 4. Inspired by this observation, we adopt classic clustering methods [15] to capture word clusters in the word embedding space and to find the hidden keywords based on the clustering result with the help of the keywords contributed by the crowd workers.

Next, we show how to identify the hidden keywords based on the clusters. The main idea is to use the keywords selected from the crowd workers as the seeds to identify other keywords based on the clustering. To be more specific, we call keywords identified by the crowd workers as *seed keywords*, and perform clustering based on seed keywords. We choose clusters that contain at

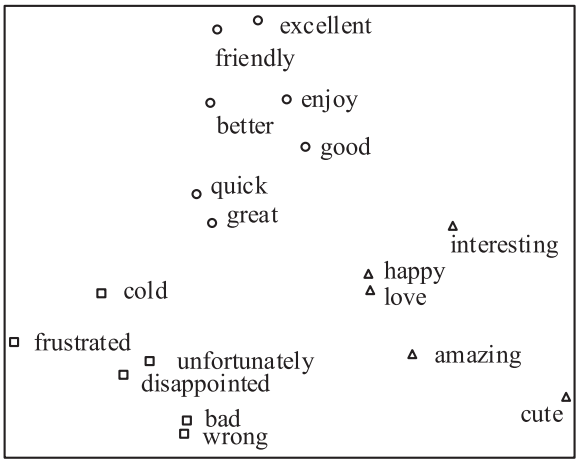


Fig. 4. A Case for the Word Embedding Space.

ALGORITHM 1: Keyword Expanding Method (KEM)

Input: an embedding vector set V for all the words, an embedding vector set S for the seed keywords

Output: the expanded keyword embedding vector set K

1 $K \leftarrow \emptyset$
2 $\cup C_i \leftarrow \text{Cluster}(V)$
3 **for** *each* C_i *in* $\cup C_i$ **do**
4 **if** $C_i \cap S \neq \emptyset$ **then**
5 $K \leftarrow K \cup C_i$
6 **return** the expanded keyword embedding vector set K

least one seed keyword as *keyword clusters*, and expand the keyword set based on those keyword clusters.

Consider the embedding space depicted in Figure 4 again. We could use the clustering method to group those words into three clusters, which are depicted as the circles in the upper region, the squares in the lower left region, and the triangles in the lower right region, respectively. Assume that the crowd workers selects *excellent*, *cold*, and *interesting* as seed keywords. All three clusters are labeled as keyword clusters, and we could include all the keywords into the expanded keyword set. In other words, we expand the keyword set which contains 3 seed keywords to an expanded keyword set of 18 keywords.

Based on these, we present ***Keyword Expanding Method*** (***KEM***), with its pseudo-code depicted in Algorithm 1. It takes as inputs an embedding vector sets V for all the words and an embedding vector set S for seed keywords, and outputs the expanded keyword embedding vector set K. First, KEM initializes K to an empty set (line 1). Then, it clusters the vector set V, with the resulting clusters preserved by $\cup C_i$ (line 2). Next, for each cluster $C_i \in \cup C_i$, KEM includes it into K if it contains at least one seed keyword (lines 3–5). After evaluating all the clusters, KEM returns the expanded keyword set K to complete the process (line 6).

Upon the completion of the first two stages of CrowdTC, we generate an expanded keyword set, which is ready to be fed into deep neural networks to guide text classification. Next, we address the

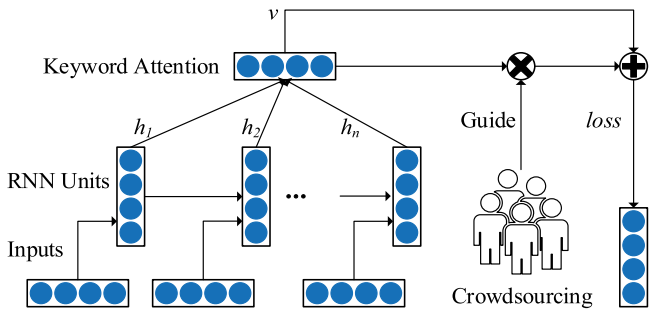


Fig. 5. The Architecture of KA-RNN.

second challenge, which is how to incorporate human intelligence into the deep neural networks for text classification. In Section 6 and Section 7, we propose two deep neural network models, respectively.

6 KEYWORD-BASED RNN WITH ATTENTION MECHANISM

In this section, we introduce the first newly proposed deep neural network, which is a kind of RNN that uses the attention mechanism [64] and takes the keyword extracted from the crowdsourcing platform into consideration. We call the RNN based on Keyword and Attention mechanism as KA-RNN. Here, we give the detail of KA-RNN.

KA-RNN emphasizes the keyword signals by enabling keyword to play a greater impact on the attention weight. The structure of this model is shown in Figure 5. It contains three parts, namely, a *word embedding input layer*, a *standard RNN layer*, and an *attention layer*. It uses the keywords that represent the human intelligence to guide the weight training of the attention layer, and combines the output in a fully connected layer to aggregate the loss.

6.1 Standard RNN

In the standard RNN layer, we use GRU [8] to construct RNN. GRU employs a gating mechanism to capture potential long-term dependencies. The gating mechanism can control the flow of information, and mitigate the gradient vanishing problem. There are two types of gates in GRU, i.e., the *reset gate* r_t and the *update gate* z_t. They control together, how a hidden state is updated. At time step t, GRU computes h_t as follows:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The computation is a linear combination of the previous state h_{t-1} and the current new state $\tilde{h}_t$ that is derived from new input information, where $\odot$ is the element-wise multiplication. The update gate z_t decides how much past information shall be forgotten, and how much new information shall be considered. z_t is computed as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z),$$

in which x_t is the input vector at time t, W_z and U_z are the weight parameters, and b_z refers to the bias. The candidate new state $\tilde{h}_t$ is computed in a way similar as a traditional RNN:

$$\tilde{h}_t = \sigma(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h).$$

where r_t is the reset gate which controls how much the previous state contributes to the candidate new state, and U_h is the weight matrix for h_{t-1}. Similar to the update gate, r_t is computed as

follows:

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right)$$

in which W_r and U_r are the weight parameters, and b_r is the bias.

6.2 Attention Mechanism

Every word in text contributes differently to the representation of text. Standard RNN cannot differentiate the important words from the rest of the input text for text classification. As a solution, we introduce the attention mechanism to extract important words, and describe how text classification can take advantage of the keywords extracted by a cluster-based crowdsourcing.

Let d-dimensional vectors $h_1, h_2, \ldots, h_n$ denote the hidden representations produced by the standard RNN from the original input text, where d is the size of the hidden layers and n is the length of the input text. The attention mechanism will produce an attention weight vector α and a weighted hidden representation v. Specifically,

$$u_i = \sigma\left(W_w h_i + b_w\right)$$

$$\alpha_i = \frac{\exp\left(u_i^\top u_w\right)}{\sum_i \exp\left(u_i^\top u_w\right)}$$

$$v = \sum_i \alpha_i h_i$$

where W_w and b_w are the weight matrix and bias parameters, respectively.

We first feed word hidden representation h_i using a non-linear active function to get u_i. Then, we measure the importance of the word as the similarity between u_i and a word-level context vector u_w, and get a normalized importance weight α through a softmax function. The word context vector u_w is randomly initialized and learned during the training process. After the attention weight vector α is produced, the vector v is computed to summarize all the information of the input text. v is considered as the feature representation of the input text. Then, a softmax function is to transform v to a conditional probability distribution,

$$p = softmax\left(W_t v + b_t\right)$$

In which W_t and b_t are the parameters of softmax function. The attention weight vector α can be seen as a high-level representation of the question "which is the informative word?". When the word in the i-th state has a greater impact on the text classification than the word in the j-th state, α_i would be larger than α_j.

We could redesign the neural network and emphasize the keyword signals by using the attention weight vector α. Here, the important design criterion is that the weight of the extracted keyword should be larger than that of others. In other words, the keyword signals should be effectively amplified. In view of this, we design a new loss function for KA-RNN as follows:

$$loss = -\sum_d \log p_{d,j} - \lambda \sum_d m_d^T \alpha_d$$

where d refers to the input text, j is the label of text d, $\lambda > 0$ is a penalty coefficient, and m_d is a mask vector to indicate whether the current state in text d is an extracted keyword or not, i.e.,

$$m_i = \begin{cases} 1, & \text{if the } i\text{-th state inputs a extracted keyword} \\ 0, & \text{otherwise} \end{cases}$$

The loss function contains two parts: (i) the cross-entropy error between p and the class label, and (ii) the regularization term for the attention weights. Since the goal of training is to minimize the

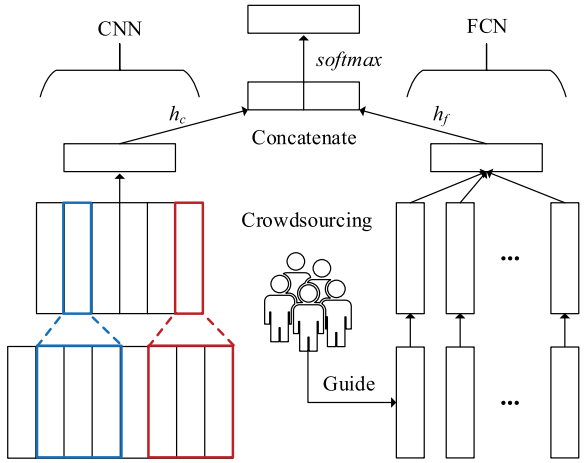


Fig. 6. The Architecture of HDNN.

loss function, the attention weight of the extracted keyword, i.e., the keyword signal, tends to be amplified during the training process.

7 HYBRID DEEP NEURAL NETWORK

In this section, we present the second newly proposed deep neural network structure, which integrates the original text and the keyword information. It is a HDNN that merges a standard CNN (or RNN) with an FCN. For the sake of brevity, we focus on the version of HDNN that is based on CNN and FCN with its architecture illustrated in Figure 6. Note that CNN in HDNN could be replaced by RNN.

HDNN takes as inputs both the original text and the keywords extracted by a cluster-based crowdsourcing, and outputs the predicted class label. HDNN consists of two main components, a CNN and an FCN. It relies on the CNN to capture the hidden representation vector for the original text, and meanwhile, it invokes the FCN to obtain the hidden representation vector for the extracted keywords. Then, it concatenates the two representation vectors to seamlessly fuse the original text information and human intelligence to effectively enhance the accuracy of the classification task. Finally, HDNN uses a softmax output layer to generate the probability for each class label. In the following, we give the details of HDNN.

7.1 Standard CNN

Standard CNN is a key component of HDNN, as shown in the left of Figure 6. It takes the original text x as an input. Text x contains n words, with each corresponding to a d-dimensional word embedding vector. Thus, the input word embedding layer contains a feature map of $d \times n$ size. Next, it is the convolution layer which is used to generate the hidden representation from sliding w-grams.For the input word embedding with n vectors, i.e., $x_1, x_2, \ldots, x_n$. let vector $c_i \in \mathbb{R}^{wd}$ be the concatenated embeddings of w entries, i.e., $x_i, x_{i-1}, \ldots, x_{i-w+1}$, where w is the filter width and $w \leq i \leq n$. The convolution layer generates the representation $p_i \in \mathbb{R}^d$ for the w-gram $x_i, x_{i-1}, \ldots, x_{i-w+1}$ using the convolutional weight $W \in \mathbb{R}^{d \times wd}$:

$$p_i = ReLU\left(Wc_i + b\right),$$

where $b \in \mathbb{R}^d$ represents the bias, and $ReLU$ is a type of active function:

$$ReLU : f(z) = max(0, z).$$

After the convolution layer, a max-pooling layer is used to capture the main information. For all w-gram representations p_i, a hidden feature $h_c \in \mathbb{R}^d$ is generated by max-pooling. For each element $h_{c,j}$ of h_c, the max-pooling layer extracts the the maximum value in each dimension of all the representations p_i:

$$h_{c,j} = \max\left(p_{1,j}, p_{2,j}, \ldots\right) (j = 1, \ldots, d)$$

The hidden feature h_c could be seen as the high-level representation of the original text.

7.2 Fully Connected Network

The other main part of HDNN is an FCN, as depicted in the right part of Figure 6. After cluster-based crowdsourcing, we feed the keywords extracted by a cluster-based clustering into the FCN to capture the representation of the keywords and guide text classification.

Suppose the number of the keywords that fed into HDNN is s. For the input keyword embedding vectors, i.e., x_1, x_2, ..., x_s, the fully connected layer firstly transforms them into the hidden representations. To be more specific, for each keyword embedding x_i $(i = 1, 2, \ldots, s)$, the fully connected layer computes the hidden representation h_i as follows:

$$h_i = ReLU\left(W_f x_i + b\right)$$

where $W_f \in \mathbb{R}^{d \times d}$ is the sharing weight matrix in the fully connected layer, $b \in \mathbb{R}^d$ is the bias, and $ReLU$ is the type of active function:

$$ReLU : f(z) = max(0, z)$$

After the fully connected layer, the max-pooling layer is again used to capture the main information. For each hidden representation h_i $(i = 1, 2, \ldots, s)$, we capture the maximum value in each dimension of all the h_i, and construct an d-dimensional vector h_f to represent the keyword information. The vector h_f is generated by:

$$h_{f,j} = \max\left(h_{1,j}, h_{2,j}, \ldots\right) (j = 1, \ldots, d)$$

The hidden feature h_f captures the information of extracted keywords.

7.3 Concatenation in HDNN

We have generated two different representations from two parts of HDNN to perform text classification. One is the output of CNN, which captures the information of the original text. The other is the output of FCN, which captures the key signals from the extracted keywords. Next, we introduce how to fuse these two representations for the final classifier.

Given the the output vector h_c from the CNN, and the output vector h_f from the FCN, $h_c, h_f \in \mathbb{R}^d$, we use another fully connected layer to transform h_c and h_f into two $\frac{d}{2}$-dimensional vectors h_{ct} and h_{ft} respectively:

$$h_{ct} = (W_{ct} h_c + b)$$
$$h_{ft} = \left(W_{ft} h_f + b\right)$$

where $W_{ct}, W_{ft} \in \mathbb{R}^{d \times d/2}$, and bias $b \in \mathbb{R}^d$. The vector h_{ct} can be seen as the hidden representation of the information from the original text, and the vector h_{ft} carries the information from the extracted keywords.

Table 1. Statistics of the Datasets Used in Our Experiments

Dataset	Class Number	Train Samples	Test Samples	Average Length	Vocabulary
AG's news	4	120,000	7,600	34	62,978
Yelp.F	5	650,000	50,000	148	268,271
Yelp.P	2	560,000	38,000	146	246,577
Amazon.F	5	3,000,000	650,000	82	1,007,324
Amazon.P	2	3,600,000	400,000	80	1,058,969

We concatenate the two $d/2$-dimensional vectors in a d-dimensional vector h_t, which could serve as a high-level representation of the text classification with the guidance of human beings. Next, a softmax layer is followed to generate the probability distribution of predicted class labels:

$$p = softmax\left(W_t h_t + b_t\right)$$

where W_t and b_t are the parameters of the softmax function.

The model can be trained by backpropagation, in which the loss function is the cross-entropy loss. The loss function is computed as:

$$loss = -\sum_{d} \log p_{d,j}$$

in which d is index of input text, and j is the label of text d.

8 EXPERIMENTAL EVALUATION

In this section, we first present our experimental settings in Section 8.1, then conduct the experiments to evaluate the effectiveness of our proposed techniques in Section 8.2, and finally compare CrowdTC against state-of-the-art models in Section 8.3.

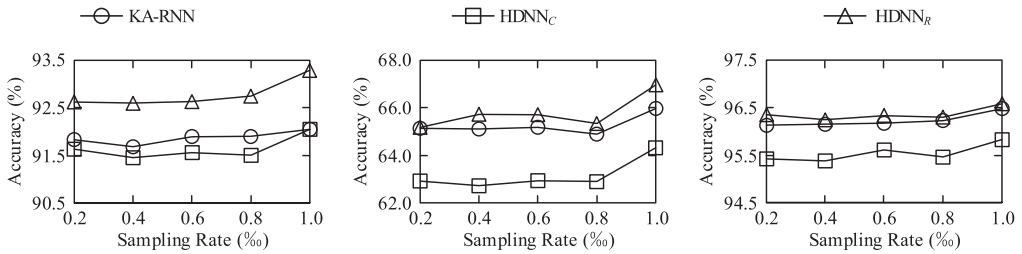
8.1 Experimental Settings

We use five large-scale public text datasets [67] in the experiments, with their statistics listed in Table 1. *AG's news* dataset is obtained from the AG's corpus of the news article. Top-four largest classes are extracted from this corpus, using only the title and description fields. The number of training samples for each class is 30,000, and the number of test samples for each class is 1900. The Yelp reviews dataset is obtained from the 2015 Yelp Dataset Challenge. Two classification datasets are constructed: one (denoted as *Yelp.F*) predicting the full number of stars given by the user, and the other (denoted as *Yelp.P*) predicting a polarity label by considering stars 1 and 2 as negative as well as 3 and 4 as positive. *Yelp.F* dataset has 130,000 training samples and 10,000 testing samples in every star, and *Yelp.P* dataset has 280,000 training samples and 19,000 test samples in each polarity. The Amazon reviews dataset is obtained from the **Stanford Network Analysis Project** (**SNAP**). Similar to the Yelp review dataset, two datasets have also been constructed: one (denotes as *Amazon.F*) is full score prediction, and the other (denotes as *Amazon.P*) is polarity prediction. *Amazon.F* dataset includes 600,000 training samples and 130,000 testing samples in every class, whereas *Amazon.P* dataset contains 1,800,000 training samples and 200,000 test samples in each polarity sentiment. Note that, the class distributions of the aforementioned datasets are balanced.

In our experiments, we sample 1‰ of the training datasets by default, and consult the crowd workers for the keywords that are important for text classification in AMT. In the neural network training processing, we utilize the 300D GloVe 42B vectors [42] as the pre-trained word embedding. Note that, we only retain the words having frequencies larger than a given threshold, and replace the rest with a special UNK token for the sake of efficiency. The threshold frequency and the

Table 2. Statistics of Filtered Datasets

Dataset	Threshold Frequency	Vocabulary
AG's news	0	62,978
Yelp.F	10	50,850
Yelp.P	10	47,085
Amazon.F	50	54,033
Amazon.P	50	55,908



(a) *AG's news* (b) *Yelp.F* (c) *Yelp.P*

Fig. 7. The Effect of Sampling Rate.

filtered vocabulary size of each dataset are reported in Table 2. Here, *AG's news* dataset is small, and thus, we retain the original dataset without filtering.

We implement three CrowdTC models, viz., *KA-RNN*, *HDNN$_C$*, and *HDNN$_R$*. KA-RNN is the model proposed in Section 6, HDNN$_C$ is one version of our proposed HDNN model discussed in Section 7, and HDNN$_R$ is the other version of HDNN model, which replaces the CNN architecture discussed in Section 7 with a standard RNN. For RNN architecture, we set the dimension of the hidden state in GRU to be 500 in small datasets (i.e., *AG's news*, *Yelp.F*, and *Yelp.P*), and 1000 in large datasets (i.e., *Amazon.F* and *Amazon.P*). For CNN architecture, we use four different convolutional filter sizes to extract features from text, and set the number of convolutionals filters to be 300. We use an Adam algorithm [21] to optimize all the trainable parameters, and apply the gradient norm clipping to tackle the issue of gradient explosion. The training batch size is set to be 128. Our CrowdTC models are implemented in Python 3.6 on Tensorflow 1.13, and accelerated by NVIDIA GeForce RTX 2080 Ti GPUs.

8.2 Performance Study

In this subsection, we conduct the first set of experiments is to evaluate the effectiveness of the techniques proposed in this article, including the effect of sampling rate for text datasets, the impact of different keyword expanding algorithms on the accuracy of the proposed models, the effect of cluster-based crowdsourcing, and the sensitivity of the length of FCN.

8.2.1 The Effect of Sampling Rate. As discussed in Section 5, we utilize the sampling technique to reduce the monetary cost in soliciting the crowd workers for the keywords. Figure 7 shows the accuracy of three CrowdTC models w.r.t. the sampling rate varying from 0.2‰ to 1‰ using *AG's news*, *Yelp.F*, and *Yelp.P* datasets.

The observation is that the accuracy of all the CrowdTC models slightly ascends with the growth of the text sampling rate. This is because that the larger sampling rate is, the more crowd-powered keywords can be selected by the crowd workers, which could bring more human intelligence

Table 3. Accuracy Scores (in Percentage) over Different Keyword Expanding Methods

	AG's news				
	k-means	Spectral	DBSCAN	Mean-shift	kNN
KA-RNN	91.36	91.24	91.27	**92.04**	91.61
HDNN$_C$	91.58	**92.04**	91.70	91.69	91.66
HDNN$_R$	92.76	92.54	**93.29**	93.29	92.46
	Yelp.F				
	k-means	Spectral	DBSCAN	Mean-shift	kNN
KA-RNN	64.84	**65.97**	65.03	65.08	61.81
HDNN$_C$	**63.52**	63.26	63.33	63.33	62.75
HDNN$_R$	65.60	64.38	**66.97**	65.01	66.01
	Yelp.P				
	k-means	Spectral	DBSCAN	Mean-shift	kNN
KA-RNN	95.73	96.18	**96.22**	96.12	95.56
HDNN$_C$	**95.83**	95.76	95.82	95.77	95.36
HDNN$_R$	96.24	96.33	**96.44**	95.41	96.28

guidance to neural networks. Moreover, the experimental results in Figure 7 demonstrate that our proposed CrowdTC models could achieve a steady increase in terms of accuracy even if only a small portion (less than 1‰) of text is sampled. The reason is that we adopt the keyword expanding approach to compromise the negative impact of sampling. Also, it is observed that HDNN$_R$ and KA-RNN models that contain RNN units perform better than HDNN$_C$ model that does not have RNN units. It implies that RNN is more suitable for taking advantage of extracting keywords than CNN.

8.2.2 The Selection of Keyword Expanding Methods. After the crowd-powered keyword selection, we employ the clustering method as an approach to expand the keyword set in order to guide the tuning of deep neural networks. This effectively reduces the number of questions that we have to consult the crowd workers, and thus, decreases the monetary cost of crowdsourcing.

In the following experiments, we sample (1‰ by default) the training dataset to consult the crowd workers for keywords. After that, we need to select a clustering method to expand the keyword set. We evaluate the performance of four popular and classic clustering methods, using three datasets (viz., *AG's news*, *Yelp.F*, and *Yelp.P*). Those four evaluated clustering methods could be grouped into two categories. One is centroid-based clustering methods, including k-means and Spectral clustering [47], and the other is density-based clustering methods, including DBSCAN [12] and mean-shift [7]. In addition, one may wonder whether the k-Nearest Neighborhood (kNN) method could be effective in expanding keywords. Thus, we get the k closest candidates to expand the keywords for each selected keyword from the crowd workers, and evaluate the performance as well. We optimize the parameters for those keyword expanding methods by using a grid search to provide competitive results. Table 3 lists the accuracy scores (in percentage) of our proposed models over different keyword expanding methods using *AG's news*, *Yelp.F*, and *Yelp.P* datasets. The results of the best method to expand keywords are given in bold.

Our experiment results show that the clustering method is generally better than kNN method. This implies that the clustering method is more suitable for capturing the hidden keywords. Besides, there is no single clustering method that can win out for all kinds of datasets. It once again verifies the "**No Free Lunch**" (**NFL**) theorem [60] in the machine learning area. At the same time,

Table 4. Accuracy and Macro-F1 Scores (in Percentage) for CrowdTC Models over k-means

	AG's news		*Yelp.F*		*Yelp.P*	
	Accuracy	Macro-F1	Accuracy	Macro-F1	Accuracy	Macro-F1
KA-RNN	91.36	91.55	64.84	64.28	95.73	95.64
HDNN_C	91.58	91.62	63.52	63.34	95.83	95.60
HDNN_R	92.76	92.78	65.60	65.55	96.24	96.25

we can observe that there is no remarkable difference between the results generated by different clustering methods. In other words, the cluster-based crowdsourcing is not sensitive to the underlying clustering algorithm, and it could achieve satisfactory performance even by using a simple clustering method, e.g., k-means.

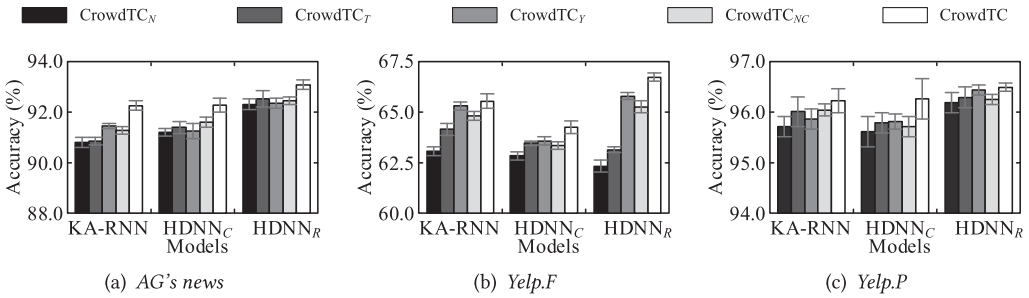
In addition, one may wonder about the performance of three CrowdTC models in the metric of macro-F1 scores. Thus, in Table 4, we additionally report the accuracy and macro-F1 scores (in percentage) for CrowdTC models over k-means clustering method using *AG's news*, *Yelp.F*, and *Yelp.P* datasets. We can observe that the performance in the accuracy scores is consistent with that in the macro-F1 scores. The reason is that the class distributions of the used datasets are balanced. The accuracy scores can blend recall and precision well. Therefore, we only use the metric of accuracy scores to evaluate the performance in the rest of our experiments.

8.2.3 The Effect of Cluster-based Crowdsourcing. In order to verify the utility of keywords and the performance of cluster-based keyword expanding, we, in addition to CrowdTC models proposed in this article, implement four variants of CrowdTC, denoted as *CrowdTC*$_N$, *CrowdTC*$_T$, *CrowdTC*$_Y$, and *CrowdTC*$_{NC}$, respectively. Here, we introduce the four additional variants of CrowdTC.

—**CrowdTC**$_N$ refers to CrowdTC *without keywords*, i.e., we do not consider the guidance of keywords for text classification.
—**CrowdTC**$_T$ **and CrowdTC**$_Y$ refer to CrowdTC *with keywords selected by two state-of-the-art automatic keyword extraction algorithms, i.e., TFIDF [19] and YAKE [4], respectively*, but not the human crowd. We use TFIDF and YAKE to pick the top-10 most important keywords for each single text in the training dataset without sampling.
—**CrowdTC**$_{NC}$ refers to CrowdTC *without clustering*. It still approaches the crowd workers to help identify the keywords from the sampled text dataset, but it does not adopt clustering algorithms to expand the keywords.

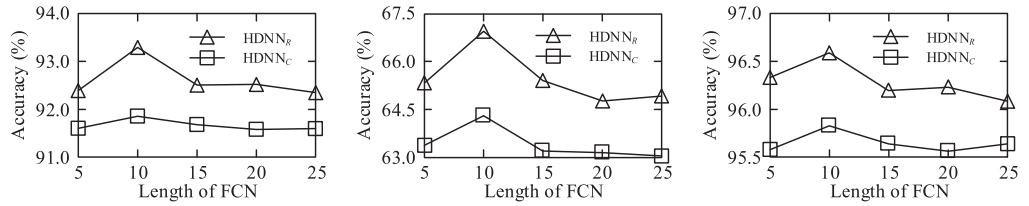
The comparisons of the five versions of CrowdTC are depicted in Figure 8, using *AG's news*, *Yelp.F*, and *Yelp.P* datasets. To alleviate the randomness, we run the models 5 times, and show the performance with average and standard deviation values (represented as error bars) in Figure 8. In general, the models with keywords, either returned by automatic keyword extraction algorithms (i.e., CrowdTC$_T$ and CrowdTC$_Y$) or identified based on the crowd workers (i.e., CrowdTC$_{NC}$ and CrowdTC), outperform the version without keywords (i.e., CrowdTC$_N$). It is worth noting that, CrowdTC improves the accuracy by 1.45% on average compared with the model without keywords (i.e., CrowdTC$_N$). This effectively demonstrates the positive impact of the keywords on the accuracy of text classification.

When we compare the performance of CrowdTC$_{NC}$ against that of CrowdTC$_T$ and CrowdTC$_Y$, it is observed that CrowdTC$_{NC}$ achieves the comparable accuracy with the other two automatic keyword extraction algorithms TFIDF and YAKE. Note that, CrowdTC$_{NC}$ only utilizes the keywords selected by the crowd workers for 1‰ of the texts in the corpus, while CrowdTC$_T$ and



(a) *AG's news* (b) *Yelp.F* (c) *Yelp.P*

Fig. 8. The Effect of Cluster-based Crowdsourcing.



(a) *AG's news* (b) *Yelp.F* (c) *Yelp.P*

Fig. 9. The Effect of FCN's Length in HDNN.

CrowdTC$_Y$ use all the keywords extracted by TFIDF and YAKE for every text in the corpus. That is to say, the quality of keywords plays a more important role in affecting accuracy than the number of keywords.

Consistent with our expectation, CrowdTC performs the best in all nine cases. This signifies that when the keywords are able to reflect the information of text accurately, the number of keywords starts to impact the performance positively. The larger the number of properly selected keywords, the higher the accuracy of the classification task. Besides, CrowdTC could effectively expand the proper keywords by using the clustering method.

8.2.4 The Effect of FCN's Length in HDNN. As discussed in Section 7, HDNN contains two main parts: CNN (or RNN) and FCN. The CNN (or RNN) part captures the information from the original text, while the FCN part captures the extracted keyword signals. The length of FCN is equivalent to the number of extracted keywords that are input to HDNN. We further evaluate the effect of FCN's Length for the performance of HDNN. Figure 9 illustrates the accuracy of HDNN models w.r.t. the length of FCN varying from 5 to 25 on *AG's news*, *Yelp.F*, and *Yelp.P* datasets.

The first observation is that, as expected, HDNN$_R$ outperforms HDNN$_C$, meaning that RNN is more suitable than CNN to benefit from the extracted keywords. Also, we can observe that the accuracy of HDNN (both HDNN$_C$ and HDNN$_R$) first ascends as the length is increased from a small value (e.g., 5), and then drops or stays stable as the length further grows. The optimal length of FCN is around 10. The reason is that, the more the keywords are fed into the neural network, the more the important information it can learn. On the other hand, as the number becomes large, those extracted keywords introduce noise to the model that could hurt the performance.

8.2.5 Case Study. Next, we perform a new set of case studies to further understand the properties of our proposed CrowdTC. Table 5 depicts one positive case and one negative case using *Yelp.P* dataset. Note that, the corresponding crowded-powered keywords are given in italic. There

Table 5. Text Cases with the Crowd-powered Keywords on *Yelp.P* Dataset

Category	Text
Positive	Had the ***special*** pasta *last* night at Mama's and it *was* unreal. Pure breast meat chicken bolognese in a tomato sauce with *perfectly* cooked rigatoni.
Negative	*Pretentious* mall with ***old*** *dated* stores.

Table 6. Accuracy Scores (in Percentage) on Five Datasets

Types	Models	*AG's news*	*Yelp.F*	*Yelp.P*	*Amazon.F*	*Amazon.P*
Feature-based	BoW [67]	88.81	57.99	92.24	54.64	90.40
	BoW-TFIDF [67]	89.64	59.86	93.66	55.26	91.00
	ngrams [67]	92.04	56.26	95.64	54.27	92.02
	ngrams-TFIDF [67]	92.36	54.80	95.44	52.44	91.54
Deep Learning	char-CNN [67]	90.49	62.05	95.12	59.57	95.07
	word-CNN [67]	91.45	60.42	95.40	57.61	94.49
	char-CRNN [62]	91.36	61.82	94.49	59.23	94.13
	D-LSTM [66]	92.1	59.6	92.6	-	-
	VDCNN [9]	91.33	64.72	95.72	63.00	95.72
	Region.emb [44]	92.8	64.9	96.4	60.9	95.3
	BERT [36, 49]	**94.75**	**70.58**	**98.08**	**61.60**	**96.04**
CrowdTC	KA-RNN	92.25 (0.18)	65.54 (0.37)	96.22 (0.24)	60.11 (0.25)	94.85 (0.04)
	$HDNN_C$	92.27 (0.28)	64.24 (0.32)	96.26 (0.40)	56.48 (0.06)	93.22 (0.05)
	$HDNN_R$	93.07 (0.19)	66.72 (0.21)	96.49 (0.08)	58.73 (0.18)	93.95 (0.08)

are two kinds of crowded-powered keywords: one is the keywords selected by the crowd workers from the sampled text; and the other is the expanded keywords based on the clustering method. We further denote the former in bold. It is observed that the selected keywords (e.g., *special* and *old*) and most of the expanded keywords (e.g., *perfectly*, *pretentious*, and *dated*) can guide the deep learning classifier to make the correct classification, although several expanded keywords (e.g., *last* and *was*) are useless for the classification. This confirms that the crowd-powered keywords can play a guiding role in text classification.

8.3 Comparison with State-of-the-art Models

Next, we compare our proposed models with the state-of-the-art models. Table 6 lists the results on five datasets. For ease of discussion, we group all the models in three categories, including feature-based models, deep learning models, and CrowdTC that refers to the models presented in this article. They correspond to the three blocks in Table 6, respectively. The best results in each category are given in underscore, and the overall best records are given in bold. Note that, we run our CrowdTC models 5 times to alleviate the randomness, and report the performance of CrowdTC in average scores with the standard deviation values (in percentage).

The top block lists the performance of four feature-based models. These traditional models achieve a strong baseline accuracy in small datasets (including *AG's news*, *Yelp.F*, and *Yelp.P*), but performs not well in large datasets (i.e., *Amazon.F* and *Amazon.P*).

The second block reports the performance of seven deep learning models. They achieve state-of-the-art performance using deep neural networks. We can observe that BERT performs the best in all five datasets, because BERT is a large-scale bidirectional transformer model that has a significant increase in model complexity. Nonetheless, BERT has its own disadvantage, it requires a large corpus, and introduces both more parameters and pre-training computation.

The last block presents our proposed CrowdTC models. The observation is that our presented models beat all the other competitors on *AG's news*, *Yelp.F*, and *Yelp.P* datasets, except for BERT. This is because we extract the keywords by a cluster-based crowdsourcing, and embrace them as human guidance into the well-designed neural network architecture. This method is efficient to improve the performance of the succinct model to win almost all the competitors, and be closer to the highly sophisticated model, i.e., BERT, which requires a much more complex model structure and a larger corpus. Also, it motivates us to investigate how to guide those BERT-like transformer models with crowd-powered keywords. We would like to explore this interesting direction in future work, considering this article aims at improving the succinct model with crowd-powered guidance.

9 CONCLUSIONS

In this article, we propose crowd-powered learning for text classification, i.e., CrowdTC. To our knowledge, this is the first attempt to use the keywords extracted from the crowdsourcing platform to guide the deep neural networks for text classification. To reduce the monetary cost of hiring the crowd workers, we design a cluster-based crowdsourcing method to extract keywords in text. Moreover, we develop two types of models to incorporate the extracted keywords into deep neural networks, i.e., KA-RNN and HDNN. KA-RNN uses the attention mechanism, and customizes the loss function to emphasize keyword signals. HDNN combines the standard CNN (or RNN) with FCN to capture both the original text and the keyword information. Experimental results demonstrate the effectiveness of our proposed CrowdTC, and verify the power of crowd-based keyword guidance to neural networks. Considering the recently proposed BERT-like transformer approaches achieve the state-of-the-art pre-training formulation for text, we would like to further explore the crowd-powered keyword guidance to those pre-training models in the future.

REFERENCES

[1] Md. Shad Akhtar, Dushyant Singh Chauhan, and Asif Ekbal. 2020. A deep multi-task contextual attention framework for multi-modal affect analysis. *ACM Transactions on Knowledge Discovery from Data* 14, 3 (2020), 32:1–32:27. DOI: https://doi.org/10.1145/3380744

[2] Slobodan Beliga, Ana Mestrovic, and Sanda Martincic-Ipsic. 2016. Selectivity-based keyword extraction method. *International Journal on Semantic Web and Information Systems* 12, 3 (2016), 1–26. DOI: https://doi.org/10.4018/IJSWIS.2016070101

[3] Adrien Bougouin, Florian Boudin, and Béatrice Daille. 2013. TopicRank: Graph-based topic ranking for keyphrase extraction. In *Proceedings of the 6th International Joint Conference on Natural Language Processing*. 543–551. Retrieved from https://www.aclweb.org/anthology/I13-1062/.

[4] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289. DOI: https://doi.org/10.1016/j.ins.2019.09.013

[5] Cornelia Caragea, Florin Adrian Bulgarov, Andreea Godea, and Sujatha Das Gollapalli. 2014. Citation-enhanced keyphrase extraction from research papers: A supervised approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1435–1446. DOI: https://doi.org/10.3115/v1/d14-1150

[6] William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*. 161–175. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.21.3248&rep=rep1&type=pdf.

[7] Yizong Cheng. 1995. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17, 8 (1995), 790–799. DOI: https://doi.org/10.1109/34.400568

[8] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of 8th Workshop on Syntax, Semantics, and Structure in Statistical Translation*. 103–111. DOI: https://doi.org/10.3115/v1/W14-4012

[9] Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. 2017. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. 1107–1116. DOI: https://doi.org/10.18653/v1/e17-1104

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 4171–4186. DOI: https://doi.org/10.18653/v1/n19-1423

[11] Samhaa R. El-Beltagy and Ahmed A. Rafea. 2009. KP-Miner: A keyphrase extraction system for english and arabic documents. *Inf. Syst.* 34, 1 (2009), 132–144. DOI: https://doi.org/10.1016/j.is.2008.05.002

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*. 226–231. Retrieved from http://www.aaai.org/Library/KDD/1996/kdd96-037.php.

[13] Amber Feng, Michael J. Franklin, Donald Kossmann, Tim Kraska, Samuel Madden, Sukriti Ramesh, Andrew Wang, and Reynold Xin. 2011. CrowdDB: Query processing with the VLDB crowd. In *Proceedings of the VLDB Endowment* . 4, 12 (2011), 1387–1390. Retrieved from http://www.vldb.org/pvldb/vol4/p1387-feng.pdf.

[14] Ryan J. Gallagher, Kyle Reing, David C. Kale, and Greg Ver Steeg. 2017. Anchored correlation explanation: Topic modeling with minimal domain knowledge. *Transactions of the Association for Computational Linguistics* 5, 5 (2017), 529–542. Retrieved from https://transacl.org/ojs/index.php/tacl/article/view/1244.

[15] Jiawei Han, Micheline Kamber, and Jian Pei. 2011. *Data Mining: Concepts and Techniques, 3rd Edition*. Morgan Kaufmann. Retrieved from http://hanj.cs.illinois.edu/bk3/.

[16] Kazi Saidul Hasan and Vincent Ng. 2014. Automatic keyphrase extraction: A survey of the state of the art. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 1262–1273. DOI: https://doi.org/10.3115/v1/p14-1119

[17] Swapnil Hingmire, Sandeep Chougule, Girish K. Palshikar, and Sutanu Chakraborti. 2013. Document classification by topic labeling. In *Proceedings of the International Conference on Information Retrieval SIGIR*. 877–880. DOI: https://doi.org/10.1145/2484028.2484140

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780. DOI: https://doi.org/10.1162/neco.1997.9.8.1735

[19] Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 60, 5 (2004), 493–502. DOI: https://doi.org/10.1108/00220410410560573

[20] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1746–1751. DOI: https://doi.org/10.3115/v1/d14-1181

[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*. Retrieved from http://arxiv.org/abs/1412.6980.

[22] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text classification algorithms: A survey. *Information* 10, 4 (2019), 150. DOI: https://doi.org/10.3390/info10040150

[23] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. 2267–2273. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9745.

[24] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J. Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319. DOI: https://doi.org/10.1109/TKDE.2016.2535242

[25] Christopher H. Lin, Mausam, and Daniel S. Weld. 2016. Re-active learning: Active learning with relabeling. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. 1845–1852. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12500.

[26] Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech*, Nelson Morgan (Ed.). ISCA, 685–689. DOI: https://doi.org/10.21437/Interspeech.2016-1352

[27] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. 2011. Crowdsourced databases: Query processing with people. In *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research*. 211–214. Retrieved from http://cidrdb.org/cidr2011/Papers/CIDR11_Paper29.pdf.

[28] Olena Medelyan, Eibe Frank, and Ian H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. 1318–1327. Retrieved from https://www.aclweb.org/anthology/D09-1137/.

[29] Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 323–333. DOI: https://doi.org/10.18653/v1/2020.acl-main.30

[30] Rui Meng, Sanqiang Zhao, Shuguang Han, Daqing He, Peter Brusilovsky, and Yu Chi. 2017. Deep keyphrase generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 582–592. DOI: https://doi.org/10.18653/v1/P17-1054

[31] Yu Meng, Jiaxin Huang, Guangyuan Wang, Zihan Wang, Chao Zhang, Yu Zhang, and Jiawei Han. 2020. Discriminative topic mining via category-name guided text embedding. In *Proceedings of The Web Conference 2020*. 2121–2132. DOI: https://doi.org/10.1145/3366423.3380278

[32] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 983–992. DOI: https://doi.org/10.1145/3269206.3271737

[33] Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2019. Weakly-supervised hierarchical text classification. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. 6826–6833. DOI: https://doi.org/10.1609/aaai.v33i01.33016826

[34] Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2020. Text classification using label names only: A language model self-training approach. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 9006–9017. DOI: https://doi.org/10.18653/v1/2020.emnlp-main.724

[35] Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*. 404–411. Retrieved from https://www.aclweb.org/anthology/W04-3252/.

[36] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep learningbased text classification: A comprehensive review. *ACM Computing Surveys* 54, 3 (2021), 62:1–62:40. DOI: https://doi.org/10.1145/3439726

[37] Barzan Mozafari, Purnamrita Sarkar, Michael J. Franklin, Michael I. Jordan, and Samuel Madden. 2014. Scaling up crowd-sourcing to very large datasets: A case for active learning. In *Proceedings of the VLDB Endowment* 8, 2 (2014), 125–136. DOI: https://doi.org/10.14778/2735471.2735474

[38] Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the 10th international conference on Asian digital libraries: looking back 10 years and forging new frontiers*. 317–326. DOI: https://doi.org/10.1007/978-3-540-77094-7_41

[39] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*. 79–86. DOI: https://doi.org/10.3115/1118693.1118704

[40] Eirini Papagiannopoulou and Grigorios Tsoumakas. 2020. A review of keyphrase extraction. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 10, 2 (2020). DOI: https://doi.org/10.1002/widm.1339

[41] Hyunjung Park, Richard Pang, Aditya G. Parameswaran, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. 2012. Deco: A system for declarative crowdsourcing. In *Proceedings of the VLDB Endowment* . 5, 12 (2012), 1990–1993. DOI: https://doi.org/10.14778/2367502.2367555

[42] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. 1532–1543. DOI: https://doi.org/10.3115/v1/d14-1162

[43] Matt Post and Shane Bergsma. 2013. Explicit and implicit syntactic features for text classification. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics ACL Short Papers*. 866–872. Retrieved from https://www.aclweb.org/anthology/P13-2150/.

[44] Chao Qiao, Bo Huang, Guocheng Niu, Daren Li, Daxiang Dong, Wei He, Dianhai Yu, and Hua Wu. 2018. A new method of region embedding for text classification. In *ICLR*. Retrieved from https://openreview.net/forum?id=BkSDMA36Z.

[45] Douglas LT Rohde, Laura M Gonnerman, and David C Plaut. 2006. An improved model of semantic similarity based on lexical co-occurrence. *Commun. ACM* 8, 627-633 (2006), 116. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.9401&rep=rep1&type=pdf.

[46] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory* 1 (2010), 1–20.

[47] Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis Machine Intelligence* 22, 8 (2000), 888–905. DOI: https://doi.org/10.1109/34.868688

[48] Yangqiu Song and Dan Roth. 2014. On dataless hierarchical text classification. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. 1579–1585. Retrieved from http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8588.

[49] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune BERT for text classification? In *CCL*. 194–206. DOI: https://doi.org/10.1007/978-3-030-32381-3_16

[50] Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1422–1432. DOI: https://doi.org/10.18653/v1/d15-1167

[51] Norases Vesdapunt, Kedar Bellare, and Nilesh N. Dalvi. 2014. Crowdsourcing algorithms for entity resolution. In *Proceedings of the VLDB Endowment* . 7, 12 (2014), 1071–1082. DOI: https://doi.org/10.14778/2732977.2732982

[52] Hanna M. Wallach. 2006. Topic modeling: Beyond bag-of-words. In *Proceedings of the 23 rd International Conference on Machine Learning*. 977–984. DOI: https://doi.org/10.1145/1143844.1143967

[53] Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. 855–860. Retrieved from http://www.aaai.org/Library/AAAI/2008/aaai08-136.php.

[54] Baoxin Wang. 2018. Disconnected recurrent neural networks for text categorization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics ACL*. 2311–2320. DOI: https://doi.org/10.18653/v1/P18-1215

[55] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. 2018. Joint embedding of words and labels for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 2321–2331. DOI: https://doi.org/10.18653/v1/P18-1216

[56] Jin Wang, Liang-Chih Yu, K. Robert Lai, and Xuejie Zhang. 2016. Dimensional sentiment analysis using a regional CNN-LSTM model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics:* 2321–2331. DOI: https://doi.org/10.18653/v1/p16-2037

[57] Sibo Wang, Xiaokui Xiao, and Chun-Hee Lee. 2015. Crowd-based deduplication: An adaptive approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1263–1277. DOI: https://doi.org/10.1145/2723372.2723739

[58] Peter Welinder and Pietro Perona. 2010. Online crowdsourcing: Rating annotators and obtaining cost-effective labels. In *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops CVPR Workshops*. 25–32. DOI: https://doi.org/10.1109/CVPRW.2010.5543189

[59] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the 4th ACM conference on Digital libraries ACM DL*. 254–255. DOI: https://doi.org/10.1145/313238.313437

[60] David H. Wolpert and William G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 67–82. DOI: https://doi.org/10.1109/4235.585893

[61] Tingmin Wu, Shigang Liu, Jun Zhang, and Yang Xiang. 2017. Twitter spam detection based on deep learning. In *Proceedings of the Australasian Computer Science Week Multiconference*. 3:1–3:8. DOI: https://doi.org/10.1145/3014812.3014815

[62] Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. arxiv:1602.00367 Retrieved from http://arxiv.org/abs/1602.00367.

[63] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. XLNet: Generalized autoregressive pretraining for language understanding. In *Proceedings of the 33rd Conference on Neural Information Processing Systems*. 5754–5764. Retrieved from http://papers.nips.cc/paper/8812-xlnet-generalized-autoregressive-pretraining-for-language-understanding.

[64] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies NAACL–HLT*. 1480–1489. DOI: https://doi.org/10.18653/v1/n16-1174

[65] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative study of CNN and RNN for natural language processing. arxiv:1702.01923 Retrieved from http://arxiv.org/abs/1702.01923.

[66] Dani Yogatama, Chris Dyer, Wang Ling, and Phil Blunsom. 2017. Generative and discriminative text classification with recurrent neural networks. arxiv:1703.01898 Retrieved from http://arxiv.org/abs/1703.01898.

[67] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. 649–657. Retrieved from http://papers.nips.cc/paper/5782-character-level-convolutional-networks-for-text-classification.

Received November 2020; revised March 2021; accepted March 2021